



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**
Detección de insectos



Presentado por Arturo Carretero Mateo
en Universidad de Burgos — 17 de abril
de 2024

Tutor: Carlos Cambra Baseca



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Carlos Cambra Baseca, profesor del departamento de Digitalización, área de Ciencia de la Computación e Inteligencia Artificial.

Expone:

Que el alumno D.Arturo Carretero Mateo, con DNI 47313259M, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Detección de Insectos.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 17 de abril de 2024

Vº. Bº. del Tutor:

D. Carlos Cambra Baseca

Resumen

Este Trabajo de Fin de Grado se enfoca en el desarrollo y aplicación de tecnologías avanzadas para la detección de insectos, con el objetivo de mejorar las estrategias de control de plagas en el ámbito agrícola.

El estudio se centra en la identificación automática de tres especies clave de insectos plaga: *Macrolophus pygmaeus*, la mosca blanca (*Bemisia tabaci*) y *Nesidiocoris tenuis*.

Dada la importancia de mantener las poblaciones de plagas bajo control así protegiendo los cultivos y asegurando la producción alimentaria, este estudio propone una solución innovadora que combina técnicas de *visión por computadora e inteligencia artificial (IA)* para la identificación y cuantificación automática de insectos plaga.

La metodología empleada se basa en el diseño y entrenamiento de *modelos de aprendizaje profundo*, mas concretamente *redes neuronales convolucionales (CNN)*, mediante la *segmentación de imágenes* obteniendo así características que nos permitirán llegar a una conclusión final. Se recopiló un conjunto de datos de imágenes de alta resolución a través de un sistema de cámaras, las cuales mediante una colocación estratégica de trampas adhesivas por todo el cultivo permitían poder llegar a agrupar a estos insectos.

Los resultados obtenidos demuestran la eficacia del sistema propuesto, logrando una precisión y exactitud significativas en la detección de plagas en comparación con los métodos convencionales. La implementación de esta tecnología no solo permite una detección temprana y precisa de las plagas, sino que también facilita la aplicación de tratamientos fitosanitarios de manera más dirigida y racional, reduciendo el uso de pesticidas y minimizando su impacto ambiental.

En conclusión, este TFG contribuye al campo del control de plagas agrícolas ofreciendo una herramienta potente para la detección automática de insectos, promoviendo así prácticas de agricultura más sostenibles y eficientes. Futuras investigaciones podrían explorar la integración de esta tecnología con sistemas de monitoreo en tiempo real y la expansión de su capacidad para cubrir una gama más amplia de especies plaga.

Descriptores

Modelo de aprendizaje profundo, vision por computadora, inteligencia artificial (IA), redes neuronales convolucionales (CNN) , segmentacion de imagenes.

Abstract

This Bachelor's Thesis focuses on the development and application of advanced technologies for insect detection, aiming to improve pest control strategies in the agricultural field.

The study centers on the automatic identification of three key insect pest species: *Macrolophus pygmaeus*, the whitefly (*Bemisia tabaci*), and *Nesidiocoris tenuis*.

Given the importance of keeping pest populations under control to protect crops and ensure food production, this study proposes an innovative solution that combines *computer vision techniques* and *artificial intelligence (AI)* for the automatic identification and quantification of pest insects.

The methodology employed is based on the design and training of *deep learning models*, more specifically *convolutional neural networks (CNNs)*, through *image segmentation* to obtain characteristics that will allow us to reach a final conclusion. A dataset of high-resolution images was collected through a camera system, which, by strategically placing adhesive traps throughout the crop, allowed for the grouping of these insects.

The results obtained demonstrate the effectiveness of the proposed system, achieving significant precision and accuracy in pest detection compared to conventional methods. The implementation of this technology not only allows for early and precise detection of pests but also facilitates the application of phyto-sanitary treatments in a more targeted and rational manner, reducing the use of pesticides and minimizing their environmental impact.

