Detección de una plataforma para aproximación de vehículos autónomos con redes neuronales convolucionales en visión artificial

José Enrique Porro Domínguez

MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

TRABAJO FIN DE MÁSTER

Jun. de 2021

DIRECTOR: GONZALO PAJARES MARTINSANZ

RESUMEN

Con el presente trabajo se han desarrollado varios métodos válidos para la detección de una plataforma de aproximación para vehículos autónomos, basados en visión artificial, haciendo uso de técnicas de visión por computador y redes neuronales convolucionales (CNN). Para ello, se revisaron los métodos existentes hasta el momento y se realizó un estudio específico de todos los aspectos científicos-técnicos involucrados. Tras ello, se diseñó y elaboró una plataforma propia con características que resolviera los problemas de plataformas de aproximación ya existentes y, además, ayudara a su identificación en una imagen digital.

A partir de la plataforma propia, se desarrollaron y validaron varios métodos para detectar dicha plataforma en una imagen digital utilizando Matlab como herramienta. Los primeros métodos de detección desarrollados combinan técnicas clásicas de visión por computador con redes CNN genéricas. Concretamente se utilizan las técnicas clásicas de visión por computador, como técnicas de extracción de regiones mediante binarización o por color y geometría, para extraer zonas de interés en una imagen digital donde exista la posibilidad de contener la plataforma. Después, con las zonas de interés obtenidas, estos métodos pasan a clasificarlas mediante redes CNN genéricas como AlexNet y GoogLeNet y con los resultados obtenidos, se decide sobre la identificación y ubicación de la plataforma en la imagen. Los segundos métodos de detección de la plataforma propia desarrollados, se realizan mediante modelos de detectores de objetos basados en redes CNN concretamente son los modelos YOLOv2 y Faster R-CNN.

Por otra parte, estos métodos de detección de la plataforma propia han sido validados para poder ser implementados futuramente en vehículos autónomos. Esta validación se ha realizado aplicándoles diferentes procedimientos cuantitativos como, por ejemplo, la matriz de confusión o métrica de Precisión/Recall y también procedimientos cualitativos. De estos procedimientos se han obtenidos unos resultados que posteriormente han sido analizados y comparados entre los diferentes métodos.

Palabras clave

vehículos autónomos, visión por computador, redes neuronales convolucionales, Matlab, AlexNet, GoogLeNet, YOLOv2, Faster R-CNN, matriz de confusión, métrica de Precisión/Recall.

SUMMARY

With the present work we have developed several valid methods for the detection of an approach platform for autonomous vehicles, based on artificial vision, making use of computer vision techniques and convolutional neural networks (CNN). To this end, the existing methods were reviewed and a specific study of all the scientific-technical aspects involved was carried out. After that, we designed and developed our own platform with features that would solve the problems of existing approach platforms and, in addition, help their identification in a digital image.

From the platform itself, several methods were developed and validated to detect said platform in a digital image using Matlab as a tool. The first detection methods developed combine classical computer vision techniques with generic CNN networks. Specifically, classical computer vision techniques are used, such as techniques for extracting regions by binarization or by color and geometry, to extract areas of interest in a digital image where there is the possibility of containing the platform. Then, with the areas of interest obtained, these methods go on to classify them through generic CNN networks such as AlexNet and GoogLeNet and with the results obtained, it is decided on the identification and location of the platform in the image. The second detection methods of the own platform developed, are made by models of object detectors based on CNN networks specifically are the models YOLOv2 and Faster R-CNN.

On the other hand, these detection methods of the own platform have been validated to be able to be implemented in future in autonomous vehicles. This validation has been carried out by applying different quantitative procedures such as, for example, the confusion matrix or Precision/Recall metric and also qualitative procedures. From these procedures have been obtained some results that have subsequently been analyzed and compared between the different methods.

Keywords

autonomous vehicles, computer vision, convolutional neural networks, Matlab, AlexNet, GoogLeNet, YOLOv2, Faster R-CNN, confusion matrix, Accuracy/Recall metric.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

CNN: Redes Neuronales Convolucionales (Convolutional Neural Networks)

CPA: Análisis Parcial Comparativo (Comparative Partial Analysis)

CPU: Unidad central de procesamiento (Central Processing Unit)

CUDA: Arquitectura Unificada de dispositivos de cómputo (Computer Unified Device Architecture)

DL: Aprendizaje profundo (Deep Learnining)

FC: Capa completamente conectado (fully connected)

FCOS: detección de objetos de una etapa (Fully Convolutional One-Stage)

FPN: Caracteristicas de red piramidal (Feature Pyramid Network)

GPS: sistemas globales de navegación por satélites (Global Positioning System)

GPU: Unidad de procesamiento gráficos (Graphics Processing Unit)

INS: sistemas de navegación inercial (Inertial Navigation System)

IoU: Ratio de unión sobre Intersección (Intersection over Union)

LIDAR: Laser Imaging Detection and Ranging.

ReLU: Unidad Lineal Rectificada (Rectified Linear Unit)

R-CNN: Redes neuronales convolucionales basadas en region ("Region-based Convolutional Neural Networks)

R-FCN: Red totalmente convolucional basada en la región (Region-based Fully Convolutional Network)

ROI: Región de interés (Region of Interest)

RPN: Red de propuesta de regiones (Region Proposal Network)

SGD: Gradiente Descendente Estocástico (Stochastic Gradient Descent)

SIFT: Transformación de características invariantes de escala (Scale-Invariant Feature Transform)

SURF: Funciones robustas aceleradas (Speeded-Up Robust Features)

SVM: Máquina de vectores de soporte (Support Vector Machine)

UAV: Vehículos aéreos no tripulados (Unmanned Aerial Vehicle)

UGV: Vehículos terrestres no tripulados (Unmanned Ground Vehicles)

USV: Vehículos de superficie no tripulados (Unmanned Surface Vehicles)

UUV: Vehículos submarinos no tripulados (Unmanned Underwater Vehicle).

YOLO: Solo miras una vez (You Only Look Once)

CAPÍTULO 1. INTRODUCCIÓN

Los vehículos autónomos (Vehículo autónomo, 2021) son aquellos que disponen de sistemas inteligentes capaces de imitar capacidades humanas referentes a la conducción, tales como el manejo y control. Para ello, los sistemas inteligentes implementados perciben el entorno que les rodea, con el fin de actuar aplicando técnicas de conducción en consecuencia al vehículo. Por otra parte, el factor humano se encarga de establecer el destino al cual dirigirse estos vehículos y pueden olvidarse de ejercer algún tipo acción en la conducción.

Dentro de los vehículos autónomos se pueden clasificar en varios tipos dependiendo del lugar donde operen como, por ejemplo, los vehículos terrestres no tripulados o UGV (Unmanned Ground Vehicles), los vehículos de superficie no tripulados o USV (Unmanned Surface Vehicles), los vehículos aéreos no tripulados o UAV (Unmanned Aerial Vehicle) y los vehículos submarinos no tripulados o UUV (Unmanned Underwater Vehicle).

Como ya hemos mencionado, estos vehículos autónomos son capaces de percibir el entorno de su alrededor y navegar por él desde un punto de partida hasta el destino establecido, evitando los obstáculos que puedan encontrarse. Para poder percibir datos del entorno, utilizan tecnologías como sistemas de visión artificial (o visión por computadora) por cámaras, sistemas de navegación inercial (INS, Inertial Navigation System), sistemas globales de navegación por satélites (GPS, GLONASS o Galileo), telémetros láser, ultrasonidos, radar y LIDAR (Laser Imaging Detection and Ranging). A partir de los datos obtenidos del entorno, los sistemas de control implementados construyen los mapas de navegación más óptimos, detectan objetos a evitar en tiempo real y pueden detectar señales que ayuden en la navegación.

Las tecnologías nombradas anteriormente pueden tener algunas limitaciones en ciertas ocasiones. Una de ellas es la pérdida de la señal de los sistemas globales de navegación por satélites (GPS), ya que ésta no cubre todas las regiones del mundo o el ancho de banda del GPS no es la adecuada. Esta limitación se traduce en un problema cuando se necesita obtener una posición y orientación relativa precisas entre el vehículo autónomo y, por ejemplo, una plataforma de aterrizaje en el caso de un UAV, una plataforma para acercarse a un pantalán en el caso de un USV o una plataforma de estacionamiento para el caso de los UGV.

Dicho problema se puede resolver mediante un sistema de visión artificial ya que, a partir de una imagen bidimensional obtenida por una cámara instalada en un vehículo, se puede obtener la posición y orientación relativa con una gran precisión.

El presente trabajo pretende contribuir a la resolución de dicho problema, centrándose en la parte de detección y clasificación de una plataforma de aproximación mediante un sistema de visión artificial. Para ello, se ha hecho uso de técnicas clásicas de visión por computador para detección de la plataforma en una imagen digital y se han clasificado las imágenes mediante técnicas de aprendizaje profundo (DL, *Deep Learnining*) basadas en Redes Neuronales Convolucionales (CNN, *Convolutional Neural Networks*) genéricas. También se han utilizado modelos de CNN de detección de objetos para la detección de la plataforma de aproximación.

1.1 Revisión de métodos existentes

Tras realizar una búsqueda bibliográfica de los métodos desarrollados hasta la fecha para identificación de plataformas en exteriores a partir de sistemas de visión para el aterrizaje de vehículos autónomos de vuelo, se han seleccionado los siguientes para la referenciación del método desarrollado en el presente trabajo.

1 – Sharp y col. (2001): El método propuesto utiliza una figura geométrica compuesta por 2 cuadrados concéntricos (el de mayor tamaño blanco y el interior negro, sobre el que se ubican otros 6 cuadrados blancos, 1 de ellos de mayor tamaño).

La adquisición de las imágenes se realiza desde el UAV, al que se ha incorporado una cámara digital. Las imágenes adquiridas son procesadas usando técnicas de visión artificial tales como el umbral, la segmentación, el etiquetado de los componentes conectados y la detección de puntos de interés, tales como esquinas.

Aplicando algoritmos de optimizaciones lineales y no lineales, estima la posición y velocidad del UAV respecto a la base, de forma que pueda realizar un aterrizaje seguro.

Como principal inconveniente nos encontramos con la simplicidad geométrica y de color de la base, que en ambientes exteriores puede confundirse con otros elementos comunes.

2 - García-Pulido y col. (2017): Proponen la utilización de figuras geométricas no repetitivas para la configuración de la base. Para ello utiliza una base rectangular de papel blanco sobre la que se ubica un círculo negro de gran tamaño y sobre éste, 4 elipses de fondo blanco en la misma posición y distancia (2 en el eje x y dos en el eje y), una de ellas incluye otra elipse menor de color negro, con dos círculos blancos interiores. En el centro del círculo negro mayor nos encontramos con otros 3 círculos concéntricos, alternando blanco y negro.

El procesamiento de las imágenes adquiridas, se realiza mediante técnicas basadas en Análisis Parcial Comparativo (CPA), utilizando descriptores geométricos como el área, ángulos y distancia euclidiana para su identificación, para lo que se aplican técnicas de umbral, segmentación y etiquetado.

La morfología compleja de la figura propuesta se puede reconocer fácilmente en un entorno al aire libre, funcionando correctamente incluso habiendo partes ciegas de la figura. Sin embargo, la utilización de colores blancos y negros, puede generar errores al confundirse con elementos del entorno o por brillos.

3 – Yu y col. (2018): Utilizan diferentes bases con formas geométricas compuestas y variedad de color. Al contrario de los modelos tradicionales, el sistema de detección se basa en la detección de puntos de referencia de un extremo a otro basados en una red neuronal convolucional profunda. Además, proponen una estrategia de implementación separada que lleva a cabo el entrenamiento y la detección de redes neuronales convolucionales en diferentes plataformas de hardware por separado; es decir, una estación de trabajo de unidad de procesamiento de gráficos y un sistema a bordo de un UAV.

El entrenamiento de este modelo de detección basado en CNN consume mucho tiempo con una gran sobrecarga computacional ya que hay que procesar la imagen ajustando la exposición, la saturación y el tono de la imagen. Este problema, se resuelve en parte gracias a la implementación separativa junto con un modelo de detección basado en Yolo y una arquitectura de red neuronal SqueezeNet, logrando la detección de puntos de referencia en tiempo real.

4 – Wubben y col. (2019): El modelo que proponen utiliza bases de aterrizajes con marcadores ArUco, semejantes a los códigos QR pero con menos información. Consiste en un borde negro y un cuadrado de 6x6 de cuadrados más pequeños en blanco y negro.

Para procesar la imagen, se utilizan las propias librerías de ArUco y Open CV para obtener una función que toma la imagen, detecta la información sobre los marcadores y estima la ubicación del marcador respecto al UAV.

Utilizar estas figuras como bases presenta dos inconvenientes: a bajas alturas el marcador no cabe en el campo de visión, mientras que a mayor altitud la imagen puede ser demasiado pequeña. Para solventar eso, se utiliza una base compuesta por marcadores ArUco de diferentes tamaños.

1.2 Motivación para la investigación

Las técnicas basadas en la visión artificial se utilizan ampliamente en sistemas de aterrizaje autónomos de UAV. Estos métodos tienden a detectar plataformas de aterrizaje específicos identificando sus características visuales sencillas, como formas y colores, por lo que carecen de la capacidad de aprender características visuales de alto nivel.

Estas metodologías solo se aplican a plataformas con variabilidad limitada y requieren condiciones ambientales concretas, ya que con cierto grado de iluminación dejan de ser efectivas. Para superar estas limitaciones, se propone un sistema de detección de plataformas basado en una red neuronal convolucional profunda, que le permita ser escalada fácilmente a un número mayor de puntos de referencia diferentes y presente robustez frente a diferentes condiciones de iluminación.

Para abordar este reto, el método desarrollado se basará en las siguientes características:

- Se utilizará una base de papel compuesta por 4 figuras geométricas de diferentes tamaños y colores.
- La complejidad de la figura permite su reconocimiento incluso si una parte es ciega.
- Se utilizarán técnicas clásicas de visión por computador para detectar la plataforma en imágenes digitales.
- Para reducir las necesidades de procesamiento se utiliza CNN genérica previamente entrenada para la clasificación de las imágenes.

- Modelo de detección basado en Yolo y en R-CNN para lograr la detección de la plataforma en tiempo real.

1.3 Objetivos

En este trabajo se han estudiado y realizado diferentes técnicas de detección para una plataforma de aproximación de vehículos autónomos en imágenes digitales. Los objetivos que se establecen son los siguientes:

1. Estudio y desarrollo de diferentes técnicas para la detección y clasificación de objetos usando Deep Learning (DL), basadas en aprendizaje profundo.
2. Identificación de objetos mediante técnicas clásicas de visión por computador.
3. Creación de una base de datos de imágenes propia para el entrenamiento y validación de las CNN.
4. Análisis y comparación entre las diferentes técnicas aplicadas.

1.4 Descripción del trabajo desarrollado

En este proyecto se han desarrollado varias técnicas software en Matlab (2021) para la detección de plataforma de aproximación en imágenes digitales y posteriormente se han comparado los resultados. A continuación, se muestra un listado de las tareas desarrolladas:

1. Revisión de métodos existentes.
2. Estudio de diferentes técnicas clásicas de visión por computador.
3. Estudio de arquitecturas de redes convolucionales genéricas: AlexNet y GoogLeNet
4. Estudio de modelos CNN especializado en detección de objetos en dos fases: Faster R-CNN.
5. Estudio de modelos CNN especializado en detección de objetos en una fase: YOLOv2.
6. Diseños de una plataforma de aproximación.
7. Base de datos de imágenes propia para el entrenamiento y validación de redes utilizadas.
8. Desarrollo de diferentes programas basándonos en los puntos estudiados, donde se han combinado técnicas clásicas de visión para la detección de la plataforma con las arquitecturas de CNN genéricas para clasificarlas. También se han desarrollado varios programas donde se han utilizado los modelos de detectores de objetos basados en CNN para la detección de la plataforma.
9. Realización de diferentes ajustes de parámetros en los programas desarrollados para alcanzar la mejor configuración posible.
10. Evaluación de resultados y comparación entre los diferentes métodos desarrollados.
11. Exposición de diferentes conclusiones alcanzadas y propuesta de posibles trabajos futuros.

1.5 Organización de la memoria

La memoria del presente trabajo se divide en cinco capítulos. A continuación, se describe de forma resumida el contenido de los mismos:

Capítulo 1. **Introducción.** Donde se describe brevemente la motivación que lleva al planteamiento del trabajo, algunos métodos existentes que combinan la visión por computador o redes neuronales convolucionales con formas geométricas para la aproximación de vehículos autónomos y las motivaciones para la investigación de este trabajo. Además, se plantean los objetivos que se quieren alcanzar, se describen los puntos desarrollados para alcanzar los objetivos y se muestra de forma resumida la organización de la memoria.

Capítulo 2. **Marco teórico.** En este capítulo se describen los conceptos teóricos de los procedimientos utilizados en este trabajo. Este se divide generalmente en dos partes donde la primera describe los conceptos de las técnicas clásicas de visión por computador utilizadas para el procesamiento de imágenes y la segunda parte describe los conceptos sobre las redes neuronales utilizadas.

Capítulo 3. **Diseño de la aplicación.** Se describen las tareas realizadas para el desarrollo de los diferentes programas. Además, se explica el diseño de plataforma propia y el desarrollo de la base de datos de imágenes propia.

Capítulo 4. **Resultados.** En este capítulo se describen los procedimientos llevados a cabo para la validación de los diferentes programas desarrollados y se analizan y comparan los resultados obtenidos.

Capítulo 5. **Conclusiones y trabajos futuros.** En este capítulo se exponen las conclusiones obtenidas de las pruebas realizadas y se proponen algunos trabajos futuros relacionados con el presente trabajo.

CAPÍTULO 2. MARCO TEÓRICO

En este capítulo se exponen los conceptos teóricos de los procedimientos utilizados durante el trabajo. Se empezará exponiendo diferentes técnicas clásicas de visión por computación utilizadas para el procesamiento de imágenes y después se analizan algunos conceptos de redes neuronales convolucionales relevantes para este trabajo.

2.1 Modelos de colores

Un modelo de color (Pajares y Cruz, 2007) es una especificación de un sistema de coordenadas 3D donde cada color se representa con un punto en el interior de dicho sistema. De los diferentes modelos de color que existen nos centramos en el de RGB, CMY, YIQ, HSI y HSV.

En el **modelo RGB** cada color se representa por sus componentes espectrales primarias rojo, verde y azul. El modelo se basa en un sistema de coordenadas cartesianas donde las componentes RGB se ubican en los tres vértices. Todos los valores de RGB se encuentran en el rango [0,1] y los colores se forman por la combinación los colores primarios RGB. La figura 2.1 tomada de (Pajares y Cruz, 2007), representa este modelo.

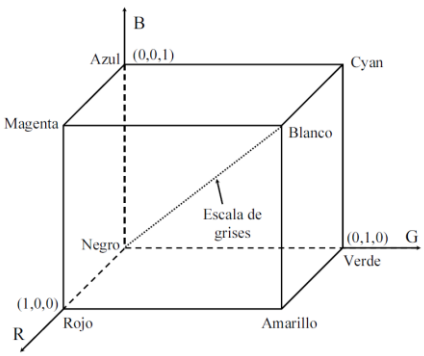


Figura [2.1]. Modelo RGB

El **modelo CMY** corresponde a los colores cyan, magenta y amarillo, que son los colores secundarios de la luz o primarios de los pigmentos. Los valores CMY se obtienen de la recta del vector unidad que representa la luz blanca y el vector de los valores RGB.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

En el **modelo YIQ** la Y corresponde a la reflectancia, la I es una componente cromática llamada infase y la Q es otra componente cromática llamada cuadratura. Las componentes de este modelo se obtienen a partir de una conversión de los valores RGB, como se muestra en la siguiente ecuación.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

En el **modelo HSI** cada color se representa por sus componentes de matiz H (*hue*), saturación S e intensidad I. El matiz representa color dominante, la saturación se refiere al grado en el que el color puro es diluido con luz blanca y la intensidad representa la iluminación percibida. La utilidad de este modelo se debe a la posibilidad de poder separar la componente de intensidad I

de la imagen y la relación de las componentes de matiz y saturación con el modo de percibir el color del sistema visual humano. En la figura [2.2], tomada de (Pajares y Cruz, 2007), se representa el modelo de color HSI.

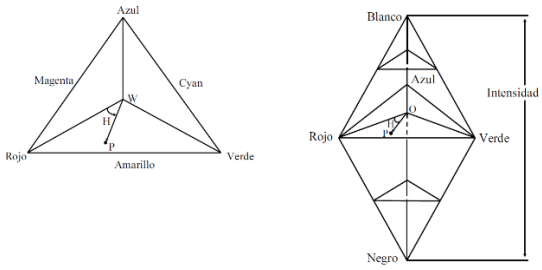


Figura [2.2]. Modelo de color HSI

El **modelo HSV** (matiz, saturación y valor) es similar al modelo de color HSI pero con la diferencia de que para la variación de blanco a negro de la intensidad se utiliza un prisma simple.

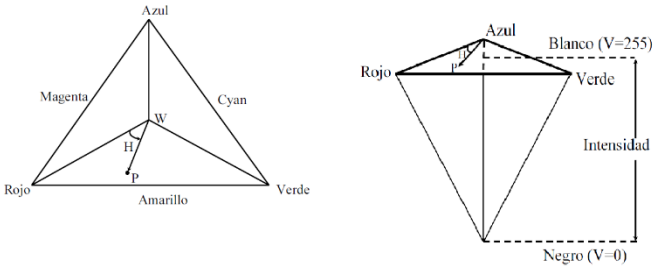


Figura [2.3]. Modelo de color HSV

Transformación de RGB a HSV

El modelo HVS (Método del valor umbral, 2021) es una transformación no lineal del espacio de color RGB en coordenadas cilíndricas.

El valor de las componentes de color (rojo, verde, azul) para el modelo RGB se define en un rango de 0 a 255 donde el 255 es el valor de la componente de color. Por otra parte, los valores del modelo HSV se definen por el matiz que se representa con un ángulo que varía de 0º a 360º, la saturación del color que representa en porcentaje que va de 0% a 100% y el valor que se representa en porcentaje que va de 0% a 100%.

Para calcular los valores de las componentes del modelo HSV a partir del modelo RGB, sé aplican las siguientes fórmulas:

$$H = \begin{cases} no\ definida & si\ \text{Max} = Min \\ 60° * \dfrac{G - B}{Max - Min} + 0° & si\ Max = R\ y\ G \geq B \\ 60° * \dfrac{G - B}{Max - Min} + 360° & si\ Max = R\ y\ G < B \\ 60° * \dfrac{B - R}{Max - Min} + 120° & si\ Max = G \\ 60° * \dfrac{R - G}{Max - Min} + 240° & si\ Max = B \end{cases}$$

$$S = \begin{cases} 0 & si\ Max = 0 \\ 1 - \dfrac{Min}{Max} & en\ otro\ caso \end{cases}$$

$$V = Max$$

Donde las variables R, G, B corresponde cada una de ellas al valor de las componentes de color del modelo RGB (rojo, verde y azul) respectivamente, las variables Max y Min representan respectivamente, el valor máximo y mínimo de las variables R, G, B.

2.2 Extracción de regiones

Las regiones [33] son áreas o zonas en una imagen que se caracterizan por estar formadas a partir de agrupaciones de píxeles conectados entre sí, y además estos píxeles comparten algunas propiedades o características. Las regiones de interés de una imagen se pueden extraer de dicha imagen mediante diferentes técnicas como sería a partir de la binarización de una imagen.

Binarización de una imagen

Con la binarización de una imagen se reduce la información de ésta, con lo que a posteriori facilita la extracción de regiones.

El proceso de binarización consiste en la transformación de una imagen en escala de grises en una imagen binaria, preservando las propiedades geométricas y espaciales esenciales. La imagen binaria es una imagen digital donde cada uno de sus píxeles solo pueden tomar dos valores posibles comúnmente representados con el valor 1 para el blanco y el 0 para el negro.

Uno de los métodos clásicos utilizados para la binarización de la imagen es mediante la detección del umbral T, el cual es un valor perteneciente al rango de niveles de intensidad de una imagen. Este binarizado de imagen consiste en que todos los píxeles con valor menor al umbral T seleccionado se igualen a 0 (negro) y con valor mayor a T se igualen a 1 (blanco).

La selección de un umbral T óptimo es bastante importante para conseguir una binarización adecuada. Existen varios métodos para seleccionar el valor del umbral de forma manual, pero también existen otros como el método Otsu, el cual selecciona de forma automática el umbral T. Cabe decir que, tanto de la forma manual como de la automática, el histograma de intensidad de la imagen es el elemento más importante para obtener el valor del umbral T.

Método de Otsu

El método de Otsu (1979) (Método del valor umbral, 2021) calcula de forma automática el valor óptimo del umbral T. Para ello, este método aplica técnicas estadísticas a los valores de los píxeles recogidos en el histograma de intensidad de una imagen. La técnica que utiliza el método es la varianza, la cual es una medida de dispersión.

El valor del umbral devuelto por dicho método separa los píxeles en dos clases, siendo una clase el primer plano de la imagen y la otra clase el fondo de la misma. Para ello, este valor se calcula de manera que la variación de intensidad dentro de la clase sea lo más pequeña posible, pero además la variación entre clases diferentes sea lo más alta posible.

Etiquetado de regiones por componente conexas

El método de etiquetado de regiones por componentes conexas (Pajares y Cruz, 2007) se aplica a imágenes binarizadas, y consiste en etiquetar las diferentes regiones y asignarle a cada región un valor identificativo. Para ello, se asigna la misma etiqueta identificativa (valor numérico) a

todos los píxeles que están conectados entre sí con valor binario “1”, es decir, a todos los píxeles de una misma región. Estas etiquetas deben ser únicas para cada región por lo que constituye su identificador.

Los algoritmos para este método consisten básicamente en agrupar los píxeles de una región mediante la asignación de una etiqueta. Su funcionamiento se basa en la conectividad bidimensional de píxeles, conectividad-4 y conecitvidad-8. Estos algoritmos van evaluando las filas de una imagen de izquierda a derecha, cuando encuentran un píxel con valor igual a “1” le asigna un valor nuevo de etiqueta e intenta propagar según el tipo de conectividad el valor de dicha etiqueta a los píxeles vecinos que estén a su derecha o debajo. Esta forma de proceder es así porque se empieza a evaluar las filas de arriba a abajo de la imagen y cada fila de izquierda a derecha. En el siguiente ejemplo, se representa una imagen binaria “a” y una imagen “b” que representa el resultado obtenido tras el etiquetado de la imagen binaria para una conectividad-4.

$$a \equiv \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \; b \equiv \begin{bmatrix} 1 & 0 & 0 & 2 & 2 & 0 \\ 1 & 0 & 0 & 2 & 2 & 0 \\ 1 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Extracción de regiones por el color

En este trabajo se utiliza la técnica de extracción de regiones por color (Pajares y Cruz, 2007) basada en el modelo de color RGB. Esta técnica, aplicada en una imagen en color, consiste en extraer de ella aquellas regiones donde predomine una determinada componente de color, como el rojo, verde o azul en el caso del modelo RGB.