In conclusion, this Bachelor's Thesis contributes to the field of agricultural pest control by offering a powerful tool for the automatic detection of insects, thereby promoting more sustainable and efficient agricultural practices. Future research could explore the integration of this technology with real-time monitoring systems and the expansion of its capacity to cover a wider range of pest species.

Keywords

Deep learning model, computer vision, artificial intelligence (AI), convolutional neural networks (CNN), image segmentation.

Índice general

Índice general	v
Índice de figuras	vii
Índice de tablas	ix
1. Introducción	1
2. Objetivos del proyecto	5
2.1. Objetivos marcados por los requisitos del software	5
2.2. Objetivos de carácter técnico	6
3. Conceptos teóricos	7
3.1. Redes neuronales	8
3.2. Redes Neuronales Convolucionales - CNNs	16
3.3. R-CNN – región basada en una red convolucional	20
3.4. Fast R-CNN – Fast Region based convolutional network . . .	22
3.5. Mask R-CNN	24
3.6. Cuatro fases fundamentales	28
3.7. Herramientas y tecnologías	30
4. Técnicas y herramientas	31
5. Aspectos relevantes del desarrollo del proyecto	35
5.1. Datos	36
5.2. COCO - Common Objects in Context	39
5.3. Objetos	40
5.4. Algoritmos	43

5.5. Resultados	43
5.6. Mejoras	43
6. Trabajos relacionados	45
7. Conclusiones y Líneas de trabajo futuras	47
Bibliografía	49

Índice de figuras

3.1. red neuronal <i>feedforward (FNN)</i> simple de tres capas, compuesta por una capa de entrada, una capa oculta y una capa de salida.	8
3.2. Comparativa del error segun el número de epochas y neuronas [9]	10
3.3. <i>Funcion de activación sigmoidea</i>	12
3.4. <i>Función de activación Tanh</i>	13
3.5. <i>Función de activación ReLU</i>	13
3.6. Gráfica comparativa precision y perdida	15
3.7. Secuencia de una CNN paso a paso [26]	16
3.8. Secuencia de entrada RGB en CNN [24]	17
3.9. Aplicación de convolución[11]	17
3.10. Demostración gráfica de Max Pooling y Average Pooling	18
3.11. Capa flatten en la estructura de red CNN[25]	19
3.12. Descripción general de la estructura de red CNN[15]	20
3.13. Representacion gráfica de el algoritmo Gredy en búsqueda selectiva[12]	21
3.14. A la izquierda, la Red de Propuestas de Región (RPN). A la derecha, se presentan ejemplos de detecciones utilizando propuestas RPN[22]	21
3.15. modelo Fast R-CNN[13]	22
3.16. Comparativas entre <i>Image recognition , Segmentatic segmentation , Object detection y Instance segmentation</i>	24
3.17. Primera máscara sobre un objeto a detectar	25
3.18. Bounding boxes y mascaras generadas por un modelo Mask R-CNN [20]	25
3.19. Perdida al hacer <i>Max Pooling</i>	26
3.20. Ejemplo de stride utilizando interpolación bilineal[16]	26
3.21. Arquitectura Mask R-CNN	27

4.1. proceso de detección de objetos utilizado por la arquitectura de red neuronal YOLO[21]	33
5.1. Ejemplo de la imagen 001.jpg y su etiquetado 001.xml	37
5.2. Ejemplo de una imagen de entrada a Mask R-CNN	41
5.3. Ejemplo de una imagen tras haber realizado predicciones	41
5.4. Figura correspondiente a los atributos de 2 insectos detectados .	42

Índice de tablas

3.1. Herramientas y tecnologías utilizadas en cada parte del proyecto 30

1. Introducción

En los últimos años han aparecido las primeras aplicaciones de la informática a la agricultura. Estas aplicaciones resuelven tareas repetitivas, mecanicistas o de manejo de grandes volúmenes de información. En el sector industrial, que tiene menos factores incontrolados y aleatorios que la agricultura, se vienen aplicando cada vez más los métodos CAD (Computer Aid Design), CAM (Computer Aid Manufacture) y la robótica.[7]