2.3 Operaciones morfológicas

La morfología matemática (Pajares y Cruz, 2007) se basa en la forma y la estructura geométrica de los objetos. Las operaciones morfologías aplicadas en una imagen tiene como resultado una simplificación de esta sin perder las principales características de forma de los objetos que contengan.

La morfología se suele utilizar para el tratamiento de regiones en el campo de visión computacional, siendo algunos de esos procesamientos los siguientes:

- Tratamiento para suavizar los bordes en una región.
- Tratamiento para producir una separación de determinadas regiones que a priori se presentan unidas.
- Tratamiento para producir una unión de determinadas regiones que a priori se han separado.
- A partir de aplicar estos tratamientos, en concreto las dos anteriores, elimina ruidos de la imagen facilitando así el computo de regiones.

De las diferentes operaciones morfológicas existentes, en este trabajo se utilizan las transformaciones morfológicas cuantitativas aplicadas en imágenes binarias. Estas transformaciones son exactamente: dilatación, erosión, apertura y cierre.

Un parte importante para las transformaciones morfológicas es el llamado elemento estructural que se utiliza para sondear la imagen de entrada. El elemento estructural para una imagen binaria es otra imagen binaria (en 0 y 1) con una cierta forma y dimensión. El elemento estructural tiene un píxel central el cual identifica el píxel que se está procesando en la imagen de entrada y define la vecindad. En la figura [2.4], tomada de Pajares y Cruz (2007) se muestran algunos elementos estructurales típicos desde el punto de vista binario donde (a) es un cuadrado 3x3, (b) es una cruz y (c) es una línea recta.

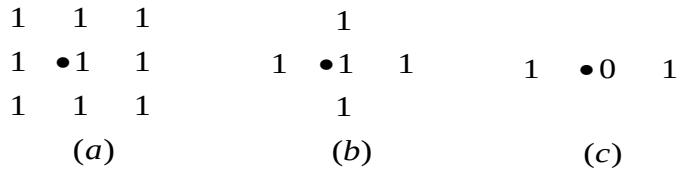


Figura [2.4]. Elementos estructurales

Dilatación

La trasformación morfológica de la dilatación (Pajares y Cruz, 2007) es isotrópica (propiedades idénticas en todas las direcciones espaciales) y su función es agregar píxeles a los límites del objeto, provocando con ello que el objeto sea más grande y compacto al rellenar pequeños agujeros que se puedan encontrar en el interior del objeto. La agregación de píxeles va relacionada con la forma y dimensión del elemento estructural isótropo seleccionado. En ocasiones a la dilatación se le denomina crecimiento o rellenado.

Erosión

La trasformación morfológica de la erosión también es isotrópica y es dual de la dilatación. Su función es eliminar píxeles en los límites de los objetos provocando con ello que el objeto sea más pequeño (disminuye su tamaño) o si tiene un ancho reducido lo elimina. La eliminación de píxeles está relacionada con la forma y dimensión del elemento estructural isótropo seleccionado. En ocasiones la erosión se denomina reducción.

Apertura y cierre

La apertura y cierre (Pajares y Cruz, 2007) se utiliza para eliminar detalles específicos de una imagen, los cuales deben ser más pequeños que el elemento estructural isótropo seleccionado.

La trasformación morfológica de apertura es creada al aplicar una erosión seguida de una dilatación. La apertura en concreto elimina pequeños salientes, así como zonas en objetos grandes que sean más pequeñas que el elemento estructural.

Por otra parte, la trasformación morfológica de cierre es creada al aplicar una dilatación seguida de una erosión. El cierre une los objetos que están próximos entre sí, elimina pequeños huecos rellenándolos y suaviza el contorno de los objetos al rellenar los pequeños huecos. Tanto en la apertura y cierre los conceptos pequeños o próximos están relacionados con la forma y dimensión del elemento estructural seleccionado.

Limpieza

Es operación morfológica de limpieza (Haralick y Shapiro, 1992) consiste en eliminar los píxeles aislados, es decir, píxeles con valor 1 que este rodeado de píxeles con valor 0.

2.4 Redes Neuronales Convolucionales

Una Red Neuronal convolucional (CNN, *Convolutional Neural Network*) es un tipo específico de red neuronal, cuyo fundamento base son las conocidas capas de convolución, que dan pie a su nombre (Pajares y col, 2021). Cualquier modelo de este tipo consta de dos fases: aprendizaje y clasificación. Durante la fase de entrenamiento se ajustan, y en esto consiste el aprendizaje, los denominados pesos del modelo en lo que se conoce como núcleos de convolución. Los modelos así ajustados son los que se utilizan para la clasificación de las diferentes imágenes.

En este trabajo se han utilizado dos modelos de CNN, concretamente el modelo AlexNet y GoogleNet, que se describen a continuación. En cualquiera de estos modelos se realizan diferentes operaciones, con definición de una serie de parámetros, que constituyen la clave del proceso de aprendizaje y clasificación.

2.4.1 Operaciones aplicadas en las CNN

a) Convolución

La convolución es una operación que se define como sigue para el procesamiento de imágenes, que como se sabe son estructuras bidimensionales 2-D,

$$C(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \qquad (2.1)$$

El resultado de la convolución es $C(i,j)$ para un determinado píxel (i,j) cuando la entrada es una imagen I o bien un elemento de un tensor cuando se trata de capas más internas. En la ecuación anterior, K hace referencia a lo que se denomina núcleo de convolución. La aplicación de esta expresión al contexto de las imágenes resulta en el esquema gráfico mostrado en la figura 2.5. El núcleo de convolución K con sus correspondientes valores, se posiciona sobre la cuadrícula superior izquierda para obtener el resultado P que se posiciona en la cuadrícula superior izquierda. A continuación, se desplaza el núcleo una posición hacia la derecha para obtener de forma similar el resultado Q y así sucesivamente se desplaza el núcleo de izquierda a derecha y de arriba hacia abajo hasta completar los valores de las celdas en la matriz resultante.

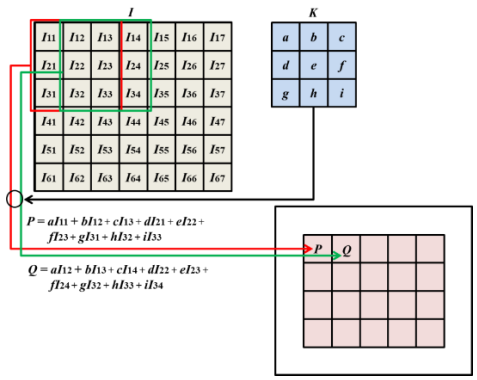


Figura [2.5]. Ejemplo de convolución 2-D

No obstante, si este núcleo en lugar de desplazarse una única celda se desplaza más de una, es necesario establecer lo que se conoce como *stride* (s) que puede tomar valores mayores que la unidad. Además, como el núcleo puede desbordar la imagen o tensor de entrada cuando la celda correspondiente al valor e del núcleo K se sitúa en las filas o columnas más exteriores, existen valores del núcleo que no se utilizan, en cuyo caso estas filas o columnas quedan sin procesar.

Por esta razón, si se desea que el resultado tenga las mismas dimensiones de la imagen, es necesario añadir filas y columnas rellenas de ceros, a esta operación se le conoce como *padding* (*p*). En función de los valores *s* y *p* para una imagen de entrada con dimensión *i* y para valores dados de *p* y *s* se obtiene la correspondiente dimensión de salida *o* según las siguientes relaciones. En estas expresiones *k* es la dimensión del núcleo de convolución, que por ejemplo en el caso del ejemplo anterior, *k* = 3.

$$\textit{Relación 5: } o = \left[\frac{i-k}{s} \right] + 1 \qquad \textit{Relación 6: } o = \left[\frac{i+2p-k}{s} \right] + 1 \tag{2.2}$$

b) ReLU

Esta función conocida como *Unidad Lineal Rectificada* (ReLU, *Rectified Linear Unit*) se define como $f(x) = max(0, x)$, cuya representación es la que se muestra en la figura 2.6, cuyas características y su utilidad se centran en lo siguiente:

a) Evitar la saturación del gradiente durante el proceso de optimización por el método del gradiente descendente.

b) Disminuir la complejidad computacional por el hecho de reducir valores negativos a cero.

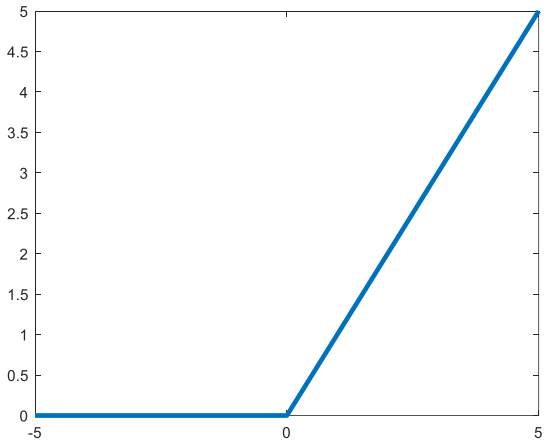


Figura [2.6]. Representación de la función ReLU

c) Normalización

La operación de normalización se aplica con el fin de acelerar la convergencia durante el proceso de aprendizaje. En la expresión siguiente se define una función de normalización por lotes,

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} \left(a_{x,y}^{i} \right)^2 \right)^{\beta} \tag{2.3}$$

donde N es el número de filtros de la capa dada, $a^i_{x,y}$ es la actividad de la neurona y k, α, n, β son hiperparametros. En Krizhevsky (2012) se proponen como más apropiados los siguientes valores para los parámetros $k = 2$, $n = 5$, $\alpha = 10^{-4}$, $\beta = 0.75$.

d) Pooling

Se trata de una función cuyo objeto es agrupar valores en las capas de características, de tal forma que dada una pequeña traslación en la entrada no afecte a los valores de las salidas. En la figura 2.7 se muestra un ejemplo gráfico de agrupamiento por máximos, es decir seleccionando el valor máximo en cada una de las ventanas de agrupamiento, en este caso de dimensión 2x2.

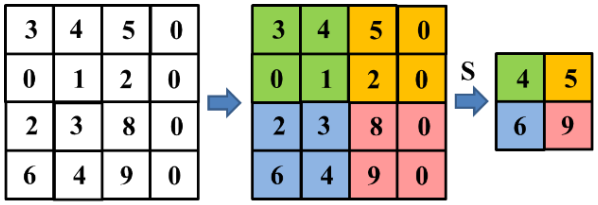


Figura [2.7]. Max pooling

e) Dropout, es un mecanismo para evitar sobresaturaciones en la red neuronal. Consiste en anular determinado tipo de neuronas de forma aleatoria mediante un hiperparámetro que indique la probabilidad de supervivencia de cada neurona.

f) Softmax, consistente en proyectar los valores de la salida en la capa final al rango [0,1], proporcionando así una especia de probabilidad de pertenencia a cada una de las clases para una imagen de entrada dada. Su función se define según la siguiente expresión.

$$softmax(\mathbf{x})_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{2.4}$$

g) Gradiente descendente, que es una técnica de minimización utilizada para el ajuste de los pesos involucrados en los modelos de las redes durante el proceso de retro-propagación. La función a optimizar se conoce como función *objetivo* y cuando se está minimizando, se le denomina también *loss function* o *error function*. Se trata de minimizar una función objetivo que tiene la forma,

$$J(w)=\frac{1}{n}\sum_{i=1}^{n} J_i(w) \tag{2.5}$$

donde el parámetro w que minimiza $J(w)$ debe estimarse, constituyendo el objetivo principal del aprendizaje. J_i se asocia con la i-ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones t,

$$w(t+1)=w(t)-\varepsilon\frac{1}{n}\sum_{i=1}^{n}\nabla J_i(w) \tag{2.6}$$

De esta forma iterativa el método recorre el conjunto de entrenamiento y realiza la actualización anterior para cada muestra de entrenamiento. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo converja. Estas muestras así seleccionadas constituyen

lo que se denomina ***batch*** como se describe más adelante a la hora de seleccionar los parámetros de entrenamiento. En este sentido, es habitual utilizar exactamente la técnica conocida como Gradiente Descendente Estocástico (SGD, *Stochastic Gradient Descent*) (Bishop, 2006; Ruder, 2017).

Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método de gradiente descendente estocástico es como sigue,

1. Elegir un vector inicial de parámetros *w* (puede ser aleatoriamente) y razón de aprendizaje ε .

2. Repetir hasta que se consigue un mínimo aproximado.

   2.1 Seleccionar aleatoriamente ejemplos en el conjunto de entrenamiento.

   2.2 Para *i* = 1,2,..,*n*, hacer $w(t+1)=w(t)-\varepsilon\nabla J_i(w)$.

h) Razón de aprendizaje (*learning rate*): que determina la rapidez según la cual la red actualiza o ajusta los pesos que aprende. Puede ser variable y disminuir según aumenta el número de iteraciones.

i) Adam: cuyo nombre se refiere a Adaptive Moment Estimation (Kingma y Ba, 2015), utiliza el mecanismo de actualización de los pesos según se define seguramente.

$$\begin{aligned} m(t) &= \beta_1 m(t-1)+\left(1-\beta_1\right)\nabla J_i(w(t-1)) \\ v(t) &= \beta_2 v(t-1)+\left(1-\beta_2\right)\left(\nabla J_i(w(t-1))\right)^2 \\ w(t) &= w(t-1)-\frac{\alpha m(t-1)}{\sqrt{v(t-1)}+\varepsilon} \end{aligned} \qquad (2.7)$$

j) Batch *size*: durante el proceso de actualización de los pesos de la red, mediante el cálculo del error y su retro-propagación mediante la técnica del gradiente descendente se dispone de *N* imágenes de entrenamiento en un proceso iterativo. Así, cuando se trata de encontrar el mínimo se realizan distintos pasos, siendo el objetivo de cada uno de ellos conseguir un mejor ajuste. La dimensión de un lote (*batch*) define el número de imágenes utilizadas antes de llevar a cabo una actualización de los pesos de la red en el modelo.

k) *Epochs* e iteraciones: define el número de veces que el conjunto total de muestras son procesadas según el método de optimización.

2.4.2 AlexNet

La red AlexNet (ImageNet, 2021; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2021) como se ha mencionado previamente, es una de las utilizadas en el ámbito de las CNN, ganadora de la competición ImageNet LSVRC (2012) para el reto del año 2012.

En la figura 2.8 se muestra la estructura de la red AlexNet con ilustración gráfica de las operaciones involucradas, identificando las mismas con indicación de las dimensiones de filtros en las capas convolucionales y totalmente conectadas, para las operaciones de Convolución

(*conv*), ReLU (*relu*), Normalización (*norm*), *Pooling* (*pool*), *dropout* (*drop*) o *softmax*. Se incluyen las dimensiones de los filtros, el *stride* (*s*), el *padding* (*p*) y el número de filtros (*K*) en cada capa.

En la tercera columna de la tabla 2.1 se indican los parámetros (pesos) que se aprenden en este modelo, que son los pesos en las capas (primera columna) de convolución (*conv1* a *conv5*) y las totalmente conectadas (*Fully Connected*, fc6, fc7 y fc8), independientemente de que se aplique *dropout* o no en ellas. Obsérvese que como pesos a aprender se incluyen en cada capa de convolución un valor de *bias* por filtro y que es un parámetro de ajuste adicional. Por otra parte, en la tabla 2.1 también se muestran en la segunda columna las dimensiones de salida correspondientes a la capa que se indica.

Nombre de capa	Dimensión de salida	Pesos
conv1	55×55×96	W = 11×11×3×96 b = 1×1×96
conv2	27×27×256	W = 5×5×48×256 b = 1×1×256
conv3	13×13×384	W = 3×3×256×384 b = 1×1×384
conv4	13×13×384	W = 3×3×192×384 b = 1×1×384
conv5	13×13×256	W = 3×3×192×256 b = 1×1×256
fc6	1×1×4096	W = 4096×9216 b = 4096×1
fc7	1×1×4096	W = 4096×4096 b = 4096×1
fc8	1×1×4096	W = 1000×4096 b = 1000×1

Tabla [2.1]. Parámetros aprendidos en el modelo AlexNet

Además, en la estructura de la figura 2.8 se indican los números de *Relación*, que establecen precisamente la relación entre las dimensiones de salida de cada capa (*o*) considerando los parámetros de entrada (*i*, *k*, *p*, *s*), tal y como se explica en el capítulo cuatro, sección 4.2.1, se definen las distintas relaciones posibles, junto con su número correspondiente asociado.

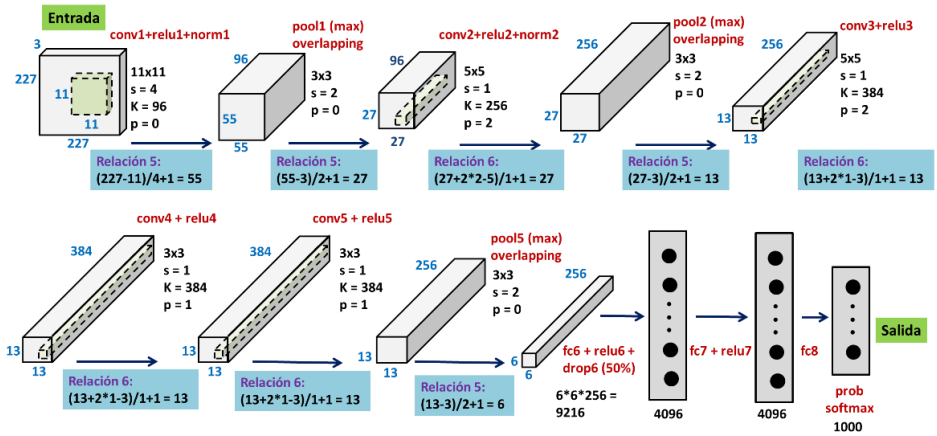


Figura [2.8]. Modelo de red AlexNet

En algunas implementaciones, Matlab (2021), tras la función *relu7* se realiza una operación de *dropout* del 50% modificando las dimensiones de la salida de esta capa.

2.4.3 GoogleNet

La red ganadora del concurso ILSVRC 2014 fue GoogLeNet (también conocida como Inception V1) de Google (BVLC GoogLeNet Model, 2021), proviene de la publicación de Szegedy y col. (2014), habiendo logrado una tasa de error de 6.67%. Se trata de un resultado muy cercano al nivel humano, de modo que los organizadores del desafío necesitaron de la intervención experta. Resulta que, en realidad, la evaluación por parte del experto era bastante difícil de realizar, requiriendo algún entrenamiento adicional para determinar la precisión de GoogLeNet. La red usó se inspiró en LeNet, añadiendo un elemento novedoso denominado módulo *inception*. Un módulo *inception* (Inc) consta de varias convoluciones relativamente pequeñas para reducir drásticamente el número de parámetros. La arquitectura del modelo GoogleLeNet consiste en una CNN de 22 capas de profundidad, que consigue una reducción en el número de parámetros de 60 millones (AlexNet) a 4 millones, de esas 22 capas, 9 son de tipo *inception* de distintas categorías. En total el número de capas es de 144, donde cada una de las 9 capas *inception* contiene la estructura mostrada en la figura 2.9, en la que se han incorporado las unidades ReLU presentes en el módulo. El modelo completo propuesto por Szegedy y col. (2014) es el mostrado en la figura 2.10, donde aparecen las distintas capas: convolución (*Conv*) indicando las dimensiones de los filtros; *Pooling*, bien *average* (*AvgPool*) o *máximo* (*MaxPool*) con indicación en este caso también del tamaño de la ventana; Normalización (*LRN*, *Local Response Normalization*); capas totalmente conectadas (FC) y por supuesto los módulos *Inception* (Inc), en este caso V1 que constituyen la parte nuclear de este tipo de redes. En las capas de convolución y *pooling* se indica también el desplazamiento (*stride*), en este caso se expresa entre paréntesis con el símbolo *s* seguido del correspondiente valor de desplazamiento en el caso '*same*' y V también con el correspondiente valor de desplazamiento, si bien en este caso de tipo '*valid*'. Como puede comprobarse, este esquema presenta dos redes extra, que son exactamente sendos clasificadores auxiliares, y constan de un *pooling* promediado de dimensión 5×5 y *s* = 3 de tipo V que proporciona salidas de tamaños 4×4×512 y 4×4×528 respectivamente. Le sigue una capa de convolución 1×1 con 128 filtros para reducción de la dimensionalidad y una unidad ReLU. Continúa una unidad *dropout* con capas totalmente conectadas finalizando con unidades *softmax* (Softmax0 y Softmax1). La salida final de la red también es una unidad *softmax* (Softmax2).

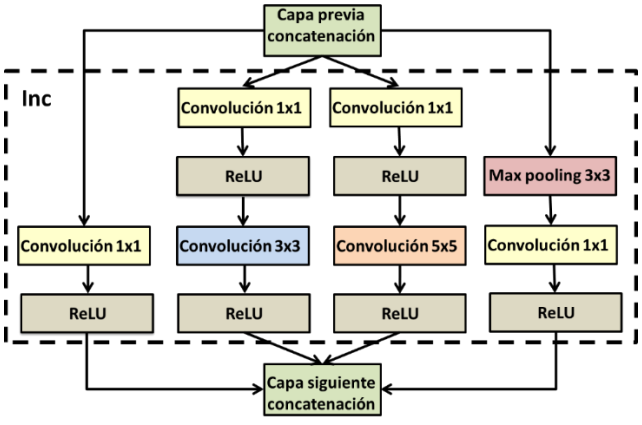


Figura [2.9]. Módulo inception con incorporación de unidades ReLU

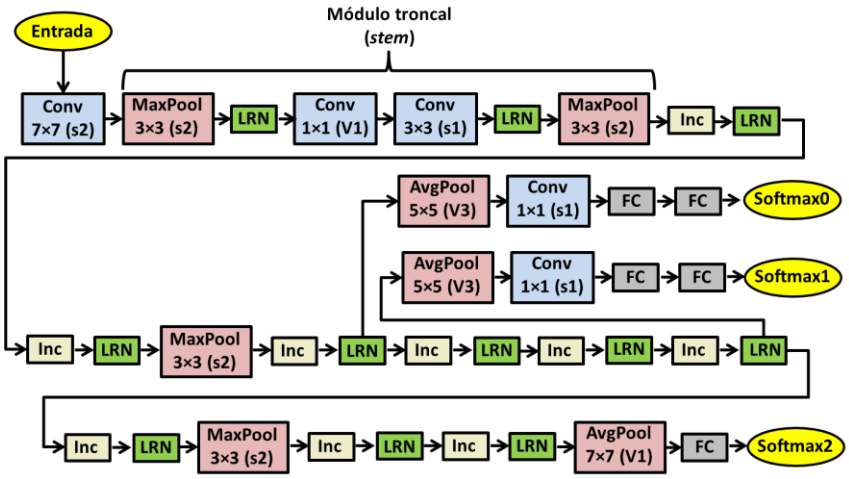


Figura [2.10]. Modelo completo GoogLeNet (inception V1)

2.5 Detección de objetos

Zhao y col. (2019) realizan una revisión de métodos sobre detección de objetos. La idea consiste en determinar dentro de una imagen dada, ciertas zonas de interés (regiones), que se recortan y se proponen para su identificación, obteniendo la probabilidad de que la región contenga un objeto de una determinada clase, que permite seleccionar el objeto de mayor probabilidad.

Generalmente, los detectores de objetos constan de dos partes: una columna vertebral o red troncal y una cabeza, que se usa para predecir clases, además de *bounding boxes* o cajas delimitadoras. Tal y como se indica en Pajares y col., (2021), para la columna vertebral se utilizan distintos tipos de detectores tales como VGG, ResNet, ResNeXt o DenseNet, por ejemplo, fundamentalmente para ejecuciones en GPU y SqueezeNet, MobileNet o ShuffleNet para ejecuciones en CPU. En relación con la parte de la cabeza, se distinguen a su vez dos categorías, concretamente: detector de un estado y de dos estados. Las más representativas de dos estados son las estrategias conocidas como Regiones con CNN (R-CNN) y sus variantes Fast, Faster o Mask o bien R-FCN y Libra R-CNN. También se sitúa en la categoría de dos estados el enfoque RepPoints, que es un detector de objetos sin detección de los *anchors boxes*. En la categoría de un estado se sitúan YOLO, SDD, RetinaNet, OverFeet, así como CornerNet, CenterNet o FCOS que no utilizan *anchors boxes*. Algunos detectores de objetos a menudo insertan algunas capas entre la columna vertebral y la cabeza, y estas capas se usan generalmente para recopilar mapas de características procedentes de diferentes etapas. En la figura 2.11 se muestra el esquema general de detección que contempla las estructuras previamente definidas.

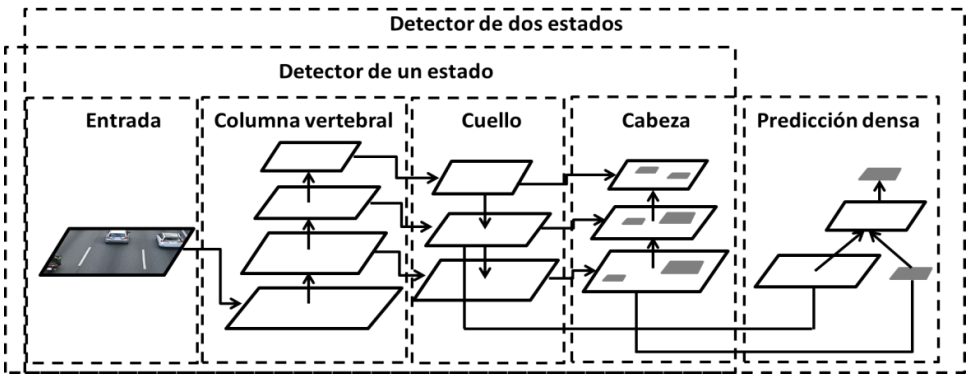


Figura [2.11]. Esquema general de los detectores de objetos

A continuación se proporciona una visión general de diferentes modelos y arquitecturas, según el esquema propuesto por Bochkovskiy y col. (2020), utilizados en las distintas etapas indicadas previamente, sin perjucio, naturalmente, de que se puedan utilizar otras diferentes.

- Columna vertebral: VGG16, ResNet50, ResNeXt, Darknet.
- Cuello: FPN, BiFPN, PANet.
- Cabeza:
  - ✓ Predicción densa (un estado):
    - SSD, YOLO, RetinaNet (basadas en anchors)
    - CornerNet, CenterNet, FCOS (sin anchors)
  - ✓ Predicción dispersa (dos estados):
    - Faster R-CNN, R-FCN, Mask R-CNN (basadas en anchors)
    - RepPoints (sin anchors)

2.5.1 R-CNN y Faster R-CNN

El modelo R-CNN se basa en el trabajo de Girshick y col. (2014). El primer paso consiste en la selección de una serie de regiones con el fin de realizar una propuesta de estas con cierta posibilidad de contener un objeto, figura 2.12. Existen diferentes métodos para realizar la selección, destacando entre otros: *a*) distinción de zonas en la imagen por diferenciación con el entorno circundante, en esta línea se sitúan las conocidas como *objectness* (Alexe y col., 2012); *b*) agrupaciones de regiones por color, textura, geometría o concentración de bordes (Endres y Hoiem, 2010); *c*) agrupación de pequeñas subregiones (*regionlets*), (Wang y col., 2015), basadas en determinar propiedades tales como las proporcionadas por descriptores del tipo SIFT (*Scale-Invariant Feature Transform*) (Lowe, 2004), SURF (*Speeded-Up Robust Features*) (Bay y col., 2008); *d*) agrupaciones de bordes (*edge-boxes*) como en Zitnick y Dollár (2014). Una vez seleccionada la región, esta se redimensiona convenientemente para ser suministrada a una determinada red, por ejemplo, AlexNet, definida previamente, que demanda como entrada una imagen de 227×227×3 píxeles. La CNN obtiene un vector de características (4096) para cada región, por ejemplo, la salida proporcionada por la red en la capa *fc*6. Este vector se utiliza como entrada a un clasificador que determina el tipo de objeto. Como clasificador se puede utilizar la propia red o por ejemplo una Máquina de Vectores Soporte (SVM) (Cherkassky y Mulier, 1998). Los rectángulos de las regiones propuestas se utilizan para estimar los parámetros mediante regresión lineal utilizando las características de la capa *pool*5 de la red AlexNet, tal y como se ha explicado previamente, de manera que el rectángulo del objeto que ha sido clasificado se reajusta para delimitar el propio objeto de forma más precisa. Conviene remarcar cómo en este modelo el modelo CNN realiza un procesamiento por cada región de interés (RoI, *Region of Interest*), que se diferencia de las propuestas que se describen a continuación en este mismo capítulo, donde la CNN procesa la imagen completa una única vez. La propuesta de regiones se realiza mediante cualquier clasificador y una función de optimización definida al efecto.

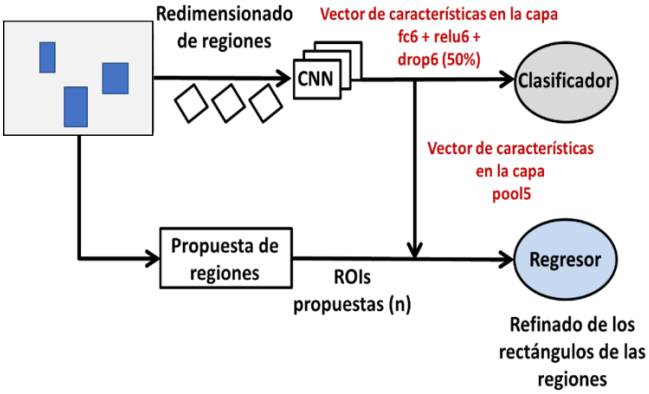


Figura [2.12]. Arquitectura del modelo R-CNN

Desde el punto de vista de la implementación, para crear una red de tipo R-CNN tomando como CNN base un modelo de tipo ResNet-50 (pre-entrenada o no), se reemplazan las tres últimas capas de clasificación con nuevas capas que soporten el número de clases de objetos a detectar, más una clase adicional de fondo. En la ResNet-50, las últimas tres capas se corresponden con una capa totalmente conectada (*fc*), una softmax y una capa de clasificación final para 1000 objetos, tal y como se ha definido previamente, siendo estas las que se desconectan de la red. Más específicamente, un modelo de tipo ResNet (He y col. (2016) (*Residual neural network*) se caracteriza por añadir ramas paralelas en forma de cortocircuitos que saltan varias capas y añaden la identidad a partir de una capa dada y cuya finalidad consiste en mejorar el proceso de optimización en redes muy profundas. Básicamente un bloque ResNet es similar al mostrado en la figura 2.13, de forma que la entrada $\boldsymbol{x}$ se proyecta a la salida (cortocircuito) y se añade a la salida proveniente de otras capas. Consta de cinco módulos cortocircuitados, concretamente C0 (1 capa), C1 (9 capas), C2 (12 capas), C3 (18 capas) y C4 (9 capas), haciendo un total de 49 que junto con la capa FC (*fully connected*) suman las 50.

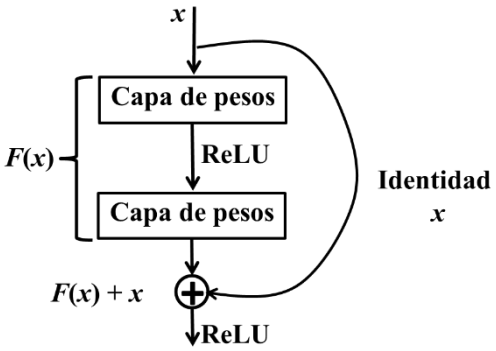


Figura [2.13]. Bloque ResNet

La figura 2.14 muestra la arquitectura para el modelo *Faster* R-CNN desde el punto de vista del entrenamiento y clasificación propuesto por Ren y col. (2017). Es un esquema con una estructura similar al anterior, si bien se añade lo que se denomina *Red para Propuesta de Regiones* que comparte las características convolucionales generadas a partir de la imagen completa. Se trata de una red convolucional que predice simultáneamente la posibilidad de existencia de los objetos y los rectángulos que envuelven a cada uno de ellos. La red RPN se entrena de principio a fin para generar propuestas con la máxima calidad posible. El hecho de compartir capas convolucionales, como se ha indicado previamente, mejora el rendimiento del esquema de detección de objetos.

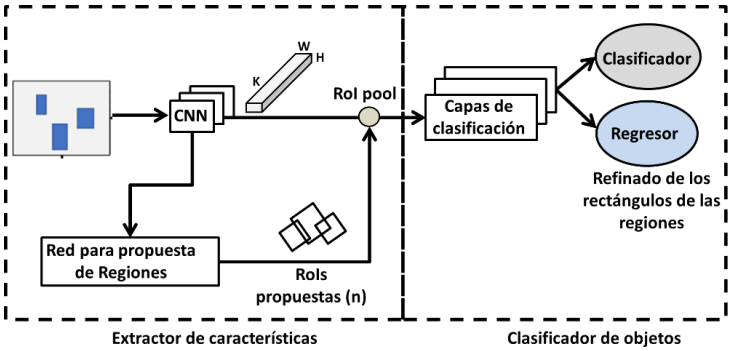


Figura [2.14]. Arquitectura del modelo Faster R-CNN

2.5.2 YOLO

YOLO (*You Only Look Once*) pertenece a la categoría de métodos conocidos como de un estado, como se ha indicado previamente, en los que la imagen completa se procesa una vez, en lugar de procesar regiones por separado, siendo una de sus principales ventajas, frente a otros detectores de objetos, la velocidad de proceso. En una primera etapa se desarrollaron tres versiones de YOLO: *a*) la primera y original (Redmon y col., 2016); *b*) la segunda (Redmon y Farhadi, 2017), tiene como principal objetivo la mejora de la primera, incrementando la velocidad de procesamiento a la vez que evoluciona hacia una mayor robustez. Esta propuesta finaliza con YOLO9000 que realiza un entrenamiento para 9000 clases combinando distintos conjuntos de datos para clasificación y detección mediante WordTree. Huang y col. (2018) proponen YOLO-LITE a partir de YOLOv2 para dispositivos portátiles; *c*) la tercera (Redmon y Farhadi, 2018), cuyo objetivo es mejorar la segunda, apareciendo también PP-YOLO que utiliza como base YOLOv3 e incorpora un modelo de tipo ResNet.

El esquema de la primera versión es relativamente simple. Una única red de tipo CNN predice múltiples *bounding boxes*, junto con sus correspondientes probabilidades de pertenencia a una determinada clase. El entrenamiento se realiza sobre las imágenes completas, no sobre partes de ellas, de ahí su inclusión como modelo de un estado, lo que supone una optimización del proceso de detección. Bajo este planteamiento se consigue una alta velocidad de procesamiento con buenos resultados de precisión, a la vez que se obtienen representaciones generalizables de los objetos.

El procedimiento consiste en dividir la imagen en una rejilla que contiene $S \times S$ celdas. Si el centro de un objeto cae sobre una de tales celdas, dicha celda es la responsable de la detección de ese objeto. Cada celda de la rejilla predice *B bounding boxes* y valores de confianza para esos *boxes*. Los valores de confianza expresan la seguridad del modelo de que el *box* (rectángulo) contenga un objeto y también la seguridad respecto de la precisión. Formalmente se define la confianza como $P(objeto) IoU^{verdadero}_{predicho}$. Si no existe objeto en esa celda, el valor de confianza es cero y en caso de que exista, lo que se desea es que dicho valor sea igual a lo que se conoce como *IoU* (*Intersection over Union*) entre el *box* (rectángulo) predicho y el *ground truth*. Cada *bounding box* consta de 5 predicciones: *x*, *y*, *w*, *h* y la confianza. Las coordenadas (*x*,*y*) representan el centro del rectángulo relativo a los límites de la celda de la rejilla. El ancho (*w*) y alto (*h*) se predicen con relación a la imagen completa. Finalmente, la predicción de confianza representa la *IoU* entre el rectángulo predicho y el *ground truth*. Cada celda de rejilla también predice *C* probabilidades condicionadas de clase que contiene un objeto, $P(C_i | objeto)$.

Solo se predice un conjunto de probabilidades de clase (C_i) por celda de rejilla, independientemente del número de *bounding boxes*, *B*.

En definitiva, el modelo de detección se plantea como un problema de regresión, de forma que la imagen se divide en una rejilla de dimensión $S \times S$ celdas y para cada celda de la rejilla se predicen *B bounding boxes*, junto con los valores de confianza de clase para esos *boxes* y las *C* probabilidades condicionadas de pertenencia a una clase determinada para un objeto dado. Esas predicciones se codifican como un tensor de dimensión $S \times S \times (5B+C)$.

Respecto al diseño de la red, en el trabajo de Redmon y col. (2016) se propone un modelo de arquitectura inspirado en GoogLeNet (Szegedy y col., 2014), descrito previamente. El modelo consta de 24 capas convolucionales seguidas por 2 totalmente conectadas. Si bien, en lugar de los módulos *inception* utilizados en GoogLeNet se usan simplemente capas de reducción 1×1 seguidas por capas de convolución de dimensión 3×3. De esta forma, alternando capas de convolución 1×1 se reducen las dimensiones del espacio de características a partir de las capas precedentes. La red completa se muestra en la figura 2.15, produciendo un tensor a la salida de dimensión 7×7×30.

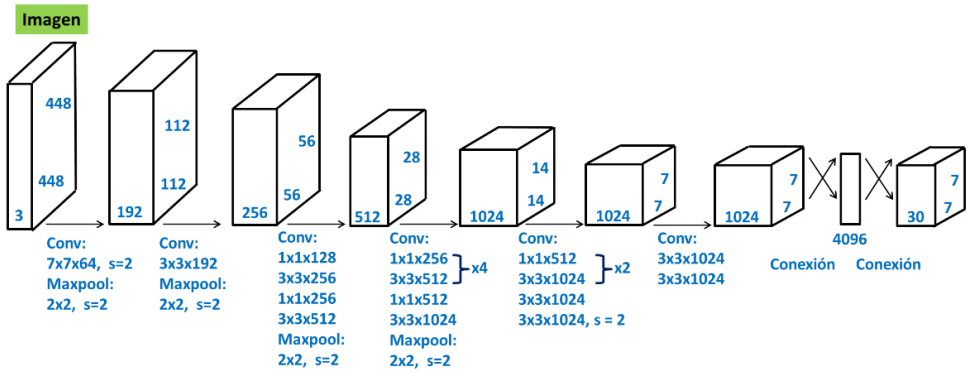


Figura [2.15]. Red neuronal en YOLO

Como se ha mencionado previamente, YOLO predice múltiples *bounding boxes* por cada celda de la rejilla. Con el fin de determinar la pérdida para el verdadero positivo, lo ideal es que solo uno de ellos sea el responsable de identificar el objeto. Por tal motivo, se selecciona el de mayor *IoU* con el *ground truth*. Esta estrategia conduce a la especialización entre las predicciones del *bounding box*. Cada predicción mejora las siguientes en cuanto a dimensiones y razones de aspecto. YOLO utiliza la suma del error cuadrático entre las predicciones y el *ground truth* para calcular la pérdida. La función de pérdida se compone de:

a) *Pérdida de clasificación*.

b) *Pérdida de localización* (errores entre el *bounding box* predicho y el *ground truth*).

c) *Pérdida de confianza* (respecto al contenido del objeto).

Como se ha indicado, previamente, a partir de la primera versión de YOLO se llega a YOLOv2 que es utilizada en este trabajo, cuyas características se resumen básicamente en lo siguiente:

a) Mejoras relativas a: las operaciones de normalización, la predicción de los *bounding boxes* a través de los *anchor boxes*, predicción sobre un mapa de características y entrenamiento multi-escala.
b) Rapidez mediante la utilización de un modelo de red específico de tipo DarkNet-19 (Redmon, 2016), con 19 capas convolucionales y 5 de *maxpooling*.

CAPÍTULO 3. DISEÑO DE LA APLICACIÓN

En este capítulo se describen distintas tareas previas y el desarrollo de las diferentes aplicaciones informáticas realizadas en este trabajo. El desarrollo de estas aplicaciones ha sido mediante el software MATLAB R2019b.

Las tareas previas han consistido en el diseño de la plataforma de aproximación y la generación de una base de datos propia de imágenes.

Respecto a las aplicaciones, se han desarrollado seis métodos para la identificación de la plataforma de aproximación, basándose todos ellos en modelos de redes neuronales convolucionales. Cuatro de estos métodos se basan en la combinación de técnicas clásicas de visión por computador con redes convolucionales genéricas, siendo estas el modelo AlexNet y el GoogLeNet. Otro método se basa en redes convolucionales específicas para la detección de regiones en dos fases, como sería la Faster R-CNN. El último método se basa en redes convolucionales específicas para la detección de regiones en una sola fase, como sería la YOLOv2.

3.1 Plataforma y base de datos propia

Para el desarrollo de este trabajo se ha diseñado una plataforma propia para la aproximación de vehículos autónomos. Esta plataforma se ha diseñado teniendo en cuenta una serie de características con las cuales se intentan cubrir algunas deficiencias presentes en plataformas ya utilizadas en otros métodos existentes, tal como se menciona previamente.

La plataforma tiene forma cuadrada con fondo blanco y en su interior contiene 4 figuras geométricas de diferentes tamaños y colores. Los colores elegidos para cada figura han sido el negro y las tres componentes del modelo de color RGB (rojo, verde y azul). Se optó por utilizar diferentes colores para obtener cierta ventaja a la hora de segmentar y distinguir mejor su naturaleza. En cuanto a la forma geométrica de las figuras, la de color rojo y la de color negro tienen forma de semicírculo y se sitúan cada una de ellas en un lateral de la plataforma, con lo que ayuda a diferenciar la propia plataforma en la naturaleza. La figura de color azul tiene forma triangular con la intención de marcar una orientación a los vehículos autónomos. Por último, la figura de color verde tiene forma circular con un vaciado en su interior con forma cuadrada. La intención de esta figura es indicar al vehículo autónomo el centro de la plataforma. Aparte de lo indicado sobre las figuras, el hecho de utilizar figuras complejas permite su reconocimiento incluso si una parte de la plataforma resulta ciega, esto es, parte de la misma no aparece en la imagen captada. En la siguiente figura 3.1 se muestra la plataforma donde se puede observar la localización de cada figura en su interior y el tamaño de cada una en el interior de la plataforma, siendo la figura negra la de mayor tamaño y, sucesivamente, la figura roja, después la azul y por último la verde. La plataforma diseñada se adjunta con mayor resolución en el ANEXO III.

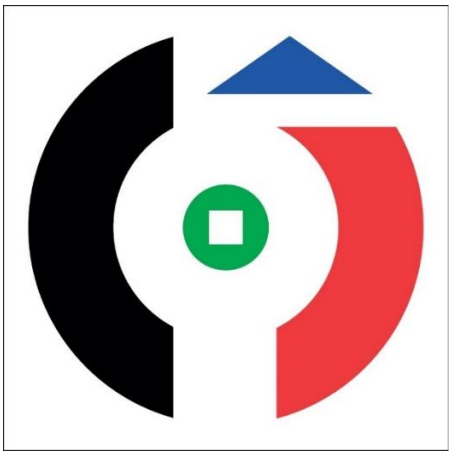


Figura [3.1]. Plataforma de aproximación propia

Dicha plataforma se ha impreso en papel con unas medidas de 300x300mm para poder realizar a posteriori una base de datos de imágenes propia.

3.1.1 Base de datos de imágenes propias

En el presente trabajo necesitamos que las CNN utilizadas clasifiquen imágenes. Esto requiere que las CNN realicen un aprendizaje supervisado, es decir, el sistema va a aprender a partir de las clases o categorías que se le proponga. Esto implica que el proceso de entrenamiento y validación de las CNN requiera de una base de datos de imágenes etiquetadas y separadas por clases o categorías.

Las bases de datos de imágenes se pueden conseguir a partir de diferentes plataformas que proporcionan repositorios de imágenes etiquetadas y clasificadas, o bien generando el propio repositorio de imágenes, etiquetándolas manualmente y separándolas por clases.

En el caso de este trabajo, el objetivo es la clasificación de la plataforma de aproximación propia, por lo que se ha requerido generar conjuntos de datos exclusivos para el entrenamiento y validación de las CNN. Las características de la base de datos de imágenes propia van a diferir en el modo de etiquetado, tamaño de imágenes o número de clases según la CNN a entrenar, pero todas ellas van a tener en común la clase “Plataforma” tanto en número de elementos como en el cuadro del etiquetado o del recorte.

La clase principal “Plataforma” cuenta con un conjunto de 188 imágenes de la plataforma propia diseñadas en diferentes entornos, realizadas con diferentes cámaras fotográficas. En la figura 3.2 se muestran algunas de las imágenes mencionadas.

Figura [3.2]. Conjunto de imágenes de la clase “Plataforma”

Tal y como ya se ha mencionado, los datos de entrenamiento van a diferir en características según las diferentes modelos de CNN a entrenar y evaluar.

Para las arquitecturas CNN genéricas (en nuestro caso AlexNet y GoogLeNet), la base de datos de imágenes se divide en tantas carpetas como clases a entrenar, donde se almacenan sus imágenes. Para este trabajo se han utilizado ocho clases (Agua, Asfalto, Carril Bici, Césped, Hierbas-Flores, Plataforma, Rosas y Tierra) con diferentes números de elementos como se muestra en la tabla 3.1. Para ambas CNN el etiquetado en la imagen de la plataforma o cualquier otra clase se realiza mediante recorte de la misma. Las imágenes recortadas, antes de entrenar las redes, hay que redimensionarlas a la dimensión que requiere dicha red. En el caso de la AlexNet a 227x227x3 y para la GoogLeNet a 224x224x3.

Clase	**Agua**	**Asfalto**	**Carril Bici**	**Césped**	**Hiervas-Flores**	**Plataforma**	**Rosas**	**Tierra**
Nº elemetos	48	45	37	40	38	188	40	60

Tabla [3.1]. Base de datos propia

Por otra parte, los modelos de CNN para la detección de objetos, como la YOLOv2 y la Faster R-CNN utilizados en este trabajo, solo hacen uso de la clase principal “Plataforma”, ya que es el objeto a detectar y las restantes clases de la tabla 3.1 constituyen partes del entorno. Además, tras varias pruebas de entrenamiento con todas las clases, estos modelos de redes no lograban clasificar la plataforma en múltiples ocasiones. Esta clase, como ya se ha mencionado, cuenta con 188 imágenes donde la plataforma es etiquetada manualmente en todas las imágenes mediante la APP de Matlab, “Image Labeler”. Para ambas redes se utiliza el mismo modelo base de CNN genérica (la ResNet-50). Por tanto, se requiere que la imagen de entrada a la red se redimensione a 224x224x3.

Los recortes de la clase principal “plataforma” para la base de datos de imágenes adecuadas a la CNN genérica AlexNet y GoogLeNet, se realizan mediante un script desarrollado en Matlab, el cual parte del etiquetado realizado en la APP “Image Labeler” para los modelos de CNN detectores. Este script realiza los recortes en la imagen original siguiendo el etiquetado realizado en ella. Con esto conseguimos que el etiquetado o recorte de la plataforma sea el mismo para todas las redes entrenadas. Además, este script, tras realizar los recortes de la plataforma, redimensiona todas las clases a la dimensión de entrada de ambas redes.

Tras obtener los conjuntos de imágenes propias adecuadas para la arquitectura de CNN que se va a entrenar, el siguiente paso que se aplica para la preparación de los datos consiste en realizar un aumento de datos (*data augmentation*).

El proceso de aumento de datos se aplica tanto al conjunto de datos para el entrenamiento como al de validación. Este consiste en aumentar y modificar automáticamente dichos conjuntos, mediante transformaciones aleatorias a las imágenes originales. Dichas transformaciones son opcionales y pueden ser cambios de tamaño, rotaciones, traslaciones o la inserción de ruido. La aplicación de este proceso proporciona al aprendizaje de la red variaciones de las imágenes originales, ayudando a conseguir un mejor entrenamiento con un menor número de imágenes originales.

La última fase de preparación de datos consiste en asignar la proporción de datos de entrenamiento y de validación. El 80% de los datos se asignan para el entrenamiento y el 20% restante para la validación. Cuando hay más de una clase, estos porcentajes son los que se asignan a los datos de cada una de ellas.

3.2 Identificación mediante Red Convolucional: AlexNet y GoogLeNet

En este apartado se describe la aplicación desarrollada para identificar la plataforma mediante redes convolucionales genéricas, las cuales, como se ha mencionado previamente son AlexNet y GoogLeNet. Cabe mencionar que los procedimientos descritos son válidos para ambas redes solo con utilizar su adecuada base de datos de imágenes propias.

Debido a que este tipo de redes convolucionales genéricas son solo de clasificación, la detección de regiones de interés (objetos) se realiza mediante técnicas o algoritmos clásicos de visión por computador.

El procedimiento para estos métodos parte de una imagen de entrada dada. Con dicha imagen, el primer paso es procesar la imagen completa para identificar zonas en las cuales existan sospechas fundadas de contener la plataforma (obtener propuestas de regiones de interés). Seguidamente, una vez localizadas las regiones que pueda contener la plataforma, éstas son pasadas a la red neuronal para ser clasificadas. Para que la red pueda realizar la clasificación de la plataforma, ésta debe ser previamente entrenada y validada con la base de datos de imágenes propias. La clasificación de la región propuesta mediante la red devuelve una etiqueta y la probabilidad de que dicha región propuesta pertenezca a una clase. Por último, tras clasificar todas las regiones propuestas, el modelo decide cuál de ellas es la plataforma de aproximación y la etiqueta en la imagen original. El proceso global se sintetiza en la figura 3.3.

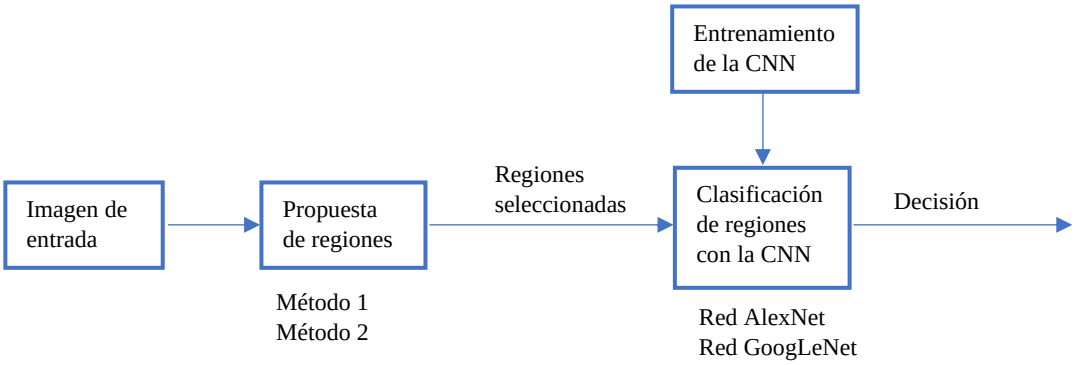


Figura [3.3]. Proceso global de los métodos de detección mediante CNN genéricas

Según el esquema de la figura 3.3 se puede deducir que, con la combinación de los dos métodos de propuesta de regiones desarrollados con las dos redes CNN genéricas utilizadas, obtenemos un total de cuatros métodos para la detección de la plataforma.

3.2.1 Propuesta de Regiones

En este apartado nos centramos en la automatización del proceso para detectar zonas donde pueda encontrarse la plataforma en una imagen digital. Para tal fin, se realiza un procesado de la imagen mediante diferentes técnicas clásicas de visión por computador.

Para identificar la región que contiene la plataforma, partimos de que la plataforma es conocida, es decir, conocemos los colores, tamaños y las distintas formas que contiene, por lo que se aplican diferentes técnicas, las cuales son válidas para identificar la plataforma en la imagen.

Una vez identificadas las diferentes regiones en las cuales existe sospecha de encontrarse la plataforma, esta región es recortada sobre la imagen original para posteriormente pasarla a la red neuronal. Se han desarrollado 2 métodos de propuesta de regiones basados en la segmentación de imágenes orientada a regiones.

- Método 1: Extracción de regiones por binarización basada en el uso de umbral.
- Método 2: Extracción de regiones por color y geometría.

3.2.1.1 Método 1 de propuesta de regiones

Por tanto, partiendo del conocimiento de la plataforma, sabemos que el fondo de la misma es de color blanco, con lo el que objetivo de este método es buscar zonas o regiones con un alto contenido de blanco en una imagen que pueda representar dicho fondo. De forma resumida, este método se basa en las siguientes técnicas para la extracción de regiones a partir de una imagen digital de entrada:

1) Binarización de la imagen basado en el uso de umbral, seleccionado mediante el método de Otsu.
2) Aplicación de operaciones morfológicas de apertura-cierre y limpieza.
3) Etiquetado de regiones por componentes conexas.
4) Selección y recorte de las regiones a partir de sus propiedades.

A continuación, se detalla el método a la vez que vamos mostrando las diferentes técnicas en una imagen real de la plataforma.

El proceso comienza a partir de una imagen de entrada, la cual contiene la plataforma a identificar. Estas imágenes de entrada siempre están representadas en el espacio de color RGB. La figura 3.4 muestra una imagen de entrada con la plataforma de aproximación en un suelo de baldosas de pizarra.

Figura [3.4]. Imagen original

El primer proceso consiste en pasar la imagen original en espacio de color RGB a escala de grises, eliminando así la información de los tonos (o matiz) y la saturación, mientras mantenemos la luminancia, figura 3.5.