En la agricultura no pueden aplicarse con tanta rapidez estos métodos tan deterministas por trabajar con seres vivos y no ser posible el control de todas las variables climáticas, ecológicas ni económicas. Es necesario recurrir a la rama más moderna de la informática que es la Inteligencia Artificial. Distintos trabajos han analizado con cierto detalle las aplicaciones de estas técnicas a la agricultura. La Inteligencia Artificial (I.A.) tiene dos campos de aplicación en la agricultura: la robótica y la construcción de sistemas expertos.[7]

Un sistema experto se define en el documento como un tipo de sistema informático que emula la capacidad de toma de decisiones de un experto humano. Estos sistemas son capaces de resolver problemas complejos mediante el uso de bases de conocimientos especializados y una forma de razonamiento que imita la lógica humana. [3]

En nuestro caso utilizaremos un sistema experto el cual utiliza la segmentación de imágenes para la obtención de características e información y poder tomar decisiones a partir de esta información obtenida.

En nuestro caso utilizaremos estas técnicas para la detección y control de los insectos: *Trialeurodes vaporariorum* y *Bemisia tabaci* los cuales están listados entre los 10 plagas más problemáticas en cultivos de vegetales en invernadero.[19]

Podremos interactuar con un modelo ya preentrenador y entrenar uno nuevo mediante la aplicación web, mostrándose en esta los resultados producidos durante la detección de dichos insectos.

2. Objetivos del proyecto

Este apéndice detalla de manera clara y breve los propósitos buscados a través de la ejecución del proyecto. Se diferencian dos tipos de metas: Objetivos marcados por los requisitos del software y Requisitos funcionales y no funcionales.

2.1. Objetivos marcados por los requisitos del software

Requisitos funcionales y no funcionales

El sistema está diseñado para cumplir con requisitos funcionales específicos que permiten a los usuarios cargar imágenes para su análisis y procesamiento. Esto incluye la carga para entrenamiento y predicción, el preprocesamiento de imágenes y la segmentación usando el algoritmo Mask-Method MRCNN. Además, proporciona una interfaz de usuario intuitiva y la visualización de los resultados de forma comprensible.

Educación y usabilidad

Un enfoque clave del sistema es educar a los usuarios sobre el proceso de segmentación de imágenes y redes neuronales. Por ende, el sistema incluye una sección educativa y está diseñado para ser intuitivo, asegurando que todos los usuarios puedan aprender a usar la aplicación web en menos de 10 minutos.

2.2. Objetivos de carácter técnico

Programación en Python

Dar conocimiento al lector sobre el lenguaje de programación python además del manejo del IDE Spyder siendo capaz este de gestionar y crear proyectos.

Desarrollo de modelos y uso de datos

El proyecto tiene como objetivo desarrollar un modelo de detección de insectos basado en el uso de redes neuronales y Mask-Method MRCNN, utilizando un conjunto de datos del repositorio 4TU.ResearchData. Los datos utilizados son imágenes etiquetadas de insectos recolectados en invernaderos comerciales, lo cual es esencial para entrenar algoritmos de aprendizaje profundo en tareas de reconocimiento y clasificación de insectos.

Antes y después del entrenamiento

Se utilizan pesos preentrenados del conjunto de datos MS COCO, que sirven como punto de partida para el modelo Mask R-CNN. Después del entrenamiento, el modelo puede ser ajustado y mejorado en función de los resultados obtenidos para garantizar su eficacia en la detección de insectos.

Metodología y planificación

Para la estructuración del trabajo, se ha elegido la metodología Kanban por su eficiencia en la representación mediante tableros y la mejora continua del flujo de trabajo. Se han definido un total de ocho epics para la planificación y ejecución del proyecto, que incluyen desde la configuración del entorno de desarrollo hasta la creación de la interfaz web y la documentación final del proyecto.

Viabilidad y expectativas

La viabilidad económica del proyecto se ha evaluado y categorizado en costes directos, indirectos, fijos y variables, asegurando así que el proyecto es económicamente factible y sostenible.