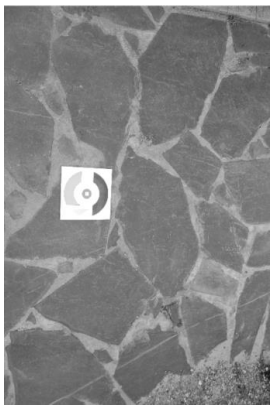


Figura [3.5]. Imagen en escala de grises

El siguiente paso es reducir información de la imagen, con el fin de facilitar la búsqueda de regiones que contenga la plataforma, para ello se realizará el proceso de binarización. Este proceso transforma una imagen en escala de grises a una imagen binaria. La técnica de binarizado de la imagen es mediante establecimiento de un umbral T. El valor de dicho umbral T se necesita obtener para este método de forma automática. Para ello se aplica el método de Otsu. Como resultado, las regiones blancas en la imagen binaria tienen la posibilidad de ser la plataforma, ya que esta tiene la base blanca. La figura 3.6 es la imagen binaria resultante al aplicar el proceso de binarización a la imagen en escala de grises previa.

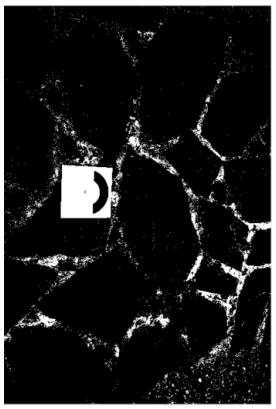


Figura [3.6]. Imagen binaria

Como se puede observar en la imagen binaria, la región blanca que predomina es la plataforma, pero también se observa una gran cantidad de pequeñas regiones blancas que se consideran ruido. Con el fin de eliminar esas pequeñas regiones y conseguir una imagen más tratable, se aplican varias operaciones morfológicas binarias tales como apertura-cierre y limpieza.

En este trabajo, primero se ha aplicado la operación apertura a la imagen binaria y la operación de cierre a la imagen resultante de la apertura, según se muestra en la figura 3.7. En ambos casos se ha seleccionado el mismo elemento estructural, el cual tiene forma de disco con un radio de 11 píxeles desde el píxel central.

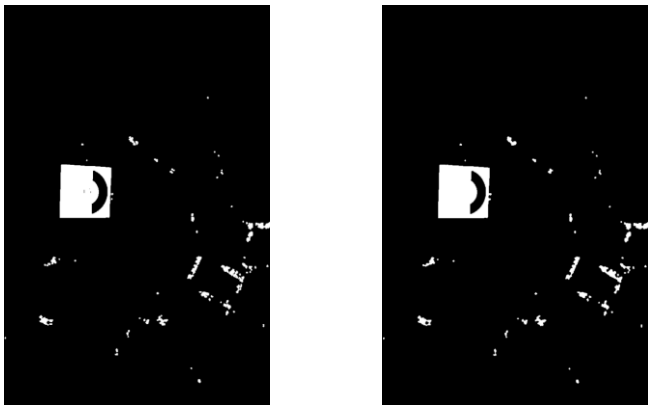


Figura [3.7]. Imagen binaria tras la apertura e imagen binaria tras el cierre

A la imagen resultante del cierre, se le aplica la operación de limpieza para eliminar los píxeles aislados, figura 3.8.

Figura [3.8]. Imagen binaria limpia

El siguiente paso es el etiquetado e identificación de las regiones con posibilidad de contener la plataforma. Para realizar este proceso se aplica sobre la imagen binaria limpia el método de etiquetado de componentes conexas. Este método determina las diferentes regiones y le asigna a cada región un valor identificativo, figura 3.9.

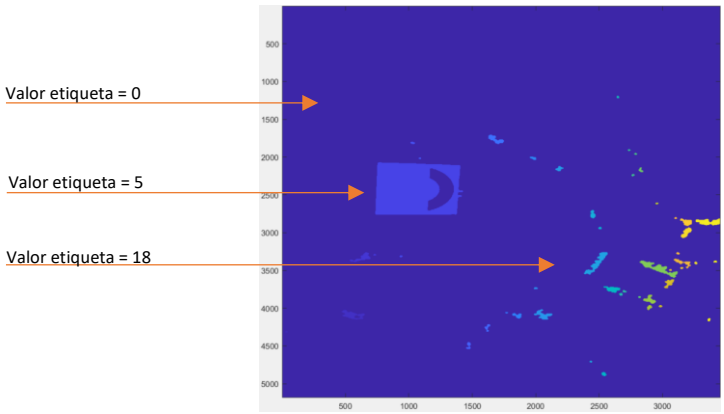


Figura [3.9]. Etiquetas de regiones

De cada región etiquetada podemos obtener varias propiedades, de las cuales para este método se van a utilizar el área de la región y el rectángulo que la delimita. La propiedad de área se utiliza para seleccionar regiones con un área mayor a un valor establecido, suponiendo que por encima de ese valor es posible que sea la plataforma. El tamaño mínimo del área se determinará

en un futuro de forma automática, dependiendo de una distancia aproximada a la que pueda encontrarse el vehículo autónomo de la plataforma, para ello se debe hacer un calibrado del sistema de visión como se indica en el apartado de trabajos futuros del capítulo 5. Mientras tanto, para el desarrollo de esta aplicación se ha establecido un valor considerado el adecuado para las pruebas.

Con las regiones que han sido seleccionadas por el tamaño de área, utilizamos para cada una de ellas la propiedad que nos proporciona el rectángulo que delimita la región, para realizar un recorte de la misma en la imagen original. Dicho rectángulo delimitador se construye a partir de los valores que devuelve esta propiedad, los cuales son: el punto de inicio del rectángulo (que es su esquina superior izquierda en coordenadas x e y), la distancia del ancho del rectángulo, que es paralelo al eje horizontal (eje x) y la distancia del largo del rectángulo, que es paralelo al eje vertical (eje y).

En los recortes resultantes cabe la posibilidad de que exista la plataforma, por lo tanto, son pasados a la red neuronal para ser clasificados, figura 3.10.

Figura [3.10]. Recorte en la imagen original de la región con etiqueta = 5

3.2.1.2 Método 2 de propuesta de regiones

Al igual que en el método anterior, conocemos la plataforma y sabemos que además de tener el fondo blanco, contiene varias figuras con diferentes formas geométricas y color. El objetivo de este método es extraer zonas o regiones de la imagen por el color considerando el modelo de color RGB, y teniendo en cuenta además la geometría de estas regiones, lo que permitirá obtener posibles cuadros delimitadores de la plataforma. De forma resumida, en este método se han aplicado las siguientes técnicas sobre cada imagen digital de entrada:

1) Separar las componentes RGB (rojo, verde, azul) de una imagen, obteniendo nuevas imágenes de grises (una por cada canal de color).
2) Ajuste de contraste a las imágenes de grises obtenidas.
3) Binarización de las imágenes ajustadas de cada canal mediante un umbral (Otsu) y condicionadas por el resto de los canales de color.
4) Aplicación de operaciones morfológicas de apertura-cierre y limpieza a las imágenes binarias resultantes de cada canal.
5) Etiquetado de regiones por cada canal de color mediante componentes conexas.
6) Selección de regiones y recorte de posibles rectángulos delimitadores de la plataforma a partir de las propiedades de las regiones.

A continuación, se detallan los pasos del método a la vez que se muestran, con carácter ilustrativo, los resultados de aplicar las técnicas sobre una imagen real conteniendo la plataforma.

Partiendo de una imagen de entrada en el espacio de color RGB donde encuentra la plataforma a identificar, y con el fin de comparar ambos métodos desarrollados, se utiliza como ejemplo la misma imagen de entrada que en el método 1. En la imagen de la figura 3.11 se muestra la plataforma en un suelo de baldosas de pizarra.

Figura [3.11]. Imagen original

El primer paso consiste en extraer las regiones por color basándonos en el modelo de color RGB. Se trata de extraer aquellas regiones donde predomine una determinada componente de color, como el rojo, verde o azul como colores primarios en el caso del modelo RGB. Se sabe que la plataforma de aproximación contiene cuatro figuras internas de las cuales una figura es de color rojo, otra de color verde y otra de color azul. Por tanto, al aplicar estas técnicas en una imagen que contenga dicha plataforma, se van a extraer un conjunto de regiones rojas de las cuales una será la figura roja, otro conjunto de regiones verdes, donde una de ella será la figura verde y otro conjunto de regiones azules, donde una de ella será la figura azul.

Empezamos dividiendo la imagen original RGB en sus tres canales de color rojo, verde y azul, obteniendo una imagen en escala de grises por cada canal. En las imágenes resultantes para cada canal, los píxeles con niveles de grises más claros corresponden a los píxeles de entrada con mayor valor en sus componentes de color. En la figura 3.12 se muestran las componentes espectrales de la imagen original correspondientes a los canales rojo (izquierda), verde (centro) y azul (derecha). Se puede observar, por ejemplo, en la imagen del canal rojo, que la figura de color rojo de la plataforma tiene niveles de grises más claros, es decir, tiende a blanco. Lo mismo sucede en los demás canales para sus correspondientes colores.

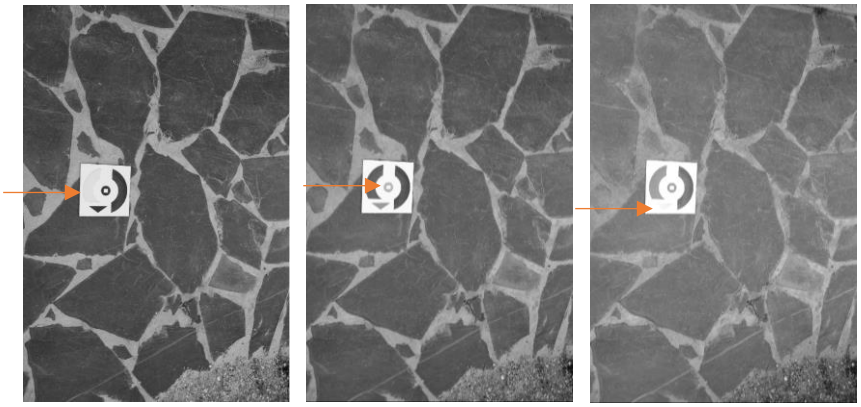


Figura [3.12]. Componentes de color correspondientes a los canales Rojo, Verde y Azul

Para conseguir una mayor diferencia entre los niveles de intensidad de grises, se realiza un ajuste de contraste a las imágenes de los tres canales, figura 3.13.

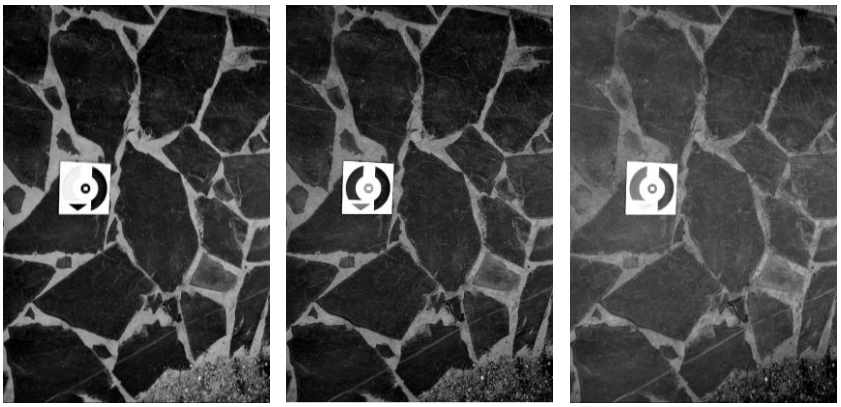


Figura [3.13]. Ajuste de contraste en los canales Rojo, Verde y Azul

Una vez ajustado el contraste, se procede a la binarización de las imágenes de cada canal con el fin de reducir la información en ellas y extraer las regiones correspondientes a cada color. En este caso, al querer extraer las regiones por color, el binarizado de la imagen de cada canal consiste en comparar el valor de intensidad de cada uno de sus píxeles con el valor de su umbral T y el valor intensidad del píxel, con su misma posición en los otros dos canales. Para ello, se establece el umbral T de forma automática para cada imagen de los diferentes canales mediante el método de Otsu. Seguidamente, se realiza el binarizado de cada imagen, donde todos los píxeles con valor de intensidad superior a su umbral T y al valor del píxel con su misma posición en las otras dos imágenes de canales, se le asigna el valor 1 (blanco) y en caso de que no cumpla alguna condición, se asigna el valor 0 (negro). En la figura 3.14 se muestran las imágenes binarias y las regiones correspondientes por cada color.

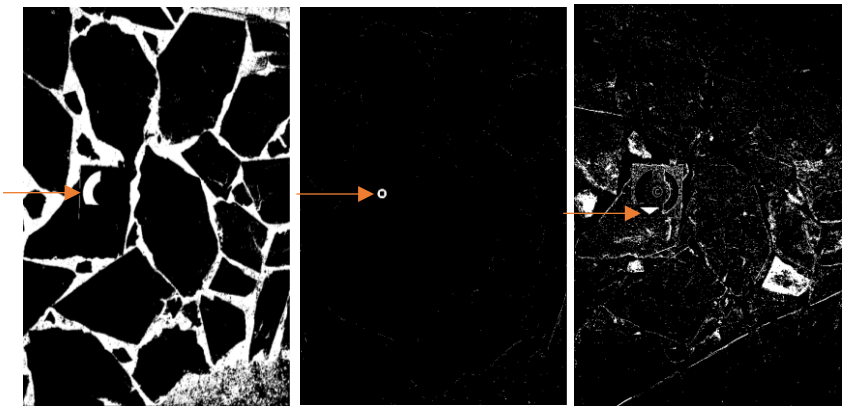


Figura [3.14]. Binarización de imágenes en los canales Rojo, Verde y Azul

En las imágenes binarias de las regiones de las distintas componentes de color, se puede observar que una de las regiones blancas es una parte de la figura global de la plataforma correspondiente a su componente de color. Además de dichas figuras, también se observa en las distintas imágenes una gran cantidad de pequeñas regiones que se pueden considerar ruido. Por tanto, al igual que en el método anterior, se aplican varias operaciones morfológicas binarias, exactamente apertura-cierre y limpieza para eliminar esas pequeñas regiones y conseguir una imagen más tratable.

Estas operaciones morfológicas binarias se han de aplicar por igual a las tres imágenes binarias de las diferentes regiones por color. Primero se ha aplicado la apertura a la imagen binaria, figura 3.15 y el cierre a la imagen resultante de la apertura, figura 3.16. Para ambos casos se ha seleccionado el mismo elemento estructural, el cual tiene forma de disco con un radio de 10

píxeles desde el píxel central. Por último, a la imagen resultante del cierre se le ha aplicado la operación de limpieza, figura 3.17.

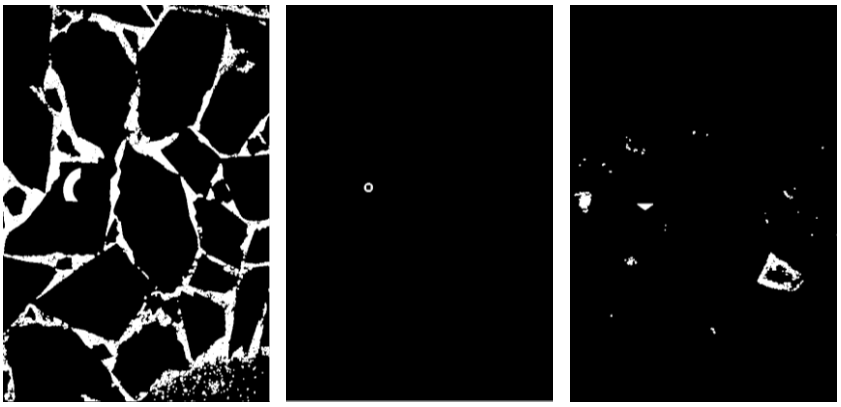


Figura [3.15]. Apertura binaria en los canales Rojo, Verde y Azul

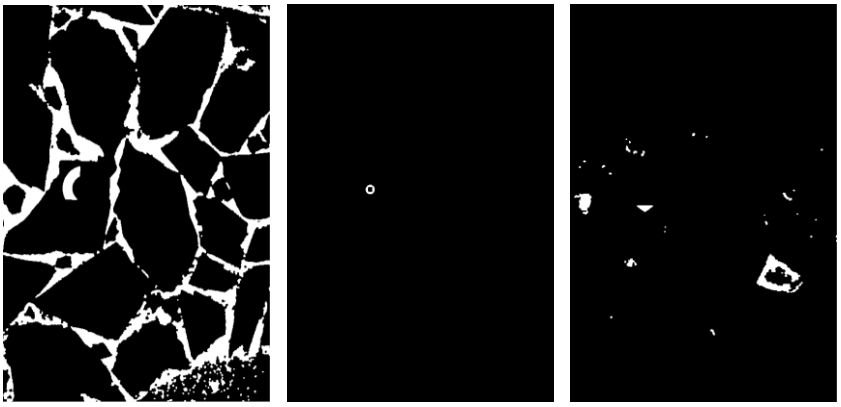


Figura [3.16]. Cierre binario en los canales Rojo, Verde y Azul

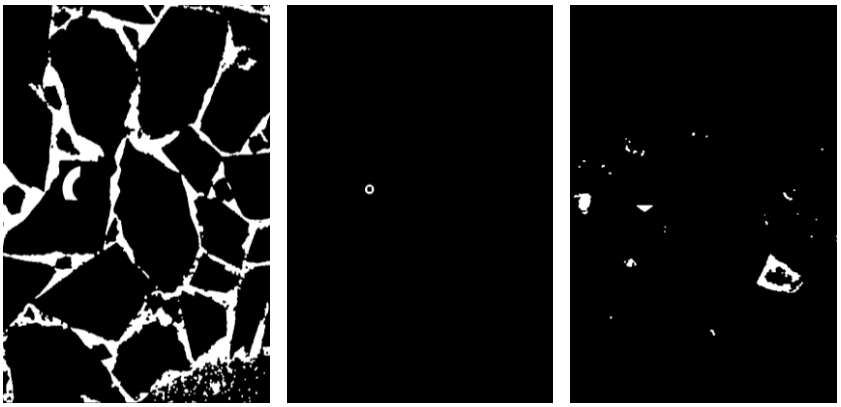


Figura [3.17]. Limpieza binaria en los canales Rojo, Verde y Azul

El siguiente paso consiste en el etiquetado e identificación de las regiones con posibilidad de contener las distintas figuras de la plataforma. Para realizar este proceso se aplica el método de etiquetado de componentes conexas a las tres imágenes binarias limpias obtenidas. Este método asigna a cada región un valor identificativo (etiqueta), figura 3.18.

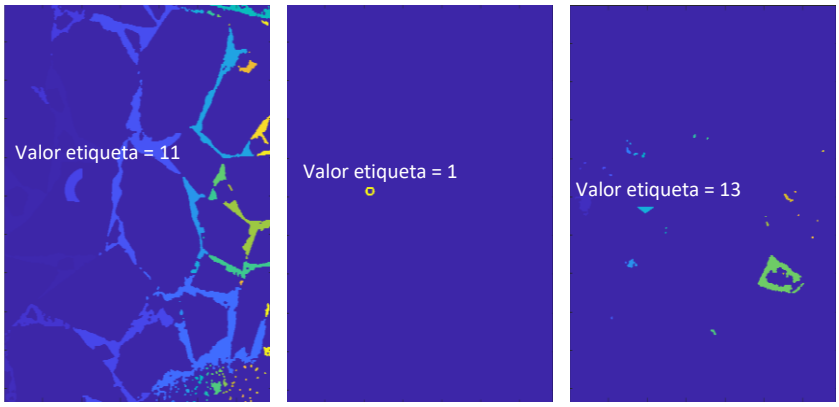


Figura [3.18]. Etiquetado de componentes conexas en los canales Rojo, Verde y Azul

Tras obtener todos los conjuntos de regiones por color y su etiqueta, pasamos a buscar el rectángulo delimitador de plataformas candidatas a partir de las regiones encontradas por cada color y teniendo en cuenta la geometría de la figura de la plataforma correspondiente a ese color. Para ello, se hace uso de algunas propiedades que nos proporcionan las diferentes regiones etiquetadas, como el área de la región y el rectángulo que la delimita.

Mediante el área se seleccionan las regiones de cada componente de color que superen un cierto tamaño establecido, suponiendo que por encima de ese tamaño es posible que sea la figura correspondiente a dicho color en la plataforma. El tamaño del área de cada figura se determinará automáticamente según la distancia a la que esté el vehículo autónomo de la plataforma y el tamaño proporcional de la figura en la misma. Para el desarrollo de la aplicación se han establecido unos valores de pruebas, donde siguiendo las proporciones de las figuras en la propia plataforma, el valor del tamaño del área de la figura roja es mayor que el de la figura azul, y el valor del tamaño del área de la figura azul, mayor que el de la figura verde.

A las regiones de las posibles figuras que han sido seleccionadas por el tamaño de área en los diferentes conjuntos de regiones, se les aplica, a cada una de ellas, la propiedad que nos proporciona su rectángulo delimitador. A partir de la geometría de dicho rectángulo delimitador y una geometría de referencia, se busca el rectángulo delimitador de una posible región que contiene la plataforma candidata. Este proceso se aplica de forma diferente a cada conjunto de regiones según su componente de color, ya que la figura correspondiente a cada componente tiene diferente geometría y diferente posición en la plataforma.

Este proceso es posible gracias a que la propiedad del rectángulo delimitador de la región siempre proporciona valores de sus datos respecto al mismo eje de coordenadas, es decir, el punto de inicio del rectángulo es siempre su esquina superior izquierda en coordenadas x e y, la distancia del ancho del rectángulo es siempre paralelo al eje horizontal (eje x) y la distancia del largo del rectángulo es siempre paralelo al eje vertical (eje y).

Por otra parte, siguiendo el formato Matlab con el que la propiedad del rectángulo delimitador de la región devuelve los valores, para formar el rectángulo delimitador de la plataforma se van a calcular sus valores con ese mismo formato: punto inicio (en x e y), ancho (eje x) y largo (eje y).

A continuación, se describe este proceso para cada conjunto de regiones etiquetadas.

1- Regiones etiquetadas para la componente de color roja.

En la figura 3.19 se muestra el cuadro delimitador de la plataforma y su figura roja con su rectángulo delimitador.

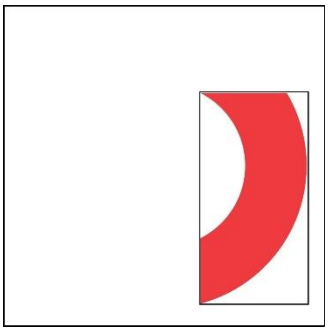


Figura [3.19]. Región roja de la plataforma

Teniendo en cuenta la imagen anterior, podemos saber aproximadamente la orientación de la plataforma respecto al sistema de coordenadas en el que se encuentra la imagen, a partir de las medidas del ancho y largo del rectángulo delimitador de la región de la posible figura. Si el ancho es mayor que el largo, se considera que la plataforma no está girada respecto el sistema de coordenada. En caso contrario, si el ancho es menor que el largo, se considera que la plataforma esta girada 90 grado respecto al sistema de referencia, figura 3.20.

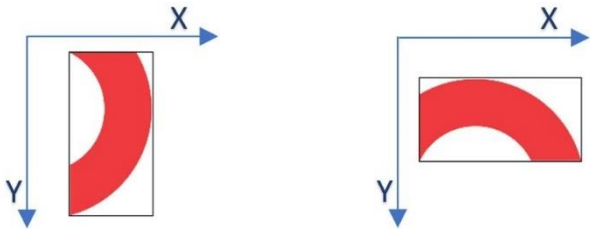


Figura [3.20]. Región roja con al ancho mayor y menor que el largo respectivamente

Para ambas orientaciones posibles de la plataforma según la figura roja, se pueden generar varias colocaciones del rectángulo delimitador de dicha plataforma respecto a la figura. Cuando el ancho es mayor que el largo (no está girada) se pueden dar dos colocaciones posibles, que el rectángulo delimitador de la figura se aproxime al lado izquierdo o que se aproxime al lado derecho del rectángulo delimitador de la plataforma, como muestra en la figura 3.21.

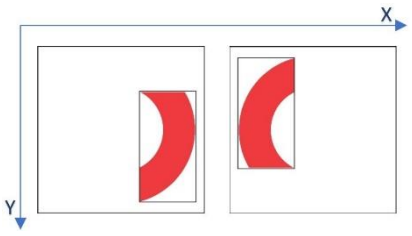


Figura [3.21]. Plataforma no girada

Para el caso en el que el ancho sea menor al largo (girada 90 grados) también se pueden dar dos colocaciones posibles: que el rectángulo delimitador de la figura se aproxime al lado superior o al lado inferior del rectángulo delimitador de la plataforma, como muestra en la figura 3.22.

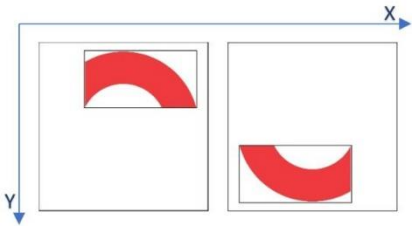


Figura [3.22]. Plataforma girada 90º.

Una vez obtenida la orientación de la posible figura con respecto al sistema de coordenada en el que se encuentra la imagen, se procede a obtener el rectángulo delimitador de la plataforma para las dos posibles colocaciones de éste con respecto a la figura. Para formar el cuadro delimitador de la plataforma es necesario obtener ciertas medidas similares a las que proporciona la propiedad del rectángulo delimitador de la región, tales como un punto izquierdo superior (en coordenadas X e Y), el ancho (siempre en eje X) y largo (siempre en eje Y). Estos valores se obtienen a partir de las medidas del rectángulo delimitador de la región y unas medidas de referencia obtenidas del plano de la plataforma.

Para ambas ubicaciones de la plataforma, se calcula el ancho (eje X) del posible rectángulo delimitador de la plataforma multiplicando la escala a la que se encuentra el ancho del rectángulo delimitador de la región en la imagen por el ancho de referencia del rectángulo delimitador de la plataforma. La escala se calcula mediante la división del ancho del rectángulo delimitador de la región obtenida en la imagen con el ancho de referencia del rectángulo delimitador de la región.

$$Ancho\ Plataforma = \left(\frac{Ancho\ Region\ Imagen}{Ancho\ Region\ Referencia}\right) * Ancho\ Plataforma\ Referencia$$

El cálculo del largo (eje Y) del posible rectángulo delimitador de la plataforma, se realiza de forma similar al ancho y es el mismo para ambas colocaciones de la plataforma.

$$Largo\ Plataforma = \left(\frac{Largo\ Region\ Imagen}{Largo\ Region\ Referencia}\right) * Largo\ Plataforma\ Referencia$$

Por otra parte, para calcular el punto izquierdo superior en las coordenadas X e Y del posible rectángulo delimitador de la plataforma, hay que tener en cuenta que éste será distinto según la colocación de la plataforma respecto a la figura. Para calcularlo se necesita obtener primero la distancia entre el punto izquierdo superior del rectángulo delimitador de la plataforma y el punto izquierdo superior del rectángulo delimitador de la región en ambos ejes de coordenadas. La distancia entre ambos puntos en el eje de coordenadas X, se calcula multiplicando la escala a la que se encuentra el ancho del rectángulo delimitador de la región en la imagen (ya que es la componente que pertenece al eje X) por la distancia de referencia entre ambos puntos. Esta distancia de referencia entre ambos puntos en el eje de coordenadas de X varía según la colocación de la plataforma.

$$Distancia\ "X"\ = \left(\frac{Ancho\ Región\ Imagen}{Ancho\ Región\ Referencia}\right) * Distancia\ "X"\ Referencia$$

La distancia entre ambos puntos en el eje de coordenadas Y, se calcula de la misma manera que en el eje X pero con la variación de que la escala se obtiene a partir del largo, ya que es su componente en el eje Y. Esta distancia de referencia entre ambos puntos en el eje de coordenadas de Y también varía según la colocación de la plataforma.

$$Distancia\ "Y"\ = \left(\frac{Largo\ Región\ Imagen}{Largo\ Región\ Referencia}\right) * Distancia\ "Y"\ Referencia$$

Una vez obtenidas las distancias en ambas coordenadas, se obtiene el punto superior izquierdo del posible cuadro delimitador de la plataforma. Para ello a las coordenadas X e Y del punto superior izquierdo del rectángulo delimitador de la región (obtenida en la imagen) se les resta la distancia calculada en el eje X y el eje Y respectivamente.

$$X\ Superior\ Izquierda\ Plataforma = X\ Superior\ Izquierda\ Región - Distancia\ X$$

$$Y\ Superior\ Izquierda\ Plataforma = Y\ Superior\ Izquierda\ Región - Distancia\ Y$$

Tras obtener los dos posibles rectángulos delimitadores de las plataformas candidatas para la región que se está evaluando del conjunto de regiones de la componente roja, se realiza un recorte de dichos rectángulos en la imagen original. En la figura 3.23 se muestra un ejemplo de aplicación del proceso a la región con etiqueta 11 del conjunto de regiones de color rojo.

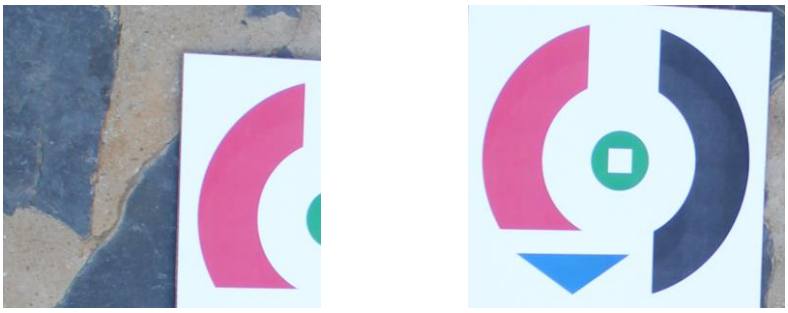


Figura [3.23]. Delimitación de la región Roja de la plataforma

2- Regiones etiquetadas para la componente de color verde.

En la figura 3.24 se muestra el cuadro delimitador de la plataforma y su figura verde con su rectángulo delimitador.

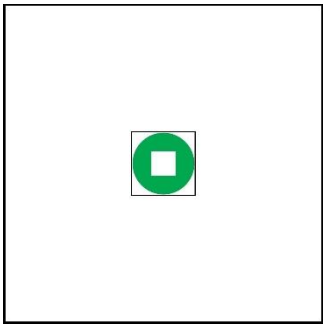


Figura [3.24]. Figura de color verde de la plataforma

Como se puede observar en la figura anterior, el rectángulo delimitador de la región tiene forma cuadrada debido a que la figura es un círculo. También cabe mencionar que el propio rectángulo delimitador de la plataforma tiene forma cuadrada debido a su diseño. Como el centro del cuadrado delimitador de la figura coincide con el centro del cuadrado delimitador de la plataforma, para este caso daría igual la orientación de la figura respecto al sistema de coordenadas, de forma que la colocación del cuadro delimitador de la plataforma respecto a la figura siempre va ser la misma, ya que la figura se encuentra en su centro y tiene forma circular.

El cálculo de las medidas necesarias para formar el posible rectángulo delimitador de la plataforma, las cuales son el punto izquierdo superior (en coordenadas X e Y), el ancho (siempre en eje X) y largo (siempre en eje Y), se realiza de la misma manera que en el caso anterior de las regiones rojas, pero teniendo en cuenta que solo hay una colocación posible.

Tras obtener el posible rectángulo delimitador de la plataforma candidata para la región que se está evaluando del conjunto de regiones de la componente verde, se realiza un recorte de dichos rectángulos en la imagen original. En la siguiente figura 3.25 se muestra un ejemplo de aplicación del proceso a la región con etiqueta 1 del conjunto de regiones de color verde.

Figura [3.25]. Delimitación de la región Verde de la plataforma

3- Regiones etiquetadas para la componente de color azul.

En la figura 3.26 se muestra el cuadro delimitador de la plataforma y su figura azul con su rectángulo delimitador.

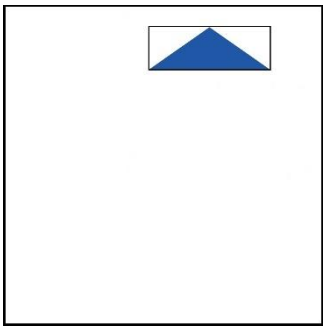


Figura [3.26]. Figura de color azul de la plataforma

Al igual que en las regiones de las componentes rojas, es posible determinar aproximadamente la orientación de la plataforma respecto al sistema de coordenadas en el que se encuentra la imagen, a partir de las medidas del ancho y largo del rectángulo delimitador de la región de la posible figura. Si el ancho es mayor que el largo, se considera que la plataforma no está girada respecto el sistema de coordenadas. En caso contrario, si el ancho es menor que el largo, se considera que la plataforma está girada 90 grados respecto al sistema de referencia, ver figura 3.27 para ambas configuraciones.

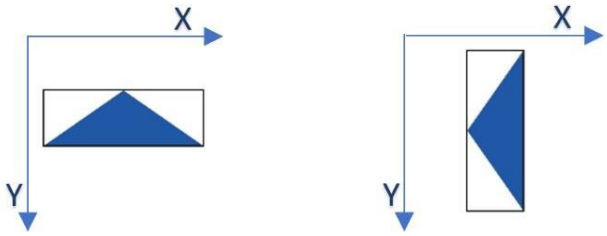


Figura [3.27]. Ancho mayor y menor que el largo, respectivamente

De la misma manera, para ambas orientaciones posibles de la plataforma según la figura azul, se pueden dar varias colocaciones del rectángulo delimitador de dicha plataforma respecto a la figura. Pero en este caso, a diferencia de las regiones rojas, cuando el ancho es mayor el largo (no está girada) se pueden considerar dos colocaciones posibles: que el rectángulo delimitador de la figura se aproxime al lado superior o al lado inferior del rectángulo delimitador de la plataforma, como muestra en la figura 3.28.

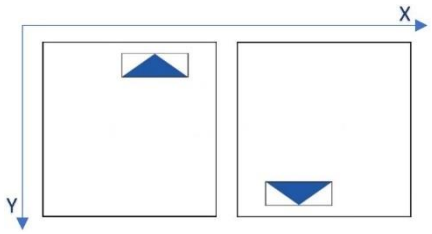


Figura [3.28]. Ubicaciones arriba-abajo relativas de la figura azul de la plataforma

Para el caso en que el ancho es menor al largo (girada 90 grados) también pueden darse dos colocaciones posibles: que el rectángulo delimitador de la figura se aproxime al lado izquierdo o se aproxime al lado derecho del rectángulo delimitador de la plataforma, como muestra en la figura 3.29.

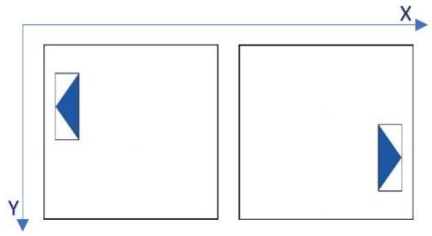


Figura [3.29]. Ubicaciones izquierda-derecha relativas de la figura azul de la plataforma

El siguiente paso consiste en obtener el rectángulo delimitador de la plataforma para las dos posibles colocaciones de éste con respecto a la figura. Las medidas necesarias para formar el rectángulo delimitador de la plataforma, las cuales son el punto izquierdo superior (en coordenadas X e Y), el ancho (siempre en eje X) y largo (siempre en eje Y), se calculan de la misma forma que en el caso de las regiones rojas.

Tras obtener los dos posibles rectángulos delimitadores de las plataformas candidatas para la región que se está evaluando del conjunto de regiones de las componentes azules, se realiza un recorte de dichos rectángulos en la imagen original. En la figura 3.30 se muestra un ejemplo de aplicación del proceso a la región con etiqueta 13 del conjunto de regiones de color azul.

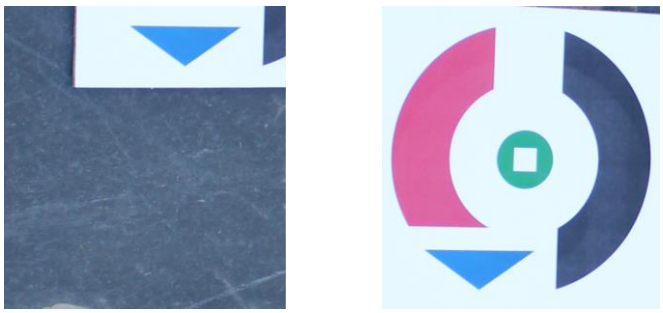


Figura [3.30]. Delimitación de la región Azul de la plataforma

En todos los recortes resultantes existe la posibilidad de que contenga la plataforma, por lo tanto, estos son transferidos a la red neuronal para ser clasificados.

3.2.2 Entrenamiento y validación de los modelos CNN: AlexNet y GoogleNet

Tanto para la red AlexNet como para la red GoogLeNet, se realiza su entrenamiento partiendo de una red pre-entrenada. La ventaja de utilizar redes pre-entrenadas es que se necesitan un menor número de datos de entrenamiento, con lo que además se reduce el tiempo de procesamiento. En cuanto a las redes pre-entrenadas, antes de volverlas a entrenar con las clases de interés para este trabajo, hay que realizar unos ajustes en sus capas. Estos ajustes consisten en reemplazar las capas finales de su estructura, las cuales contienen las clases de pertenencia de salidas del entrenamiento anterior. En la siguiente tabla [3.2] se muestran las capas finales que se reemplazan en la red AlexNet y GoogLeNet.

Red CNN	Nombre de capas	Tipo de capa
AlexNet	fc8	FullyConnectedLayer
	prob	SoftmaxLayer
	output	ClassificationOutputLayer
GoogLeNet	loss3-classifier	FullyConnectedLayer
	prob	Softmax
	output	ClassificationOutputLayer

Tabla [3.2]. Capas finales a reemplazar en AlexNet y GoogLeNet

Tras ajustar la estructura de capas de la red, se procede a su entrenamiento y validación utilizando la base de datos de imágenes propias adecuada a dicha red. La base de datos de imágenes propias tanto para la AlexNet como para la GoogleNet es la especificada en el apartado 3.1.1.

El proceso de entrenamiento y validación de ambas CNN genéricas se ha llevado a cabo con las mismas opciones de entrenamiento y el mismo repositorio de imágenes (cambiando la dimensión de las imágenes de entrada a la red). Las opciones de entrenamiento establecidas son: el gradiente descendente, una dimensión mínima de lote de 10 imágenes, 15 épocas, una tasa de aprendizaje inicial de 10^{-4}, y se ha establecido a infinito la opción de paciencia de validación con el fin de que el proceso no llegue a detenerse si no consigue mejorar la precisión tras el transcurso de un número determinado de épocas.

Dicho proceso se lleva a cabo mediante el software Matlab, el cual nos devuelve una interfaz gráfica con información relativa a la evaluación del proceso de entrenamiento. En dicha interfaz se muestran dos gráficas, una con los resultados de precisión en % (*accuracy*) frente a la evolución de las épocas, y la otra con el ajuste de la función de pérdida (*loss*) frente a la evolución de las épocas. Además de las gráficas, nos muestra resultados finales de validación, tiempo de entrenamiento, los ciclos de entrenamiento y otros datos tales como el hardware utilizado.

La figura [3.31] muestra la evolución del entrenamiento y validación de la red AlexNet donde se puede observar que el entrenamiento ha alcanzado una buena precisión, concretamente del 96,04%, en un tiempo de entrenamiento de 2min con 24seg y se han realizado las 20 épocas establecidas con 39 interacciones por cada época.

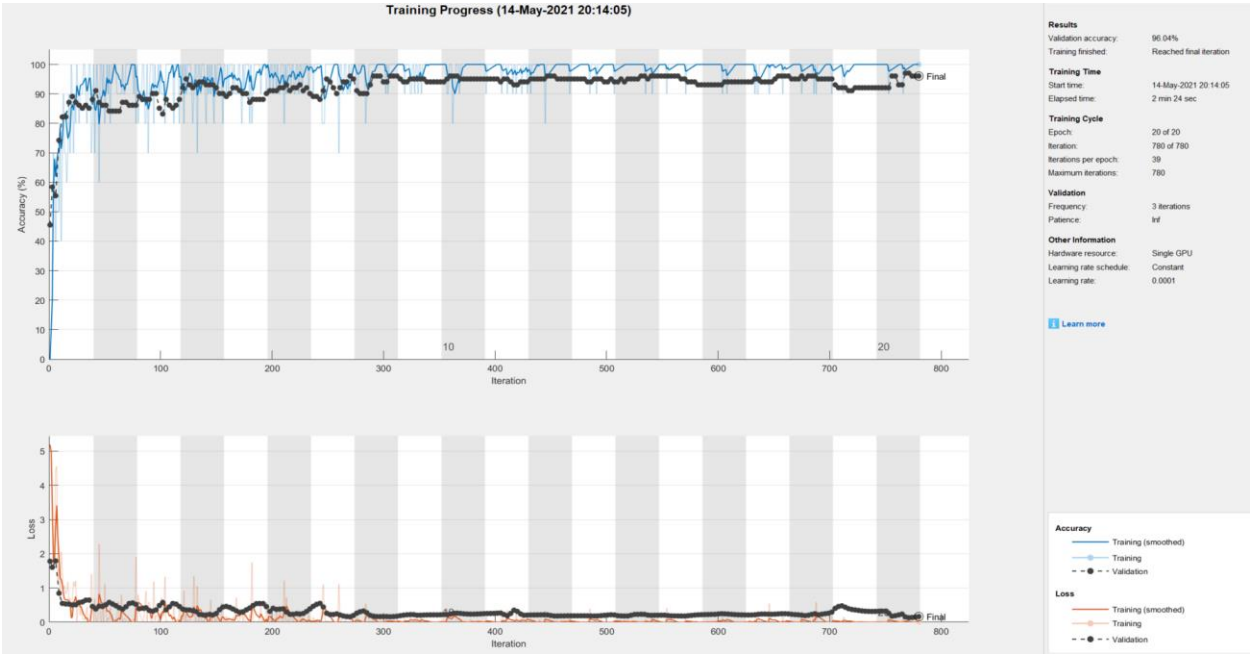


Figura [3.31]. Evolución del entrenamiento de AlexNet

En la figura 3.32 se muestra la evolución del entrenamiento y validación de la red GoogleNet donde se puede observar que el entrenamiento ha alcanzado también una buena precisión del 94,06%, en un tiempo de entrenamiento de 5min con 5seg y se han realizado las 20 épocas establecidas con 39 interacciones por cada época.

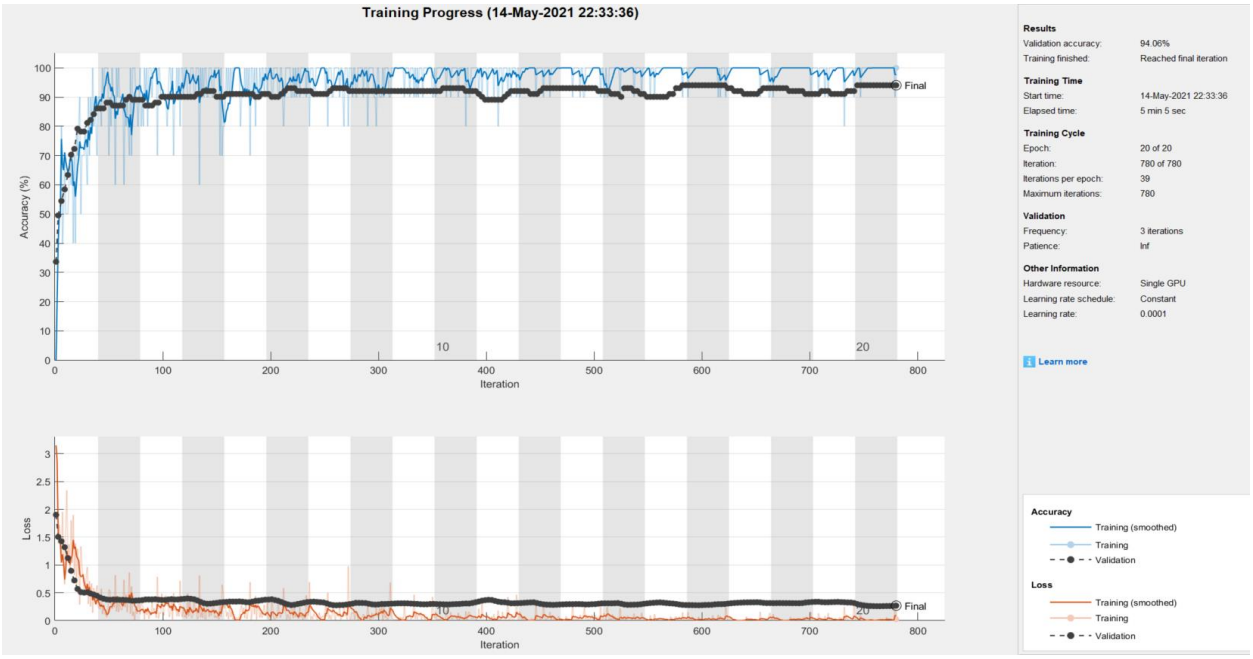


Figura [3.32]. Evolución del entrenamiento de GoogleNet

3.2.3 Clasificación de regiones mediante CNN: AlexNet y GoogleNet

Los recortes de las posibles plataformas sobre la imagen original obtenidos mediante los métodos de propuesta de regiones descritos anteriormente, se pasan a los correspondientes modelos CNN para su clasificación, obviamente previamente entrenadas con la base de datos

de imágenes propias. Estos recortes antes de ser clasificados son redimensionados a los tamaños requeridos por la red que se esté usando en cada momento.

La CNN una vez realizada la clasificación de los diferentes recortes de las posibles plataformas, devuelve, por cada recorte, la etiqueta de la clase con la mayor probabilidad de pertenencia, junto con la probabilidad obtenida para dicha clase.

Tras obtener la clasificación de todos los recortes, se seleccionan los que pertenezcan a la clase "plataforma" y con una probabilidad de pertenencia superior al 50%. Estas serían las plataformas candidatas y de todas ellas se elige la de mayor probabilidad. Si existen varias regiones con la máxima probabilidad, entonces se escoge entre ellas la que tenga mayor área.

Una vez tomada la decisión sobre cuál es la región que mejor identifica la plataforma, se dibuja en la imagen original su cuadrado delimitador y se le añade la etiqueta de la clase junto con la probabilidad de su pertenencia a ella. El cuadro delimitador es el mismo que se utilizó para realizar el recorte de la plataforma. En la figura [3.33] se muestra un ejemplo de la clasificación de una imagen mediante la CNN genérica AlexNet y el método 1 de propuesta de regiones.

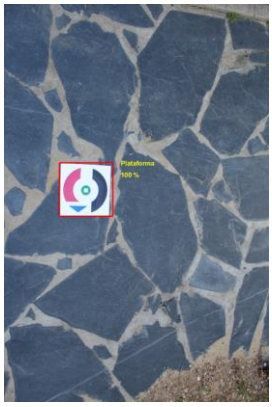


Figura [3.33]. Clasificación mediante AlexNet

3.3 Identificación mediante detectores de objetos: Faster R-CNN y YOLOV2

A diferencia de las CNN genéricas vistas en el apartado anterior, tal y como se mencionó previamente, estos modelos son detectores de objetos y además realizan una clasificación de los objetos detectados. Esto significa que no se necesita detectar regiones de interés de una imagen de entrada mediante técnicas clásicas de visión por computador, ya que esas regiones son propuestas por el propio modelo.

Se ha seleccionado para este trabajo el modelo YOLOV2 que, como se indicó previamente, pertenece a la categoría de detector de un solo estado, y también el modelo Faster R-CNN, que es un modelo que pertenece a la categoría de detector de dos estados.

3.3.1 Entrenamiento y validación de los modelos YOLOV2 y Faster R-CNN

Tanto en el modelo YOLOV2 como en el modelo Faster R-CNN se ha tomado como CNN base una red del tipo ResNet-50 pre-entrenado. A dicha red, antes de proceder a su entrenamiento se le reemplazan las tres últimas capas de clasificación por unas nuevas que soporten la nueva

clase principal “Plataforma” a detectar, más una adicional de fondo. En la siguiente tabla [3.3] se muestran las capas finales que se reemplazan en el modelo ResNet-50.

Red CNN	Nombre de capas	Tipo de capa
ResNet-50	fc1000	FullyConnectedLayer
	fc1000-softmax	SoftmaxLayer
	ClassificationLayer_fc1000	ClassificationOutputLayer

Tabla [3.3]. Capas finales a reemplazar en ResNet-50

Tras el ajuste de la estructura de la red ResNet-50, se pasa al entrenamiento y validación utilizando la base de datos de imágenes propias adecuada al modelo en cuestión (YOLOV2 o Faster R-CNN). La base de datos de imágenes propias para ambos modelos, se especifica en el apartado 3.1.1.

Las opciones de entrenamiento para YOLOV2 se llevan a cabo mediante el optimizador de Adam, con una dimensión mínima de lote de 16 imágenes, 20 épocas, con una tasa de aprendizaje inicial de 10^{-3}, estableciéndose que no se mezclen los datos de entrenamiento o validación.

Para el entrenamiento de estos modelos, el software Matlab no devuelve ninguna interfaz gráfica para mostrar los resultados, sino que se muestra por ventana de comandos. El tiempo de entrenamiento ha sido de 1hora con 14min y 39seg.

Las opciones de entrenamiento para la Faster R-CNN se llevan a cabo mediante el optimizador de Adam, con una dimensión mínima de lote de 2 imágenes, 30 épocas, con una tasa de aprendizaje inicial de 10^{-4}, y se ha establecido que se mezclen los datos de entrenamiento en cada época y de validación antes de cada validación de la red. Se establece esta mezcla de datos con el fin de que, si la dimensión mínima de lote no divide uniformemente el número de muestra de datos, evite descartar los datos que no encajen.

El proceso de entrenamiento de la Faster R-CNN, a diferencia de la YOLOV2 y de las CNN genéricas, realiza una secuencia automática dividida en 4 fases:

1) El entrenamiento de forma independiente de la red de propuesta de región (RPN).
2) El entrenamiento de forma independiente de la red de detectores Fast R-CNN.
3) Reentrenamiento de la RPN con los pesos obtenidos del entrenamiento Fast R-CNN.
4) Reentrenamiento de la Faster R-CNN a partir de la RPN reentrenada.

El resultado obtenido en este entrenamiento es de una precisión del 99,79% y una precisión RPN del 99,22%, en un tiempo de entrenamiento de 1hora con 30min y 31seg.

CAPÍTULO 4. RESULTADOS

En este capítulo se han llevado a cabo diferentes procedimientos de evaluación de los diferentes programas desarrollados en este trabajo para la detección de una plataforma de aproximación. Se ha analizado el comportamiento individual y posteriormente, comparado por una parte los dos modelos desarrollados con redes CNN genéricas AlexNet y GoogLeNet, y por otra parte los dos modelos de detectores de objetos basados en CNN, a saber: YOLOv2 y Faster R-CNN.

Para comprender los resultados obtenidos, hay que tener en cuenta el hardware y el software utilizado en el desarrollo de este trabajo, ya que las redes están optimizadas para obtener su mayor rendimiento respecto a ellos. A continuación, se describen los recursos hardware/software utilizados. Los programas Matlab se incluyen en el ANEXO I.

- Ordenador portátil MSI GE63VR 7RE Raider con procesador Core i7-7700HQ @ 2,80GHz con 16Gb de RAM, disco almacenamiento tipo SSD y una GPU Nvidia GeForce GTX 1060 de 6Gb.
- Cámara de fotos principal de 48 megapíxeles (sensor Sony IMX58) del móvil OnePlus Nord.
- Sistema Operativo Windows 10 64 bits.
- Software Matlab 2019b:
    - Deep Learning Toolbox.
    - Deep Learning Toolbox Model for AlexNet Network.
    - Deep Learning Toolbox Model for GoogLeNet Network.
    - Deep Learning Toolbox Model for ResNet-50 Network.
    - Computer Vision Toolbox.
    - Image Processing Toolbox.
    - Parallel Computing Toolbox.
- APP “Image Labeler” perteneciente a Matlab 2019b.
- Controladores Nvidia CUDA versión 10.1.0.

4.1 Comparativa de entrenamiento de redes generales: AlexNet y GoogLeNet

Los resultados que se presentan a continuación relativos a la validación para ambas redes generales son obtenidos a partir de las opciones de entrenamientos y características de base de datos de imágenes propias que se mencionan en el capítulo anterior.

Con el fin de evaluar el rendimiento de las redes CNN genéricas utilizadas, tras el entrenamiento de las mismas se ha generado la matriz de confusión. A partir de la información en dicha matriz se puede deducir visualmente el rendimiento de la red a través del porcentaje de aciertos y fallos obtenido de la clasificación de un conjunto de imágenes de referencias de cada clase disponible en dicha red.

En este trabajo se ha utilizado como imágenes de referencia el conjunto de validación proporcionado por la base de datos de imágenes propias. Este conjunto de imágenes dispone de 8 clases diferentes (Agua, Asfalto, Carril Bici, Césped, Hierbas-Flores, Plataforma, Rosas y Tierra). En las figuras 4.1 y 4.2, se muestran las matrices de confusión obtenidas a partir del conjunto de imágenes de validación para las redes CNN genéricas AlexNet y GoogLeNet.

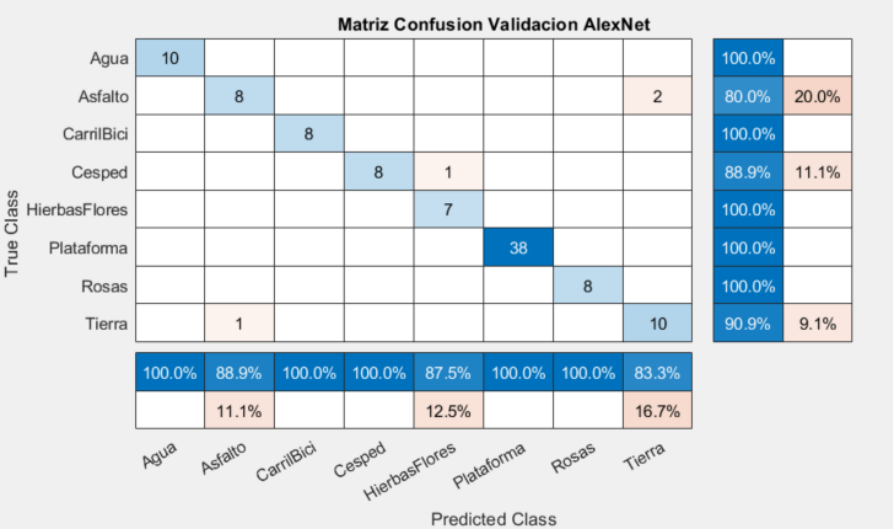


Figura [4.1]. Matriz de confusión de la red AlexNet

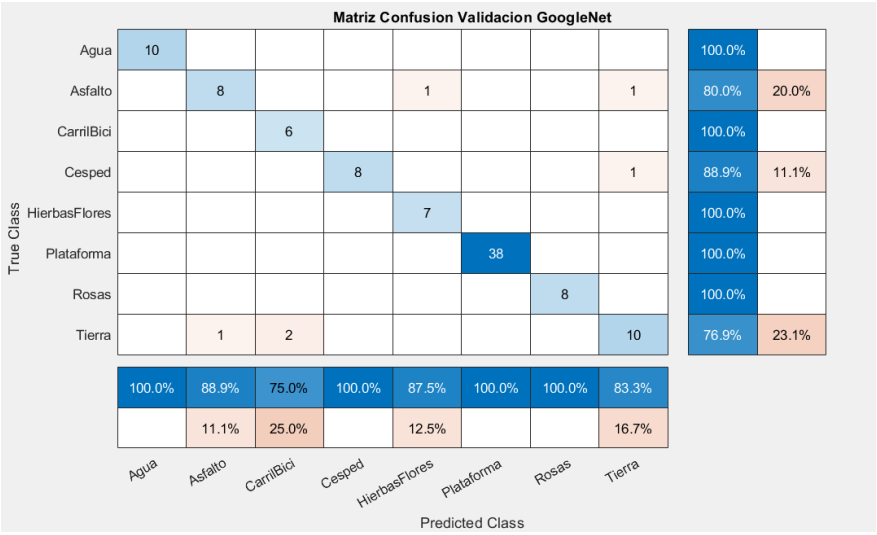


Figura [4.2]. Matriz de confusión de la red GoogLeNet

Como se puede observar en ambas matrices de confusión, el entrenamiento para las redes AlexNet y GoogLeNet es similar y con buenos resultados, sobre todo para la clase principal "Plataforma". Por lo tanto, se deduce que ambos entrenamientos son adecuados para la clasificación de regiones donde exista la posibilidad de contener la plataforma de aproximación.

A continuación, se muestra una tabla [4.1] con los resultados obtenidos en el entrenamiento y validación de las redes CNN generales AlexNet y GoogLeNet. En ella se muestran los tiempos de entrenamiento, datos de validación, error y el tamaño del archivo de la red. Los datos de error mostrados se obtienen a partir de clasificar en la red entrenada las imágenes, tanto de entrenamiento como de validación.

RED CNN	**Tiempo de entrenamiento (min:seg)**	**Precisión de validación (%)**	**Error de validación (%)**	**Error de entrenamiento (%)**	**Tamaño del archivo de la red (MB)**
AlexNet	2min 24seg	96,04	3,9604	0,25316	201
GoogLeNet	5min 5seg	94,06	5,9406	0.50633	21,2

Tabla [4.1]. Resultados del entrenamiento de la red AlexNet y la red GoogLeNet

A partir de los resultados mostrados en la tabla se puede observar que los datos obtenidos en el entrenamiento y validación de la red AlexNet son mejores que los obtenidos en la GoogLeNet, aunque la diferencia es poco significativa. También se observa que el tamaño de archivo de la red es mayor en la AlexNet, lo cual provoca a posteriori un tiempo de carga mayor en comparación con la GoogLeNet. En cuanto a los tiempos de entrenamiento, la red AlexNet realiza el entrenamiento y validación en un menor tiempo que la red GoogLeNet. Para finalizar, también se observa que el error obtenido en la validación y entrenamiento es menor, en ambos casos, para la AlexNet. Como resumen, se puede concluir que el entrenamiento de la red AlexNet es mejor que el de la red GoogLeNet, aunque tenga un tamaño mayor de archivo y por tanto, tiempo mayor de carga; lo cual no es significativo para este trabajo, ya que en la aplicación desarrollada solo carga la red una vez al inicio.

En las figuras [4.3] [4.4] se muestran unos ejemplos significativos de resultados obtenidos al utilizar las redes entrenadas para la clasificación de algunas de las imágenes pertenecientes al conjunto de validación, mostrándose en cada una de ellas los porcentajes de aciertos respecto de la clase identificada.

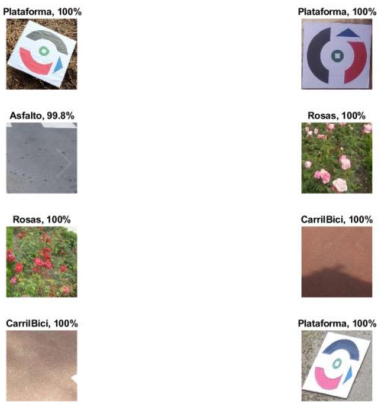


Figura [4.3]. Ejemplos de clasificación con la red AlexNet

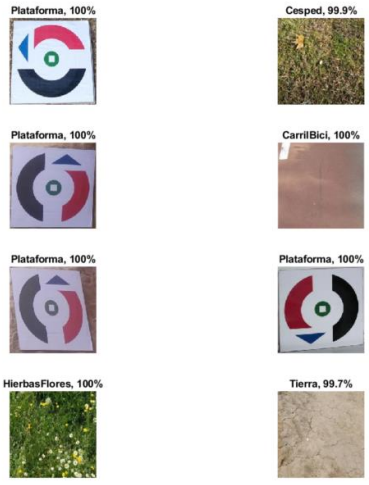


Figura [4.4]. Ejemplos de clasificación con la red GoogLeNet

4.2 Análisis y comparativa de detectores con los métodos de propuestas de regiones y clasificación de CNN generales

En este apartado se analizan y comparan los resultados obtenidos al someter un mismo conjunto de imágenes a los diferentes métodos de propuesta de regiones desarrollado y a las diferentes redes CNN generales utilizadas, AlexNet y GoogLeNet. El conjunto de imágenes utilizado en este procedimiento está formado por las 38 imágenes de la clase "plataformas" reservadas dentro de la base de datos de imágenes propias para la validación, tanto de las redes CNN generales como de los detectores de objetos.

A modo de recordatorio respecto de los dos métodos de propuesta de regiones descritos en el capítulo anterior, ambos tienen como objetivo la detección y recorte de regiones con posibilidad de contener la plataforma en una imagen de entrada, basándose el primer método en la extracción de regiones mediante la binarización, utilizando un umbral obtenido automáticamente y el segundo método en la extracción de regiones por color y geometría.

Para realizar el análisis y comparación de dichos resultados se aplica un procedimiento que se divide en cuatro fases:

- En la primera fase se realiza un análisis y comparación de forma cuantitativa de ambos métodos desarrollados.
- En la segunda fase se realiza un análisis y comparación de forma cuantitativa de ambas redes CNN generales utilizadas.
- En la tercera fase se realiza un análisis y comparación de forma cualitativa del etiquetado resultante según el método y la red CNN utilizadas.
- En la cuarta fase se realiza una análisis y comparación de forma cuantitativa de los tiempos de procesado generales.

A continuación, se exponen los resultados para las diferentes fases del procedimiento. De los resultados obtenidos para las 38 imágenes del conjunto de validación, se muestran los resultados de 10 imágenes seleccionadas, para que se puedan observar ciertos comportamientos igualitarios y diferenciadores entre los distintos elementos a analizar y comparar en cada fase. El resto de resultados de las imágenes restantes se exponen en el ANEXO II.

En la primera fase se analizan los resultados obtenidos mediante el procesamiento de las imágenes por ambos métodos. Cabe decir que estos resultados se mantienen invariables independientemente de las redes CNN genéricas que se vayan a utilizar después. Los datos de interés de cada método son, el tiempo de procesado por imágenes y el número de regiones extraídas con posibilidad de contener la plataforma. Estos datos son cuantitativos y nos sirven para analizar el comportamiento individual de los métodos y así, posteriormente, poderlos comparar. En la siguiente tabla [4.2] se exponen los resultados de las imágenes seleccionadas para ambos métodos (método 1 y método 2).

Referencia Imagen	Método 1 de propuesta de regiones		Método 2 de propuesta de regiones	
	Tiempo de Proceso (seg)	Nº de regiones	Tiempo de Proceso (seg)	Nº de regiones
B7r	1,8413	14	12,5989	61
B68r	4,4397	3	9,8036	12
B83r	1,6663	2	7,9698	25
B86r	1,1478	9	8,1115	61
B90r	1,5944	7	9,4062	127
B115r	2,2553	16	10,2201	69
B123r	3,4134	9	11,0413	89
B149r	0,9812	1	7,7322	4
B153r	2,0323	4	10,7804	49
B172r	1,1070	4	8,4574	24

Tabla [4.2]. Resultados para los métodos 1 y 2

A partir de los resultados mostrados en la tabla anterior se puede observar que el método 2 para una misma imagen tiene un tiempo de procesado significativamente mayor al del método 1, aunque también extrae muchas más regiones respecto al método 1, aumentando así la posibilidad de que alguna de ellas contenga la plataforma. Esta gran diferencia de tiempo de proceso y número de regiones se debe a que el método 2 extrae regiones por cada componente de color y después, por cada una de esas regiones propone diferentes alternativas según su geometría. Cabe decir que, aunque se tenga un número mayor o menor de regiones o se tarde más o menos en procesar, un método no es mejor que el otro, ya que cada uno se va a comportar de mejor o peor manera según la situación, como se irá viendo a lo largo de este apartado.

La segunda fase analiza el comportamiento de las diferentes redes CNN generales al clasificar las diferentes regiones que tienen la posibilidad de contener la plataforma propuestas por cada método. Los datos cuantitativos obtenidos que se analizan son el tiempo que tarda cada red en clasificar las diferentes regiones propuestas por cada método, el número de regiones de plataformas candidatas obtenidas tras la clasificación (las cuales son regiones que pertenecen a la clase "plataforma" y han obtenido un porcentaje de clasificación mayor al 50%) y, por último, el valor porcentual de clasificación máximo obtenido de las diferentes regiones. En la tabla [4.3] se muestran estos valores para las diferentes redes CNN AlexNet y GoogLeNet según los dos métodos previstos.

Métodos	Imágenes	Nº Regiones Extraídas	CNN AlexNet			CNN GoogLeNet		
			Tiempo de clasificación (seg)	Nº de Plataformas Candidatas	% Clasif.	Tiempo de clasificación (seg)	Nº de Plataformas Candidatas	% Clasif.
Método 1 de propuesta de regiones	B7r	14	0,1058	1	100	0,2243	1	99,9993
	B68r	3	0,0592	0	0	0,0823	1	62,3492
	B83r	2	0,0287	1	100	0,0451	1	100,000
	B86r	9	0,0682	2	100	0,1295	2	99,9999
	B90r	7	0,0636	2	100	0,1042	1	99,9999
	B115r	16	0,1430	3	100	0,2557	2	99,9998
	B123r	9	0,0973	1	99,7664	0,1575	0	0
	B149r	1	0,0086	1	100	0,0154	1	100
	B153r	4	0,0386	0	0	0,0667	0	0
	B172r	4	0,0311	1	100	0,0592	1	99,9995
Método 2 de propuesta de regiones	B7r	61	0,8759	7	100	1,3139	6	99,9997
	B68r	12	0,2238	4	100	0,3380	11	99,9996
	B83r	25	0,2806	8	100	0,4984	13	100,000
	B86r	61	0,4649	2	99,9999	0,9732	1	99,9687
	B90r	127	1,1681	6	93,1266	2,2689	1	61,0916
	B115r	69	1,0954	31	100	1,6545	13	99,9998
	B123r	89	1,1630	20	100	1,9744	10	99,9996
	B149r	4	0,0337	2	100	0,0636	2	99,8730
	B153r	49	0,6726	2	96,0098	1,1074	3	95,7117
	B172r	24	0,2544	8	100	0,4631	8	100

Tabla [4.3]. Resultados de clasificación de AlexNet y GoogLeNet

A partir de los resultados mostrados en la tabla [4.3] como en los resultados analizados y no expuestos en ella, se observan diferentes patrones de comportamiento en los resultados. El primero de ellos, en lo relativo al tiempo de clasificación de regiones para un mismo número de regiones a clasificar, se puede observar que la red CNN AlexNet tarda menos tiempo en clasificarlas que la GoogLeNet, siendo en muchas ocasiones la mitad del tiempo de proceso. Por tanto, en cuanto a este parámetro de tiempo, podemos concluir que la red CNN AlexNet es más rápida en clasificación que la GoogLeNet.

En cuanto al número de plataformas candidatas encontradas, se observa en la tabla [4.3] que en algunas imágenes una red encuentra un mayor número que la otra y viceversa en otras imágenes, por lo que en cuanto a datos, ambas redes se comportan de un modo similar. En cuanto a los resultados obtenidos a partir del conjunto de 38 imágenes, con el método 1 la red AlexNet encontró en 33 imágenes al menos una plataforma candidata, en cambio, para este mismo método la red GoogLeNet encontró alguna plataforma candidata en 35 imágenes, y para el método 2 tanto red AlexNet como la red GoogLeNet encontró al menos una plataforma candidata en las 38 imágenes del conjunto.

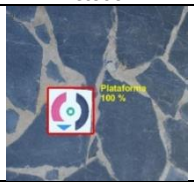
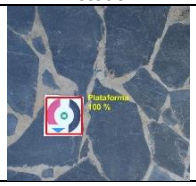
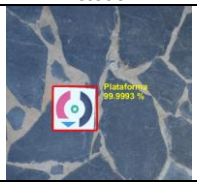
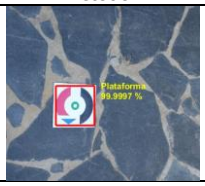
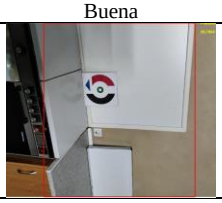
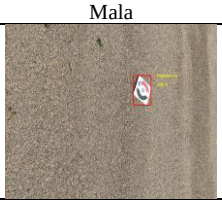
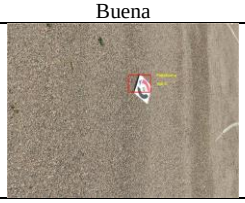
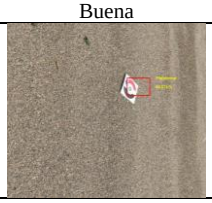
Otro patrón que se repite en la tabla es la precisión del valor dado por las diferentes redes en la clasificación. Se puede observar que en la red AlexNet se obtiene casi siempre un valor de la clasificación próximo al 100%, en cambio en la red GoogLeNet se obtienen los resultados con más precisión al contener decimales, lo cual es beneficioso a la hora de elegir una plataforma candidata para etiquetar, ya que, si hay varias plataformas candidatas con el 100% de posibilidades, la elección del etiquetado no va a depender de este porcentaje sino del área, lo cual puede producir errores al etiquetar.

Como se podrá observar en la siguiente fase, podemos encontrar algunas clasificaciones con un valor de porcentaje alto y un etiquetado resultante que no corresponde a la plataforma, lo que se consideran falsos positivos.

La tercera fase consiste en analizar y comparar el etiquetado resultante de la plataforma en las diferentes imágenes de entrada, en este caso siguiendo un criterio visual. El cual se basa en que el etiquetado ideal será cuando el rectángulo delimitador del etiquetado contenga en su totalidad la plataforma y se ajuste lo máximo posible a los bordes de ésta. Siguiendo ese concepto de etiquetado ideal, se valorarán los resultados con estos tres criterios de evaluación:

- BUENO: cuando el rectángulo delimitador se posicione en la plataforma y éste contenga en su totalidad la plataforma ajustándose más o menos a sus bordes.
- REGULAR: cuando el rectángulo delimitador se posicione en la plataforma y éste contenga en su totalidad la plataforma, pero no se ajuste nada a sus bordes.
- MALA: cuando el rectángulo delimitador no se posicione en la plataforma.

En la tabla [4.4] se muestra el etiquetado de la plataforma en la imagen junto al tiempo de etiquetado y la valoración para los diferentes métodos y diferentes redes CNN.

Datos	CNN AlexNet		CNN GoogLeNet	
	Método 1	**Método 2**	**Método 1**	**Método 2**
Imagen B7r				
Tiempo (s)	0,5973	0,5973	0,6001	0,5720
Valoración	Buena	Buena	Buena	Buena
Imagen B68r	-----			
Tiempo (s)	0	0,6414	0,6420	0,6473
Valoración	-----	Buena	Mala	Buena
Imagen B83r				
Tiempo (s)	0,6014	0,5970	0,6748	0,6194
Valoración	Buena	Regular	Buena	Mala
Imagen B86r				
Tiempo (s)	0,5956	0,5923	0,7134	0,6169
Valoración	Buena	Regular	Buena	Regular
Imagen B90r				
Tiempo (s)	0,6237	0,6010	0,7359	0,6112
Valoración	Buena	Mala	Buena	Mala
Imagen B115r				
Tiempo (s)	0,5609	0,5460	0,5519	0,5481
Valoración	Buena	Buena	Buena	Buena
Imagen B123r			-----	
Tiempo (s)	0,4124	0,4481	0	0,4163
Valoración	Mala	Buena	-----	Buena
Imagen B149r				
Tiempo (s)	0,4335	0,4367	0,4247	0,4369

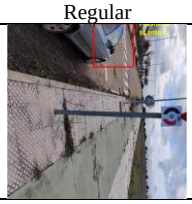
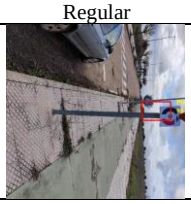
Valoración	Buena	Regular	Buena	Regular
Imagen B153r	-----		-----	
Tiempo (s)	0	0,5503	0	0,5658
Valoración	-----	Mala	-----	Mala
Imagen B172r				
Tiempo (s)	0,5766	0,5727	0,5878	0,6008
Valoración	Buena	Buena	Buena	Regular

Tabla [4.4]. Resultados de etiquetado de AlexNet y GoogLeNet

A partir de los resultados mostrados en la tabla [4.4] se observa en algunas imágenes que el etiquetado es igual de correcto para todos los casos, en cambio en otros difiere, siendo algunos buenos y otros malos. Esto es debido a una segmentación errónea de plataforma en la imagen de entrada por parte de los métodos o por falsos positivos en la clasificación de las diferentes regiones. Por otra parte, en la mencionada tabla se observa que los tiempos de etiquetado son similares para los cuatros casos, siempre que el etiquetado sea el mismo. Cuando el etiquetado difiere de un caso a otro, el tiempo de etiquetado también varía, aunque en poca cantidad.

Para finalizar el análisis de resultados de este procedimiento, en la cuarta fase se muestra la tabla [4.5] con los tiempos de proceso por cada imagen seleccionada, para poder tener una visón más general del rendimiento de cada red CNN con los diferentes métodos desarrollados.

Imágenes	**Método**	**CNN**	**Tiempo de carga de CNN (seg)**	**Tiempo de extracción regiones (seg)**	**Tiempo de clasificación (seg)**	**Tiempo de etiquetado (seg)**	**Tiempo total (seg)**
B7r	Método 1 de propuesta de regiones	AlexNet	2,5112	1,8413	0,1058	0,5973	2.5443
B68r				4,4397	0,0592	0	4.4990
B83r				1,6663	0,0287	0,6014	2.2963
B86r				1,1478	0,0682	0,5956	1.81161
B90r				1,5944	0,0636	0,6237	2.2818
B115r				2,2553	0,1430	0,5609	2.9592
B123r				3,4134	0,0973	0,4124	3.9231
B149r				0,9812	0,0086	0,4335	1.4233
B153r				2,0323	0,0386	0	2.07103
B172r				1,1070	0,0311	0,5766	1.7148
B7r	Método 2 de propuesta de regiones			12,5989	0,8759	0,5981	14.0730
B68r				9,8036	0,2238	0,6414	10.6688
B83r				7,9698	0,2806	0,5970	8.8475
B86r				8,1115	0,4649	0,5923	9.1688
B90r				9,4062	1,1681	0,6010	11.1753
B115r				10,2201	1,0954	0,5460	11.8615
B123r				11,0413	1,1630	0,4481	12.6525
B149r				7,7322	0,0337	0,4367	8.2027
B153r				10,7804	0,6726	0,5503	12.0033
B172r				8,4574	0,2544	0,5727	9.2844
B7r	Método 1 de propuesta de regiones	GoogLeNet	1.7143	1,8413	0,2243	0,6001	2,6657
B68r				4,4397	0,0823	0,6420	5,164
B83r				1,6663	0,0451	0,6748	2,3862
B86r				1,1478	0,1295	0,7134	1,9907
B90r				1,5944	0,1042	0,7359	2,4345
B115r				2,2553	0,2557	0,5519	3,0629
B123r				3,4134	0,1575	0	3,5709
B149r				0,9812	0,0154	0,4247	1,4213
B153r				2,0323	0,0667	0	2,099
B172r				1,1070	0,0592	0,5878	1,754

B7r	Método 2 de propuesta de regiones			12,5989	1,3139	0,5720	14,4848
B68r				9,8036	0,3380	0,6473	10,7889
B83r				7,9698	0,4984	0,6194	9,0876
B86r				8,1115	0,9732	0,6169	9,7016
B90r				9,4062	2,2689	0,6112	12,2863
B115r				10,2201	1,6545	0,5481	12,4227
B123r				11,0413	1,9744	0,4163	13,432
B149r				7,7322	0,0636	0,4369	8,2327
B153r				10,7804	1,1074	0,5658	12,4536
B172r				8,4574	0,4631	0,6008	9,5213

Tabla [4.5]. Tiempos de procesado de AlexNet y GoogLeNet.

A partir de los resultados mostrados en la tabla anterior se puede observar que el tiempo de carga de la red CNN es proporcional al tamaño de memoria del archivo de la red, como se mencionó en el apartado anterior, siendo el tiempo de carga mayor en la red AlexNet. Este tiempo solo se contabiliza una vez en los programas desarrollados, ya que la red solo se carga una vez al principio del proceso. Por ello, el tiempo total que se muestra es la suma del tiempo de extracción de regiones, el tiempo de clasificación y el tiempo de etiquetado. Todos estos tiempos pueden variar en magnitud dependiendo de la carga de proceso que tenga la CPU del PC en el momento que se está ejecutando el Script de Matlab.

4.3 Comparativa de entrenamiento de los modelos de redes YOLOv2 y Faster R-CNN

Los resultados que se presentan sobre los entrenamientos para ambos modelos de detectores de objetos basados en CNN son obtenidos a partir de las opciones de entrenamientos y características de bases de datos de imágenes propias que se describieron en el capítulo anterior. Como ya se mencionó anteriormente, la base de datos de imágenes propia para este modelo está formada solo por la clase principal “Plataforma” y está proporcionada en un 80% para el conjunto entrenamiento y el otro 20% para el conjunto de validación. En la tabla [4.6] se muestra un resumen de los entrenamientos de ambos detectores, donde se exponen las opciones de entrenamiento establecida, el tiempo de entrenamiento y el tamaño del archivo de modelo.

Modelo Detector	**Red CNN base**	**Opciones de entrenamiento**	**Tiempo de entrenamiento (h:min:seg)**	**Tamaño de archivo (MB)**	**NOTA**
YOLOv2	ResNet-50	Adam MiniBatchSize 16 InitialLearningRate 1e-3 MaxEpochs 20	01:14:39	97,6	
Faster R-CNN	ResNet-50	Adam MiniBatchSize 2 InitialLearningRate 1e-4 MaxEpochs 30 NegativeOverlapRange [0 0.3] PositiveOverlapRange [0.7 1]	01:30:31	119	Mini-batch Accuracy: 99,79% RPN Mini-batch Accuracy: 99,22%

Tabla [4.6]. Resumen del entrenamiento de YOLOv2 y Faster R-CNN.

Cabe destacar que el motivo de utilizar un MiniBatchSize tan pequeño para el entrenamiento de la Faster R-CNN es porque se saturaba la memoria de la GPU del PC utilizado y acaba interrumpiéndose el entrenamiento.

4.4 Análisis y comparativa de modelos de detectores de objetos basados en redes CNN

En este apartado se aplica un procedimiento a los diferentes modelos de detectores de objetos basados en CNN seleccionados en este trabajo (YOLOv2 y Faster R-CNN) para poder analizar resultados obtenidos en cada modelo y posteriormente compararlos entre ellos.

Este procedimiento se divide en tres fases, donde se obtienen resultados diferentes a comparar:

- La primera fase evalúa el rendimiento de ambos detectores, utilizando la métrica de Precisión/Recall.
- La segunda fase analiza y compara el comportamiento de forma cuantitativa, sometiendo ambos modelos a un mismo conjunto de imágenes.
- La tercera fase también analiza y compara de forma cualitativa mediante un criterio visual (cualitativo) los resultados de anotación de ambos modelos.

Para llevar a cabo las fases de este procedimiento, es necesario un mismo conjunto de imágenes, que es el mismo para todas ellas y está formado por las 38 imágenes de la clase "plataformas", reservadas en la base de datos de imágenes propias para la validación de los diferentes modelos de detectores.

En la primera fase se calcula la métrica *precisión-recall* de cada modelo de detector de objetos con la intención de poder comparar de forma cuantitativa los resultados obtenidos. Con la métrica *precisión-recall* (EvaluateDetectionPrecision, 2021; Breaking Down Mean Average Precision, 2021) se obtiene el rendimiento de los detectores, ya que evalúa la precisión de dicho detector. Esta evaluación se realiza a partir de un conjunto de imágenes de referencia etiquetadas; es decir, imágenes con un rectángulo delimitador en el objeto perteneciente a la clase que se quiere evaluar. El cálculo de la métrica *precisión-recall* parte del solapamiento del rectángulo delimitador en la imagen de referencia con el cuadro delimitador obtenido tras pasar esa misma imagen (sin rectángulo delimitador de referencia) por el detector. El resultado que devuelve utilizando la herramienta de Matlab, es el valor de la precisión media obtenida (entre 0 y 1) y la curva de precisión respecto a la recuperación (curva PR) para la clase que se esté evaluando, siendo la precisión la capacidad de un detector para llevar a cabo clasificaciones correctas y la recuperación, la capacidad de este mismo para encontrar los objetos de las clases establecidas.

En la figura [4.5] se muestra la precisión media y la curva PR obtenida al evaluar el detector YOLOv2 utilizando la métrica *precisión-recall* con el conjunto de imágenes de validación.

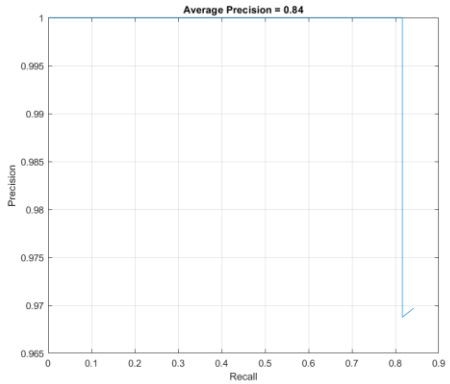


Figura [4.5]. Métrica Precisión-Recall de YOLOv2

En la figura [4.6] se muestra la precisión media y la curva PR obtenida al evaluar el detector Faster R-CNN utilizando la métrica *precisión-recall* con el conjunto de imágenes de validación.

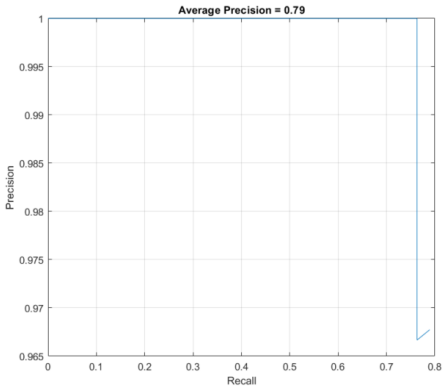


Figura [4.6]. Métrica Precisión-Recall de Faster R-CNN

La precisión media se calcula a partir del área bajo la curva PR, por tanto, cuanto mayor sea esta área mejor serán los resultados. En los resultados obtenidos para ambos detectores, se puede observar que hasta la caída de la curva PR, la precisión se mantiene en su máximo, mientras que aumenta la recuperación (recall), lo cual es un buen resultado ya que se está obteniendo el máximo de área posible hasta que se produce la caída de la curva PR. Aunque los resultados en ambos detectores son similares, el resultado obtenido en la YOLOv2 es algo mejor en esta fase del procedimiento.

Para las fases restantes del procedimiento, se exponen los resultados obtenidos sobre 10 imágenes del conjunto de validación. Estas imágenes se han seleccionado con la intención de que se puedan observar ciertos comportamientos igualitarios y diferenciadores entre ambos detectores. El resto de resultados de las imágenes restantes se exponen en el ANEXO II.

En la segunda fase del procedimiento se van a analizar y posteriormente comparar de forma cuantitativa los resultados de tiempos de procesos y porcentajes de clasificación obtenidos al aplicar los detectores YOLOv2 y Faster R-CNN sobre un mismo conjunto de imágenes de validación. Con el fin de tener una aproximación de las velocidades de ambos detectores, se analiza el tiempo de detección, que es el tiempo que tarda el modelo en detectar y clasificar los objetos en una imagen de entrada. También se analiza el tiempo de anotación, que es el tiempo que tarda el modelo en realizar los rectángulos delimitadores de los objetos en la imagen. En la tabla [4.7] se muestran los resultados de tiempos de proceso y el valor porcentual de la clasificación. Dentro de los tiempos de proceso se encuentra el tiempo total, el cual es el resultado de la suma del tiempo de detección con el tiempo de anotación.

Imágenes	**Modelo de detector**	**Tiempo de carga (seg)**	**Tiempo de detección (seg)**	**Tiempo de anotación (seg)**	**Tiempo total (seg)**	**% Clasificación**
B7r	YOLOv2	1,3481	0,0197	0,009	0,0287	63,88
B27r			0,0243	0,0132	0,0375	58,37 60,95
B42r			0,0204	0,0074	0,0279	53,85
B86r			0,0218	0,0100	0,0318	50,30
B90r			0,0284	0,0100	0,0384	0
B94r			0,0182	0,0071	0,0253	65,39
B109r			0,0586	0,0299	0,0884	51,60 61,90
B115r			0,1420	0,0470	0,1890	59,97
B153r			0,0253	0,0097	0,0350	52,38
B172r			0,0235	0,0412	0,0647	64,96 69,58

B7r	Faster R-CNN	11,4048	0,2499	0,0084	0,2583	98,49
B27r			0,2341	0,0072	0,2414	99,50
B42r			0,2614	0,0070	0,2684	99,79
B86r			0,2534	0,0065	0,2599	0
B90r			0,2323	0,0061	0,2384	51,82
B94r			0,2486	0,0067	0,2553	97,35
B109r			0,3314	0,0208	0,3522	99,47
B115r			0,4302	0,0985	0,5287	93,37 53,61
B153r			0,2494	0,0070	0,2564	95,40
B172r			0,2198	0,0077	0,2274	98,71

Tabla [4.7]. Tiempos de proceso y porcentajes de clasificación con la YOLOv2 y la Faster R-CNN

Cabe decir que ambos modelos no han detectado la plataforma en todas las imágenes del conjunto de validación. De las 38 imágenes del conjunto, tanto la YOLOv2 como Faster R-CNN han detectado la plataforma en 37 imágenes de dicho conjunto. Estas imágenes aparecen en la tabla [4.7] y son las que tienen valor 0% en la clasificación.

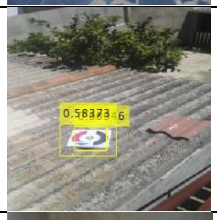
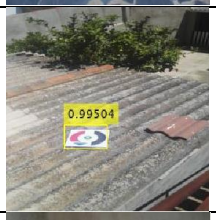
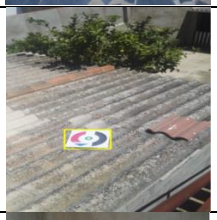
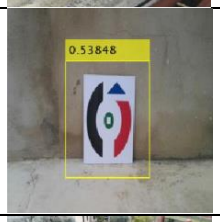
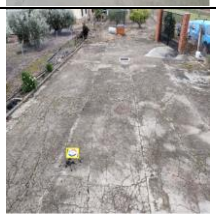
Analizando los resultados de la tabla [4.7] se puede observar que el tiempo de carga del detector Faster R-CNN es mucho mayor que el correspondiente a YOLOv2. A diferencia de las redes CNN generales, este tiempo no es proporcional al tamaño de memoria del archivo de la red, ya que ambos detectores tienen tamaños similares. Este tiempo solo se contabiliza una vez en los programas desarrollados, ya que los modelos solo se cargan una vez al principio del proceso.

Se puede ver en la tabla [4.7] que los tiempos de detección de la YOLOv2 son bastante menores a los de la Faster R-CNN, por lo tanto, podemos afirmar que el detector YOLOv2 es más rápido en detección del objeto y clasificación del mismo en todas las imágenes. Por otra parte, el tiempo de anotación es proporcional al tamaño del rectángulo delimitador de la plataforma en la imagen, por tanto, en las imágenes procesadas por ambos modelos que tienen como resultado un rectángulo delimitador similar para la plataforma, el tiempo de anotación también será similar. En la misma tabla se pueden observar tiempos similares; por tanto, las anotaciones serán similares. Partiendo de los tiempos indicados anteriormente, se ha calculado el tiempo total, donde se puede observar que el tiempo de anotación no afecta a la diferencia de velocidad entre ambos detectores. Cabe decir que todos estos tiempos pueden variar en magnitud dependiendo de la carga de proceso que tenga la CPU del PC en el momento en que se está ejecutando el Script de Matlab.

En cuanto al valor de la clasificación obtenida, sucede lo contrario a la velocidad del detector. En este caso, la Faster R-CNN obtiene mejores resultados que la YOLOv2, por lo que se podría decir que el detector Faster R-CNN es más preciso cuando se trata de clasificar objetos pequeños en una imagen.

En la tercera fase se analizan y comparan de forma cualitativa los resultados de anotación de los diferentes modelos de detectores. Para ello, se exponen los resultados obtenidos en los modelos a partir de las 10 imágenes seleccionadas junto a sus imágenes de referencia con anotación, y mediante un criterio de naturaleza cualitativa visual se comparan dichas imágenes, siendo mejor el resultado cuanto más similar sea la anotación resultante de los modelos a la anotación de la imagen de referencia. La imagen de referencia tiene la anotación deseada, la cual se realizó manualmente mediante la APP de Matlab, “Image Labeler” para el entrenamiento de los diferentes modelos. En la tabla [4.8] se exponen los resultados a analizar y comparar en esta fase obtenidos en 10 imágenes seleccionadas del conjunto de validación, sus imágenes de referencia correspondientes y la valoración visual obtenida. Para la valoración visual se han establecido tres criterios de evaluación:

- BUENA: cuando el rectángulo delimitador de la imagen resultante del modelo es casi el mismo o el mismo al rectángulo delimitador de la imagen de referencia.
- REGULAR: cuando el rectángulo delimitador de la imagen resultante del modelo se aproxime al rectángulo delimitador de la imagen de referencia.
- MALA: cuando el rectángulo delimitador de la imagen resultante del modelo sea muy diferente al rectángulo delimitador de la imagen de referencia.

IMAGEN	Modelo YOLOv2	Modelo Faster R-CNN	IMAGEN DE REFERENCIA	VALORACIÓN
B7r				YOLOv2: REGULAR Faster R-CNN: REGULAR
B27r				YOLOv2: REGULAR Faster R-CNN: REGULAR
B42r				YOLOv2: REGULAR Faster R-CNN: BUENA
B86r		-----		YOLOv2: BUENA Faster R-CNN: -----
B90r	-----			YOLOv2: ----- Faster R-CNN: MALA
B94r				YOLOv2: BUENA Faster R-CNN: BUENA
B109r				YOLOv2: REGULAR Faster R-CNN: BUENA

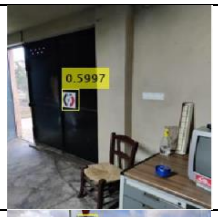
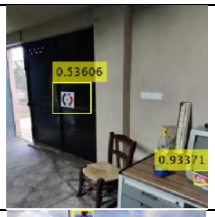
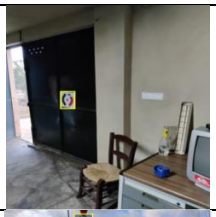
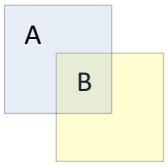
B115r				YOLOv2: BUENA Faster R-CNN: REGULAR
B153r				YOLOv2: BUENO Faster R-CNN: BUENO
B172r				YOLOv2: REGULAR Faster R-CNN: BUENA

Tabla [4.8]. Resultado de anotación de YOLOv2 y Faster R-CNN

Como se puede observar en la tabla [4.8], en la mayoría de las imágenes tanto para estos resultados expuestos como para los resultados restantes del conjunto de validación, las anotaciones resultantes de la modelo Faster R-CNN son mejores que las del modelo YOLOv2. También se puede observar un caso diferenciador en la imagen "B115r" resultante de la Faster R-CNN, en la cual aparece un falso positivo y una anotación regular de la plataforma, mientras su imagen homóloga en el modelo YOLOv2 obtiene un buen resultado. Por otra parte, con el método YOLOv2 se produce en algunas imágenes el solapamiento de varias anotaciones en el mismo objeto, como se observa en las imágenes B27r, B109, y B172r. Este efecto se puede solucionar mediante una técnica de filtrado de anotaciones para los detectores de objetos, basada en la relación de solapamientos de rectángulos delimitadores (SelectStrongestBboxMulticlass, 2021) y el procedimiento conocido como *Non-maximum Suppression* (NMS) (Sambasivarao, 2021) siguientes.



$$Unión = \frac{área\ (A \cap B)}{área\ (A \cap B)}$$

$$Mínimo = \frac{área\ (A \cap B)}{\text{mínimo}(área(A), área(B))}$$

Figura [4.7]. Relación de solapamiento de rectángulos delimitadores

La relación de unión es el resultado de la división del área de intersección entre rectángulo delimitador de A y B con el área de unión entre A y B. La relación del mínimo es la división del área de intersección entre rectángulo delimitador de A y B con el área mínima de los dos rectángulos delimitadores. Ambas relaciones toman un valor entre 0 y 1, donde 0 significa que entre ambos rectángulos delimitadores no existe solapamiento y 1 significa que ambos rectángulos delimitadores están solapados al 100%, es decir, que ambos rectángulos son iguales.

En este caso se ha utilizado para filtrar la relación del mínimo donde se ha establecido un valor umbral de relación del 0,8, el cual significa que para resultados de la relación del mínimo por encima de ese valor, se eliminan los rectángulos delimitadores solapados al rectángulo

delimitador que tenga mayor porcentaje de clasificación. En la tabla [4.9] se exponen las imágenes con el efecto de solapamiento obtenidas anteriormente junto a sus imágenes homologas filtradas.

IMAGEN	Resultado YOLOv2 con solapamiento	Resultado YOLOv2 filtrado
B27r		
B109r		
B172r		

Tabla [4.9]. Filtrado de anotaciones en el modelo YOLOv2

CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS

Con el presente trabajo se han desarrollado varios métodos válidos para la detección de una plataforma de aproximación para vehículos autónomos, basados en visión artificial haciendo uso de técnicas de visión por computador y de redes neuronales convolucionales (CNN). Para ello, se plantearon una serie de objetivos, los cuales se han conseguido.

En cuanto a la metodología seguida, la cual queda totalmente descrita en la documentación presente, se comenzó realizando una revisión bibliográfica de los métodos propuestos hasta el momento para este tipo de planteamientos. Esto, junto con un estudio específico de todos los aspectos científicos-técnicos involucrados en el desarrollo, nos ha permitido diseñar y validar los diferentes métodos de detección de este trabajo.

Teniendo en cuenta la información adquirida, tanto por los métodos existentes como por el estudio específico realizado, se diseñó y elaboró una plataforma de aproximación con características que ayudaran a su detención mediante técnicas de visión artificial. Su principal característica tal como se explica en la documentación, es su geometría compleja, compuesta por diferentes figuras a color sobre un fondo blanco, consiguiendo así la identificación de la plataforma a partir de la detección de alguna parte de ésta y facilitando su diferenciación del entorno de origen natural donde normalmente se debe ubicar.

Una vez diseñada la plataforma de aproximación, se generó una base de datos de imágenes propias necesarias para el desarrollo y validación de los diferentes métodos. La base de datos de imágenes propias generada, se tuvo que dividir en imágenes para el entrenamiento e imágenes para la validación y, además, adecuarla a cada método desarrollado.

A partir de todo lo anterior, se realizó el desarrollo de los diferentes métodos mediante el software Matlab, obteniendo de ello varias aplicaciones informáticas válidas para la detección de la plataforma, y verificadas para futuramente poder implementar alguna de ellas en el sistema de visión de un vehículo autónomo. Se desarrollaron un total de seis métodos para la detección de la plataforma de aproximación, donde cuatro de ellos combinan técnicas clásicas de visión por computador con redes CNN genéricas y los otros dos hacen uso de modelos de detectores de objetos basados en redes CNN.

Mediante las técnicas clásicas de visión por computador se desarrollaron dos métodos de propuesta de regiones, con el fin de extraer zonas de interés en una imagen digital donde exista la posibilidad de contener la plataforma, basándose el primer método en técnicas de extracción de regiones mediante binarización utilizando un umbral y el segundo método, en técnicas para la extracción de regiones por color y geometría. Estos dos métodos de propuesta de regiones se combinan con dos redes CNN genéricas (AlexNet y GoogLeNet) ajustadas y entrenadas para clasificar la plataforma en las zonas de interés extraídas por alguno de estos métodos. La combinación de estos dos métodos de propuesta de regiones con las dos redes CNN genéricas, dan como resultado los cuatro métodos diferentes de detección de la plataforma.

Estos cuatro métodos de detección, se sometieron a un mismo proceso de validación donde se obtuvieron los resultados que se muestran, analizan y comparan detalladamente en la documentación. Como conclusión general para estos cuatro métodos, los resultados obtenidos en ellos han sido buenos según lo esperado, por tanto, ambos métodos de propuesta de regiones

y ambas redes CNN generales son válidas para la detección y clasificación de la plataforma de aproximación en imágenes digitales. Ahora bien, según el método seleccionado, el cual depende de la combinación de un método de propuesta de regiones con una red CNN genérica, se obtienen cualidades diferentes, por ejemplo, si se combina el método 1 y la red AlexNet se ganaría en rapidez de procesado, o si se combina el método 2 y la red GoogLeNet se obtendría una mayor precisión de clasificación. La combinación se elegirá según las exigencias de rendimiento que requiera el vehículo autónomo al cual se le quisiera dotar de la capacidad de detección.

Por otro lado, los otros dos métodos desarrollados para la detección de la plataforma, se basaron en el uso de un modelo de detector de objetos basado en redes CNN diferentes para cada uno, siendo estos modelos de detectores YOLOv2 y Faster R-CNN. Estos dos modelos de detectores de objetos fueron ajustados y entrenados para detectar dicha plataforma de aproximación. Posteriormente, los dos métodos de detección desarrollados, también fueron sometidos a un mismo proceso de validación donde se obtuvieron resultados, los cuales también se muestran, analizan y comparan detalladamente en la documentación. Como conclusión general para estos dos métodos, los resultados obtenidos en ellos han sido buenos según lo esperado, por lo tanto, ambos modelos de detectores son válidos para la detección y clasificación de la plataforma de aproximación en imágenes digitales. Para estos dos métodos también se pueden observar cualidades que los diferencian entre sí, por ejemplo, el método que hace uso del modelo YOLOv2 requiere menor tiempo de proceso, por lo cual es más rápido y mejor si se quiere usar para obtener datos en tiempo real. En cuanto al método que hace uso del modelo Faster R-CNN, obtiene mejores resultados en la clasificación de los objetos de una clase en una misma imagen. En cambio, en los resultados de las anotaciones, se ha observado que ambos métodos están equilibrados en esta cualidad, ya que métricas realizadas a cada método son similares y también se ha observado que las anotaciones son similares en la mayoría de las imágenes donde aleatoriamente acierta un método más que el otro. Como se comentó para los otros cuatro métodos, la elección de uno de estos dos métodos se basará en función de las exigencias de rendimiento que requiera el vehículo autónomo sobre el que se quisieran implementar.

Para el desarrollo de los métodos propuestos se eligieron ciertas técnicas, redes CNN genéricas o modelos basados en detectores teniendo en cuenta su futura implementación en el sistema de visión de un vehículo autónomo. En cualquier caso, siempre se ha de suponer que estos vehículos autónomos necesitan un método de detección de la plataforma que sea robusto, eficiente, preciso y con tiempo de procesado lo más bajo posible para poder aumentar la velocidad de aproximación a la plataforma. Entonces, entre los diferentes métodos desarrollados y basándonos en los resultados obtenidos sobre un mismo hardware y software, los métodos que hacen uso de los modelos de detectores de objetos cumplen mejor estas necesidades supuestas para un vehículo autónomo. Según los resultados obtenidos en el presente trabajo, se puede concluir que existe una preferencia clara por estos métodos por su eficiencia de detección y, sobre todo, por su reducido tiempo de procesado, ya que eliminan la parte de identificación mediante técnicas clásicas de visión por computador la cuales requieren tiempo y recurso de procesamiento altos. Por otra parte, entre estos dos métodos, la elección se decantaría por el que hace uso del modelo Faster R-CNN, ya que, aunque su tiempo de procesado es algo mayor al del modelo YOLOv2, obtiene mayor precisión en las clasificaciones.

Es importante destacar, en cuanto al desarrollo y validación de los métodos, la elección del software proporcionado por Matlab, que se ha adaptado muy bien a las necesidades surgidas y facilitado la resolución de los problemas presentados, debido al gran número de *toolboxes* disponibles y la gran cantidad de documentación y ejemplos existentes en su centro de ayuda.

5.1 Trabajos futuros

En este apartado, tras haberse alcanzado los objetivos planteados en este trabajo, se exponen diferentes propuestas de trabajos futuros para continuar a partir de los resultados obtenidos.

- Aunque el desarrollo de los distintos métodos de detección de la plataforma mediante software en Matlab ha sido bastante cómodo, debido a las facilidades que nos proporciona dicho software, estos mismos métodos se podrían desarrollar mediante software de código abierto (*open-source*) utilizando, por ejemplo, la librería "OpenCV" para la parte de visión por computador y las bibliotecas de "Keras", "TensorFlow" o "Pytorch" para la parte de redes neuronales convencionales, todo ello codificado en lenguaje de programación de alto nivel como el Python. La ventaja de utilizar este tipo de software, es la flexibilidad que se obtiene a la hora de implementar los métodos desarrollados en sistemas embebidos.
- Partiendo de los métodos de detección de la plataforma, se podría desarrollar una aplicación en lenguaje Python con librerías *open-source* para ser integrada en un *framework* de guiado automático de vehículos, como ROS (Robot Operating System). La aplicación desarrollada se convertiría en un nodo de ROS, el cual guiaría la navegación del vehículo autónomo hacia la plataforma una vez sea detectada.
- Siguiendo con las prestaciones del software Matlab, desarrollar un sistema de visión que posteriormente se pueda adaptar a un vehículo autónomo. Para ello, se podría implementar una aplicación informática desarrollada en Matlab en un sistema embebido Raspberry Pi, el cual disponga de una cámara y sensores que fueran necesarios. La aplicación informática partiría de alguno de los métodos de los detectores desarrollados, pero adaptándola para que haga uso de la cámara del sistema embebido y proporcione unos datos de salida válidos para el vehículo autónomo al que se le quiera implementar.
- Calibrar la cámara de un sistema de visón al cual se le quiera implementar uno de los métodos de detección, utilizando para ello algún método existente de calibrado. Con ello, sería posible poder determinar la distancia a la que se encuentra el vehículo autónomo de la plataforma según el tamaño de esta vista desde la cámara o determinar el posible tamaño de la plataforma según la distancia a la que se pueda encontrar el vehículo autónomo en relación con esta.
- Desarrollar un sistema de posición y orientación relativa de precisión entre el vehículo autónomo y la plataforma mediante visón artificial, combinando redes neuronales convolucionales con técnicas clásicas de visión por computador. Para ello, se tendría que partir de un sistema de visión calibrado respecto a un tamaño establecido de la plataforma. Después, mediante uno de los métodos de detección desarrollado, por ejemplo, el que usa la Faster R-CNN, utilizarlo para detectar la plataforma y según el tamaño de su rectángulo delimitador, determinar la distancia a la que se encuentra la plataforma del vehículo autónomo. A medida que el vehículo autónomo se vaya acercando a la plataforma, mediante técnicas clásicas de visón por computador, este puede posicionarse y orientarse respecto a la plataforma haciendo uso de puntos de

referencia, por ejemplo, en la plataforma diseñada, la figura verde indica el centro y la figura azul indica la orientación. Para esto último, se puede partir del método 2 de propuesta de regiones, ya que hace uso de la técnica de extracción por color y geometría.

BIBLIOGRAFÍA

1. Alexe, B., Deselaers, T., Ferrari, V. (2012). Measuring the objectness of image windows. IEEE Trans. Pattern Anal. Machine Intell., 34(11), 2189-2102.
2. Bay, H., Ess, A., Tuytelaars, T., van Gool, L. (2008). SURF: Speeded Up Robust Features. Computer Vision and Image Understanding, 110, 346-359.
3. Bishop, C.M. (2006). Pattern Recognition and Machine Learning, Springer, NY, USA.
4. Bochkovskiy, A., Wang, C.Y., Mark-Liao, H.Y (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934v1 [cs.CV].
5. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J. (2019). YOLACT Real-time Instance Segmentation. arXiv:1904.02689v2 [cs.CV].
6. Breaking Down Mean Average Precision (mAP). Towards data science. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (Accedido Mayo 2021).
7. BVLC GoogleNet Model (2021). Disponible on-line: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet (Accedido Marzo 2021).
8. Cherkassky, V., Mulier, F. (1998). Learning form Data: concepts, theory and methods, Wiley, NY. USA.
9. Endres, I., Hoiem, D. (2010). Category Independent Object Proposals. In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6315. Springer, Berlin, Heidelberg.
10. EvaluateDetectionPrecision. Mathworks. https://es.mathworks.com/help/vision/ref/evaluatedetectionprecision.html (Accedido Mayo 2021).
11. Farneback, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, 363 - 370. Halmstad, Sweden: SCIA.
12. Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proc. 2014 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'14), pp. 580-587.
13. Haralick, R.M., Shapiro, L.G. (1992). Computer and Robot Vision. Addison-Wesley, Boston.
14. He, K., Gkioxari, G., Dollár, P., Girshick, R. (2018). Mask R-CNN. arXiv:1703.06870v3.
15. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proc. of the IEEE conference on computer vision and pattern recognition, pp. 770-778.
16. Imagenet (2021). Disponible on-line: http://www.image-net.org (Accedido Marzo 2021).
17. ImageNet LSVRC (2012). Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Disponible on-line: http://www.image-net.org/challenges/LSVRC/2012/ (Accedido Marzo 2021).

18. García-Pulido, J.A., Pajares, G., Dormido, S., de la Cruz, J.M. (2017). Recognition of a landing platform for unmanned aerial vehicles by using computer vision-based techniques Expert Systems with Applications, 76, 152-165.
19. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.
20. Leijian Yu, Cai Luo, Xingrui Yu, Xiangyuan Jiang, Erfu Yang, Chunbo Luo and Peng Ren. (2018). Deep learning for vision-based micro aerial vehicle autonomous landing. International Journal of Micro Air Vehicles, Vol. 10(2), 171-185
21. Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints, cascade filtering approach. International Journal of Computer Vision, 60(2), 91-110.
22. Matlab (2021). Deep Learning Toolbox. Disponible on-line: https://es.mathworks.com/products/deep-learning.html (Accedido Abril 2021).
23. Método del valor umbral. Wikipedia. https://es.wikipedia.org/wiki/M%C3%A9todo_del_valor_umbral (Accedido Mayo 2021).
24. Modelo de color HSV. Wikipedia. https://es.wikipedia.org/wiki/Modelo_de_color_HSV#Transformaciones (Accedido Marzo 2021).
25. Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Trans. Sys. Man. Cyber. 9(1), 62–66.
26. Pajares, G., Cruz, J.M. (2007). Visión por Computador: Imágenes digitales y aplicaciones. RA-MA, Madrid.
27. Pajares, G., Herrera, P.J., Besada, E. (2021). Aprendizaje Profundo. RC-Libros, Madrid.
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788. Las Vegas, USA.
29. Redmon, J., Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'17), 6517–6525. Honolulu.
30. Redmon, J. (2016). Darknet: Open Source Neural Networks in C. Disponible on-line: https://pjreddie.com/darknet/ (Accedido Diciembre 2021).
31. Ren, S., He, K., Girshick, R., Sun, J. (2017) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.
32. Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 [cs.LG].
33. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). 115(3), pp. 211–252.
34. Sambasivarao, K. (2021). Non-maximum Suppression (NMS). Towards data science. https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (Accedido Mayo 2021).

35. SelectStrongestBboxMulticlass. Mathworks. https://es.mathworks.com/help/vision/ref/selectstrongestbboxmulticlass.html (Accedido Mayo 2021).
36. Sharp, C. S., Shakernia, O., Sastry, S. S. (2001). A vision system for landing an unmanned aerial vehicle. In Proc. IEEE international conference on robotics and automation, pp. 1720–1727.
37. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. Computing Research Repository. arXiv:1409.4842 [cs.CV].
38. Vehículo autónomo. Wikipedia. https://es.wikipedia.org/wiki/Veh%C3%ADculo_aut%C3%B3nomo (Accedido Abril 2021).
39. Wang, X., Ming, Y., Zhu, S., Lin, Y. (2015). Regionlets for Generic Object Detection. IEEE Trans. Pattern Anal. Machine Intell., 37(10), 2071–2084.
40. Wubben, J., Fabra, F., Calafate, C. T., Krzeszowski, T., Marquez-Barja, J.M., Cano, J.C., Manzoni, P. (2019). Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. Electronics 8, 12,1532.
41. Zhao, A., Zheng, P., Xu, S.T., Wu, X. (2019). Object Detection with Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212-3232.
42. Zitnick, C.L., Dollár, P. (2014). Edge boxes: Locating object proposals from edges. Computer Vision-ECCV. Springer International Publishing, pp 391-405.

ANEXO

ANEXO I. Scripts desarrollados en Matlab 2019b

En la tabla A.1 se muestra los scripts desarrollados en Matlab 2019b, los cuales contienes las aplicaciones de los diferentes métodos de detección de la plataforma y aplicaciones complementarias necesarias para el desarrollo de este trabajo.

SCRIPT Matlab 2019b	DESCRIPCIÓN
BaseDataGroundTruth.mat	Fichero generado por la APP "Image Labeler" que contiene las imágenes de la clase plataforma con las anotaciones realizadas manualmente para el entrenamiento y validación de los modelos YOLOv2 y Faster R-CNN.
RecortarRedimensionar_Plataforma_AlexNet_GoogLeNet.m	Script encargado de realizar el recorte de la plataforma en las imágenes originales de la clase plataforma a partir de la anotación guardada en el fichero "BaseDataGroundTruth.mat" y posteriormente redimensionar dichos recortes para la entrada de las redes AlexNet y GoogLeNet.
Redimensionar_EntornoUrbano_AlexNet_GoogLeNet.m	Scrip encargado de redimensionar todas las clases de la base de datos propia de imágenes para la entrada de las redes AlexNet y GoogLeNet.
Division_Base_Imagenes_Propia_AlexNet.m	Script encargado de dividir la base de datos de imágenes propias de la red AlexNet, asignando una proporción de datos para el entrenamiento y otros para la validación.
Division_Base_Imagenes_Propia_GoogLeNet.m	Script encargado de dividir la base de datos de imágenes propias de la red GoogLeNet, asignando una proporción de datos para el entrenamiento y otros para la validación.
EntrenamientoCNNAlexNet_Plataforma.m	Script encargado de entrenar y validar la red CNN genérica AlexNet.
Metodo_Dtc_Plataforma_CNNAlexNet_M1_Binarizacion.m	Script que contiene la aplicación del método de detección de la plataforma realizado mediante el método 1 de propuesta de regiones y la red CNN genérica AlexNet. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Metodo_Detc_Plataforma_CNNAlexNet_Met2_RGB.m	Script que contiene la aplicación del método de detección de la plataforma realizado mediante el método 2 de propuesta de regiones y la red CNN genérica AlexNet. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Validacion_Metodo_CNNAlexNet_M1_Binarizacion.m	Script desarrollado para validar el método de detección de la plataforma realizado con el método 1 de propuesta de regiones y la red CNN genérica AlexNet. Este script nos proporciona los resultados para este método mostrado en la documentación.
Validacion_Metodo_CNNAlexNet_M1_RGB.m	Script desarrollado para validar el método de detección de la plataforma realizado con el método 2 de propuesta de regiones y la red CNN genérica AlexNet. Este script nos proporciona los resultados para este método mostrado en la documentación.
EntrenamientoCNNGoogLeNet_Plataforma.m	Script encargado de entrenar y validar la red CNN genérica GoogLeNet.
Metodo_Dtc_Plataforma_CNNGoogLeNet_M1_Binarizacion.m	Script que contiene la aplicación del método de detección de la plataforma realizado mediante el método 1 de propuesta de regiones y la red CNN genérica GoogLeNet. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Metodo_Detc_Plataforma_CNNGoogLeNet_Met2_RGB.m	Script que contiene la aplicación del método de detección de la plataforma realizado mediante el método 2 de propuesta de regiones y la red CNN genérica GoogLeNet. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Validacion_Metodo_CNNGoogLeNet_M1_Binarizacion.m	Script desarrollado para validar el método de detección de la plataforma realizado con el método 1 de propuesta de

	regiones y la red CNN genérica GoogLeNet. Este script nos proporciona los resultados para este método mostrado en la documentación.
Validacion_Metodo_CNNGoogLeNet_M1_RGB.m	Script desarrollado para validar el método de detección de la plataforma realizado con el método 2 de propuesta de regiones y la red CNN genérica GoogLeNet. Este script nos proporciona los resultados para este método mostrado en la documentación.
Entrenamiento_YOLOv2_Plataforma_CNNBase_ResNet50.m	Script encargado de entrenar el modelo YOLOv2 con la red CNN base ResNet-50.
Metodo_Dtc_Plataforma_YOLOv2.m	Script que contiene la aplicación del método de detección basado en el modelo YOLOv2. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Validacion_Metodo_Dtc_Plataforma_YOLOv2.m	Script desarrollado para validar el método de detección basado en el modelo YOLOv2. Este script nos proporciona los resultados para este método mostrado en la documentación.
Entrenamiento_FasterRCNN_Platafor_CNNBase_ResNet50.m	Script encargado de entrenar el modelo FasterRCNN con la red CNN base ResNet-50.
Metodo_Dtc_Plataforma_FasterRCNN.m	Script que contiene la aplicación del método de detección basado en el modelo FasterRCNN. Esta aplicación abre una carpeta donde se selecciona la imagen a evaluar.
Validacion_Metodo_Dtc_Plataforma_FasterRCNN.m	Script desarrollado para validar el método de detección basado en el modelo FasterRCNN. Este script nos proporciona los resultados para este método mostrado en la documentación.

Tabla [A.1]. Scripts desarrollados en Matlab 2019b

ANEXO II. Resultados cuantitativos de los procesos de validación

A continuación, se mostrarán varias tablas con los resultados obtenidos en los procesos de validación de los diferentes métodos de detección de la plataforma desarrollada.

En la tabla A.2 se muestran los resultados obtenidos en la validación del método de detención de la plataforma que combina el método 1 de propuesta de regiones con la red CNN genérica AlexNet.

Referencia Imagen	Tiempo de proceso M1 (s)	Nº de regiones	Tiempo de clasificación (s)	Nº Plataformas Candidatas	% Clasificación	Tiempo Etiquetado (s)	Tiempo Total (s)
B102r.jpg	2,3230	6	0,8149	1	99,9665	0,7787	3,9166
B103r.jpg	2,6759	6	0,0729	1	100	0,5179	3,2667
B109r.jpg	3,2538	3	0,0509	1	100	0,6290	3,9337
B110r.jpg	2,0779	6	0,0577	2	100	0,4504	2,5859
B115r.jpg	2,2553	16	0,1430	3	100	0,5609	2,9592
B116r.jpg	1,8556	12	0,1064	1	100	0,5040	2,4660
B120r.jpg	1,6907	7	0,0616	1	100	0,4357	2,1880
B123r.jpg	3,4134	9	0,0973	1	99,7664	0,4124	3,9231
B130r.jpg	1,4440	3	0,0235	1	100	0,4863	1,9538
B132r.jpg	1,0639	1	0,0098	1	100	0,4258	1,4996
B145r.jpg	1,4230	10	0,0843	1	100	0,6204	2,1278
B147r.jpg	1,2229	8	0,0602	1	100	0,6086	1,8917
B149r.jpg	0,9812	1	0,0086	1	100	0,4335	1,4233
B153r.jpg	2,0323	4	0,0386	0	0	0,0001	2,0710
B155r.jpg	0,9622	1	0,0093	1	100	0,4311	1,4025

B156r.jpg	1,0129	3	0,0228	1	100	0,5750	1,6107
B170r.jpg	0,9487	1	0,0082	1	100	0,5732	1,5301
B172r.jpg	1,1070	4	0,0311	1	100	0,5767	1,7148
B177r.jpg	1,6743	5	0,0383	1	100	0,4609	2,1735
B20r.jpg	2,1524	3	0,0393	0	0	0,0000	2,1917
B27r.jpg	1,3902	16	0,1216	1	100	0,6243	2,1361
B31r.jpg	1,4660	10	0,0866	1	99,9999	0,6493	2,2020
B33r.jpg	1,6021	9	0,0805	1	100	0,7682	2,4508
B37r.jpg	0,8081	0	0,0000	0	0	0,0000	0,8081
B42r.jpg	1,0076	2	0,0193	1	100	0,7042	1,7310
B43r.jpg	0,9954	2	0,0168	1	100	0,6689	1,6812
B44r.jpg	1,5459	2	0,0287	1	100	0,5919	2,1665
B4r.jpg	1,8292	3	0,0257	1	100	0,5981	2,4530
B60r.jpg	1,6667	5	0,0505	0	0	0,0000	1,7172
B68r.jpg	4,4397	3	0,0592	0	0	0,0000	4,4989
B72r.jpg	1,4823	7	0,0732	2	100	0,6858	2,2413
B79r.jpg	1,1561	4	0,0356	1	100	0,4815	1,6732
B7r.jpg	1,8413	14	0,1058	1	100	0,5973	2,5443
B83r.jpg	1,6663	2	0,0287	1	100	0,6014	2,2963
B86r.jpg	1,1478	9	0,0682	2	100	0,5956	1,8116
B90r.jpg	1,5944	7	0,0636	2	100	0,6238	2,2818
B94r.jpg	1,3037	8	0,0624	1	100	0,6109	1,9770
B9r.jpg	1,7312	4	0,0328	1	100	0,7262	2,4901

Tabla [A.2]. Resultados método detección con método 1 y AlexNet

En la tabla A.3 se muestran los resultados obtenidos en la validación del método de detención de la plataforma que combina el método 2 de propuesta de regiones con la red CNN genérica AlexNet.

Referencia Imagen	**Tiempo de proceso M2 (s)**	**Nº de regiones**	**Tiempo de clasificación (s)**	**Nº Plataformas Candidatas**	**% Clasificación**	**Tiempo etiquetado (s)**	**Tiempo Total (s)**
B102r.jpg	9,1066	36	1,3712	16	99,9993	0,7733	11,2511
B103r.jpg	10,4414	128	1,3700	14	100,0000	0,5305	12,3420
B109r.jpg	8,8086	75	0,8862	13	100,0000	0,6398	10,3345
B110r.jpg	8,7193	49	0,8087	13	100,0000	0,4179	9,9458
B115r.jpg	1,0220	69	1,0954	31	100,0000	0,5460	2,6634
B116r.jpg	9,1075	45	0,6883	10	100,0000	0,4715	10,2673
B120r.jpg	9,4037	34	0,5532	15	100,0000	0,4169	10,3738
B123r.jpg	11,0413	89	1,1630	20	100,0000	0,4481	12,6525
B130r.jpg	10,3887	151	1,5780	15	100,0000	0,4285	12,3951
B132r.jpg	7,9473	47	0,3978	4	100,0000	0,4300	8,7751
B145r.jpg	8,1495	48	0,4185	2	100,0000	0,5517	9,1197
B147r.jpg	8,2425	43	0,3789	5	100,0000	0,5627	9,1841
B149r.jpg	7,7322	4	0,0337	2	100,0000	0,4367	8,2027
B153r.jpg	10,7804	49	0,6726	2	96,0098	0,5503	12,0033
B155r.jpg	7,6149	5	0,0445	3	100,0000	0,4389	8,0983
B156r.jpg	7,4015	8	0,0688	6	100,0000	0,5744	8,0447

B170r.jpg	7,7904	16	0,1237	3	100,0000	0,5797	8,4938
B172r.jpg	8,4574	24	0,2544	8	100,0000	0,5727	9,2844
B177r.jpg	9,3492	78	0,7045	4	100,0000	0,4740	10,5277
B20r.jpg	8,8480	96	1,0737	7	100,0000	0,6546	10,5763
B27r.jpg	8,3367	73	0,7557	13	100,0000	0,6562	9,7486
B31r.jpg	7,9546	45	0,6272	15	100,0000	0,6277	9,2095
B33r.jpg	7,8370	38	0,5371	8	100,0000	0,6205	8,9946
B37r.jpg	6,9283	29	0,2842	1	93,5205	0,6455	7,8580
B42r.jpg	6,9057	21	0,2991	6	100,0000	0,5658	7,7705
B43r.jpg	6,8669	20	0,2031	3	100,0000	0,6283	7,6983
B44r.jpg	7,0423	27	0,3141	8	100,0000	0,5684	7,9248
B4r.jpg	14,2576	188	1,6920	10	99,9825	0,5646	16,5142
B60r.jpg	8,3837	54	0,6079	5	94,5368	0,6348	9,6264
B68r.jpg	9,8036	12	0,2238	4	100,0000	0,6414	10,6688
B72r.jpg	7,8618	52	0,5249	16	100,0000	0,6457	9,0325
B79r.jpg	9,1640	43	0,5912	13	100,0000	0,4756	10,2308
B7r.jpg	12,5989	61	0,8759	7	100,0000	0,5981	14,0730
B83r.jpg	7,9698	25	0,2806	8	100,0000	0,5970	8,8475
B86r.jpg	8,1115	61	0,4649	2	99,9999	0,5924	9,1688
B90r.jpg	9,4062	127	1,1681	6	93,1260	0,6010	11,1753
B94r.jpg	8,7412	86	0,9025	6	100,0000	0,6100	10,2537
B9r.jpg	13,2484	82	0,7604	6	100,0000	0,5961	14,6048

Tabla [A.3]. Resultados método detección con método 2 y AlexNet

En la tabla A.4 se muestran los resultados obtenidos en la validación del método de detención de la plataforma que combina el método 1 de propuesta de regiones con la red CNN genérica GoogLeNet.

Referencia Imagen	**Tiempo de proceso M1 (s)**	**Nº de regiones**	**Tiempo de clasificación (s)**	**Nº Plataformas Candidatas**	**% Clasificación**	**Tiempo Etiquetado (s)**	**Tiempo Total (s)**
B102r.jpg	2,3230	6	1,0379	1	82,6158	0,7770	4,1379
B103r.jpg	2,6759	6	0,1304	1	99,9999	0,5199	3,3262
B109r.jpg	3,2538	3	0,0749	1	100,0000	0,6311	3,9598
B110r.jpg	2,0779	6	0,1226	1	100,0000	0,4697	2,6701
B115r.jpg	2,2553	16	0,2557	2	99,9997	0,5519	3,0629
B116r.jpg	1,8556	12	0,1877	1	100,0000	0,4167	2,4600
B120r.jpg	1,6907	7	0,1086	2	99,9999	0,4161	2,2153
B123r.jpg	3,4134	9	0,1575	0	0,0000	0,0001	3,5711
B130r.jpg	1,4440	3	0,0449	1	100,0000	0,4237	1,9125
B132r.jpg	1,0639	1	0,0165	1	99,9999	0,4214	1,5018
B145r.jpg	1,4230	10	0,1536	1	99,9408	0,5952	2,1718
B147r.jpg	1,2229	8	0,1184	1	99,9999	0,5757	1,9170
B149r.jpg	0,9812	1	0,0154	1	99,9999	0,4247	1,4213
B153r.jpg	2,0323	4	0,0667	0	0,0000	0,0000	2,0990
B155r.jpg	0,9622	1	0,0161	1	100,0000	0,4392	1,4174
B156r.jpg	1,0129	3	0,0427	1	99,9999	0,6231	1,6787
B170r.jpg	0,9487	1	0,0161	1	100,0000	0,5684	1,5331

B172r.jpg	1,1070	4	0,0592	1	99,9995	0,5877	1,7539
B177r.jpg	1,6743	5	0,0726	1	99,9993	0,4306	2,1776
B20r.jpg	2,1524	3	0,0609	0	0,0000	0,0000	2,2133
B27r.jpg	1,3902	16	0,2415	1	99,9991	0,6346	2,2664
B31r.jpg	1,4660	10	0,1533	1	99,9921	0,6337	2,2530
B33r.jpg	1,6021	9	0,1405	1	99,9997	0,6291	2,3717
B37r.jpg	0,8081	0	0,0000	0	0,0000	0,0000	0,8081
B42r.jpg	1,0076	2	0,0324	1	99,9991	0,5832	1,6233
B43r.jpg	0,9954	2	0,0307	1	99,9998	0,7487	1,7748
B44r.jpg	1,5459	2	0,0406	1	100,0000	0,6998	2,2862
B4r.jpg	1,8292	3	0,0466	1	100,0000	0,6734	2,5492
B60r.jpg	1,6667	5	0,0869	1	66,0631	0,7784	2,5320
B68r.jpg	4,4397	3	0,0823	1	62,3492	0,6420	5,1641
B72r.jpg	1,4823	7	0,1422	1	99,9999	0,7071	2,3315
B79r.jpg	1,1561	4	0,0659	1	99,9999	0,5081	1,7302
B7r.jpg	1,8413	14	0,2243	1	99,9993	0,6001	2,6657
B83r.jpg	1,6663	2	0,0451	1	99,9999	0,6748	2,3862
B86r.jpg	1,1478	9	0,1295	2	99,9999	0,7134	1,9907
B90r.jpg	1,5944	7	0,1042	1	99,9999	0,7359	2,4345
B94r.jpg	1,3037	8	0,1194	1	99,9999	0,5991	2,0222
B9r.jpg	1,7312	4	0,0615	1	99,9998	0,5974	2,3901

Tabla [A.4]. Resultados método detección con método 1 y GoogLeNet

En la tabla A.5 se muestran los resultados obtenidos en la validación del método de detención de la plataforma que combina el método 2 de propuesta de regiones con la red CNN genérica GoogLeNet.

Referencia Imagen	**Tiempo de proceso M2 (s)**	**Nº de regiones**	**Tiempo de clasificación (s)**	**Nº Plataformas Candidatas**	**% Clasificación**	**Tiempo etiquetado (s)**	**Tiempo Total (s)**
B102r.jpg	9,1066	36	1,8636	14	99,9462	0,7648	11,7349
B103r.jpg	10,4414	128	2,5283	16	99,9999	0,4935	13,4633
B109r.jpg	8,8086	75	1,5330	11	99,9999	0,6309	10,9725
B110r.jpg	8,7193	49	1,2696	13	100,0000	0,4331	10,4220
B115r.jpg	1,0220	69	1,6545	13	99,9998	0,5481	3,2246
B116r.jpg	9,1075	45	1,1028	4	99,9998	0,4135	10,6239
B120r.jpg	9,4037	34	0,8727	10	99,9999	0,4235	10,6999
B123r.jpg	11,0413	89	1,9744	10	99,9995	0,4163	13,4320
B130r.jpg	10,3887	151	2,8557	18	99,9999	0,4120	13,6564
B132r.jpg	7,9473	47	0,7904	4	99,9999	0,4288	9,1665
B145r.jpg	8,1495	48	0,8245	2	99,9871	0,5724	9,5464
B147r.jpg	8,2425	43	0,7384	5	99,9998	0,5807	9,5616
B149r.jpg	7,7322	4	0,0636	2	99,8730	0,4370	8,2328
B153r.jpg	10,7804	49	1,1074	3	95,7117	0,5658	12,4537
B155r.jpg	7,6149	5	0,0867	4	100,0000	0,4364	8,1380
B156r.jpg	7,4015	8	0,1350	6	99,9997	0,5701	8,1066
B170r.jpg	7,7904	16	0,2550	4	99,9998	0,5883	8,6337
B172r.jpg	8,4574	24	0,4631	8	99,9999	0,6008	9,5212

B177r.jpg	9,3492	78	1,3756	5	99,9995	0,4711	11,1959
B20r.jpg	8,8480	96	2,1753	5	99,9998	0,7027	11,7260
B27r.jpg	8,3367	73	1,3751	6	99,9999	0,6286	10,3404
B31r.jpg	7,9546	45	1,0342	16	99,9949	0,6437	9,6325
B33r.jpg	7,8370	38	0,8915	9	99,9997	0,6532	9,3817
B37r.jpg	6,9283	29	0,5316	2	98,0042	0,6514	8,1114
B42r.jpg	6,9057	21	0,4537	7	99,9996	0,6243	7,9837
B43r.jpg	6,8669	20	0,3850	4	99,9998	0,6525	7,9044
B44r.jpg	7,0423	27	0,5477	8	99,9999	0,6083	8,1984
B4r.jpg	14,2576	188	2,9871	7	99,9822	0,5841	17,8288
B60r.jpg	8,3837	54	1,0727	2	99,9656	0,6814	10,1377
B68r.jpg	9,8036	12	0,3380	11	99,9996	0,6473	10,7889
B72r.jpg	7,8618	52	0,9823	18	99,9999	0,6647	9,5088
B79r.jpg	9,1640	43	0,9954	14	99,9999	0,4863	10,6457
B7r.jpg	12,5989	61	1,3139	6	99,9997	0,5720	14,4848
B83r.jpg	7,9698	25	0,4984	13	99,9999	0,6194	9,0877
B86r.jpg	8,1115	61	0,9732	1	99,9687	0,6168	9,7015
B90r.jpg	9,4062	127	2,2689	1	61,0916	0,6111	12,2862
B94r.jpg	8,7412	86	1,6438	4	99,9989	0,6344	11,0194
B9r.jpg	13,2484	82	1,3244	7	99,9999	0,6008	15,1736

Tabla [A.5]. Resultados método detección con método 2 y GoogLeNet

En la tabla A.6 se muestran los resultados obtenidos en la validación del método de detención de la plataforma basado en la YOLOv2.

Referencia Imagen	**Tiempo de detección (s)**	**% Clasificación**	**Tiempo de anotación (s)**	**Tiempo Total (s)**
B102r.jpg	2,1225	75,1395	1,0084	3,1309
B103r.jpg	0,0749	73,6861	0,0255	0,1005
B109r.jpg	0,0586	61,9653	0,0299	0,0884
B110r.jpg	0,0531	72,5505	0,0443	0,0974
B115r.jpg	0,1420	59,9702	0,0470	0,1890
B116r.jpg	0,0276	78,1080	0,0093	0,0369
B120r.jpg	0,0249	64,5502	0,0117	0,0366
B123r.jpg	0,0375	64,7279	0,0079	0,0454
B130r.jpg	0,0216	70,2590	0,0087	0,0303
B132r.jpg	0,0201	74,8691	0,0076	0,0277
B145r.jpg	0,0238	58,6143	0,0122	0,0360
B147r.jpg	0,0304	72,4853	0,0094	0,0398
B149r.jpg	0,0202	74,5605	0,0111	0,0313
B153r.jpg	0,0253	52,3827	0,0097	0,0350
B155r.jpg	0,0200	73,9256	0,0080	0,0280
B156r.jpg	0,0205	71,9550	0,0091	0,0296
B170r.jpg	0,0216	71,8403	0,0091	0,0307
B172r.jpg	0,0235	69,5789	0,0412	0,0647
B177r.jpg	0,0225	72,8131	0,0076	0,0300
B20r.jpg	0,0201	56,2535	0,0077	0,0278
B27r.jpg	0,0243	60,9460	0,0132	0,0375
B31r.jpg	0,0201	62,5031	0,0119	0,0320
B33r.jpg	0,0210	64,9128	0,0097	0,0307
B37r.jpg	0,0199	60,6400	0,0080	0,0279
B42r.jpg	0,0204	53,8485	0,0074	0,0279
B43r.jpg	0,0184	70,6168	0,0069	0,0252
B44r.jpg	0,0198	69,9881	0,0084	0,0282

B4r.jpg	0,0209	73,6462	0,0109	0,0317
B60r.jpg	0,0186	69,0098	0,0080	0,0265
B68r.jpg	0,0204	71,5489	0,0098	0,0302
B72r.jpg	0,0258	78,4030	0,0105	0,0363
B79r.jpg	0,0194	75,4257	0,0087	0,0281
B7r.jpg	0,0197	63,8769	0,0090	0,0287
B83r.jpg	0,0198	71,6396	0,0079	0,0277
B86r.jpg	0,0218	50,2967	0,0100	0,0318
B90r.jpg	0,0284	0,0000	0,0100	0,0384
B94r.jpg	0,0182	65,3928	0,0071	0,0253
B9r.jpg	0,0202	70,8407	0,0082	0,0284

Tabla [A.6]. Resultados método detección basado en YOLOv2

En la tabla A.7 se muestran los resultados obtenidos en la validación del método de detención de la plataforma basado en la Faster R-CNN.

Referencia Imagen	**Tiempo de detección (s)**	**% Clasificación**	**Tiempo de anotación (s)**	**Tiempo Total (s)**
B102r.jpg	2,3804	97,8871	1,1445	3,5249
B103r.jpg	0,3913	98,9860	0,0244	0,4157
B109r.jpg	0,3314	99,4694	0,0208	0,3522
B110r.jpg	0,2843	99,1394	0,0288	0,3131
B115r.jpg	0,4302	93,3708	0,0985	0,5287
B116r.jpg	0,2220	98,8507	0,0255	0,2475
B120r.jpg	0,2510	98,8238	0,0081	0,2591
B123r.jpg	0,2414	98,6192	0,0075	0,2488
B130r.jpg	0,2163	99,2828	0,0095	0,2258
B132r.jpg	0,2390	99,2138	0,0073	0,2463
B145r.jpg	0,2413	92,1023	0,0071	0,2484
B147r.jpg	0,2493	98,6628	0,0076	0,2569
B149r.jpg	0,2121	99,0082	0,0089	0,2209
B153r.jpg	0,2494	95,4006	0,0070	0,2564
B155r.jpg	0,2365	98,7593	0,0073	0,2438
B156r.jpg	0,2502	98,8398	0,0069	0,2571
B170r.jpg	0,2471	98,9076	0,0068	0,2539
B172r.jpg	0,2198	98,7067	0,0077	0,2274
B177r.jpg	0,2443	99,0988	0,0067	0,2510
B20r.jpg	0,2398	99,2543	0,0080	0,2478
B27r.jpg	0,2341	99,5041	0,0072	0,2414
B31r.jpg	0,2510	99,7250	0,0072	0,2583
B33r.jpg	0,2383	99,6546	0,0065	0,2448
B37r.jpg	0,2363	98,9358	0,0077	0,2440
B42r.jpg	0,2614	99,7856	0,0070	0,2684
B43r.jpg	0,2250	99,4520	0,0061	0,2311
B44r.jpg	0,2623	99,4358	0,0088	0,2711
B4r.jpg	0,2451	97,9745	0,0058	0,2509
B60r.jpg	0,2464	98,6323	0,0075	0,2539
B68r.jpg	0,2684	99,1814	0,0072	0,2756
B72r.jpg	0,2361	98,8923	0,0074	0,2435
B79r.jpg	0,2298	97,5160	0,0075	0,2373
B7r.jpg	0,2499	98,4944	0,0084	0,2583
B83r.jpg	0,2457	99,5390	0,0065	0,2523
B86r.jpg	0,2534	0,0000	0,0065	0,2599
B90r.jpg	0,2323	51,8240	0,0061	0,2384
B94r.jpg	0,2486	97,3491	0,0067	0,2553
B9r.jpg	0,2366	97,1718	0,0059	0,2424

Tabla [A.7]. Resultados método detección basado en Faster R-CNN

ANEXO III. Plataforma diseñada

Se adjunta la imagen de la plataforma diseñada para este trabajo.