3. Conceptos teóricos

En esta sección, abordaremos de manera teórica el proceso de detección y clasificación de insectos profundizando en *instance segmentation* la cual utiliza Mask RCNN además de los métodos previos que han ido evolucionando hasta llegar al destacado. Siendo estos:

1. RNA . Redes neuronales artificiales
2. CNN. red neuronal convolucional
3. R-CNN.region basada en una red convolucional
4. Faster R-CNN. Faster Region based convolutional network
5. Mask R-CNN. Mask (faster) Region-based Convolutional Network (RoIAlign)

Ademas posteriormente de los diferentes procesos desde la adquisición inicial de las imágenes hasta la identificación de los insectos, detallando cada fase crucial que compone este sistema automatizado. Siguiendo con el siguiente apendice, hablando así sobre las herramientas que han hecho posible que este proceso se realice de manera efectiva y eficiente.

3.1. Redes neuronales

Una red neuronal artificial es una simulación por computadora que intenta modelar los procesos del cerebro humano para imitar la forma en que aprende. [4].

Las redes neuronales artificiales se estructuran a partir de múltiples nodos de procesamiento, conocidos como neuronas, interrelacionados entre sí. Estos nodos colaboran de manera distribuida para procesar la información de entrada, aprendiendo de manera conjunta para mejorar la precisión de los resultados obtenidos.

Cargaríamos la entrada, generalmente en forma de un vector multidimensional, en cuya capa de entrada la distribuiremos a las capas ocultas. Luego, las capas ocultas tomarán decisiones a partir de la capa anterior y sopesarán cómo un cambio estocástico dentro de sí mismo perjudica o mejora el resultado final, y esto se conoce como proceso de aprendizaje. [5]

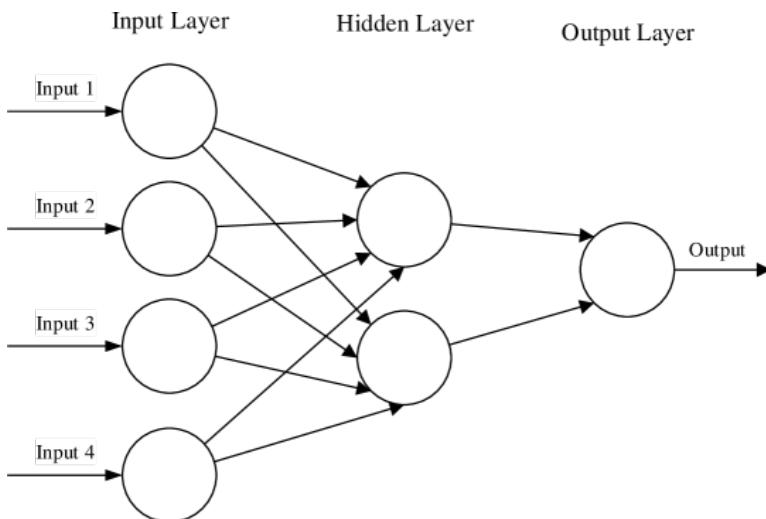


Figura 3.1: red neuronal *feedforward (FNN)* simple de tres capas, compuesta por una capa de entrada, una capa oculta y una capa de salida. [5]

Cada nodo individual se puede considerar como su propio modelo de regresión lineal, formado por datos de entrada x_i , ponderaciones w_i , un sesgo (o umbral) y una salida. La fórmula sería similar a la siguiente:

$$\sum w_i x_i + \text{sesgo} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{sesgo}$$

La salida se determina de la siguiente manera:

$$\text{salida} = f(x) = \begin{cases} 1 & \text{if } \sum w_i x_i + b \geq 0 \\ 0 & \text{if } \sum w_i x_i + b < 0 \end{cases}$$

Épocas

Una época significa entrenar la red neuronal con todos los datos de entrenamiento durante un ciclo. En una época, utilizamos todos los datos exactamente una vez.^[6] En ese momento del proceso, el modelo realiza ajustes a sus parámetros internos tomando como referencia la diferencia entre los valores predichos y los reales.

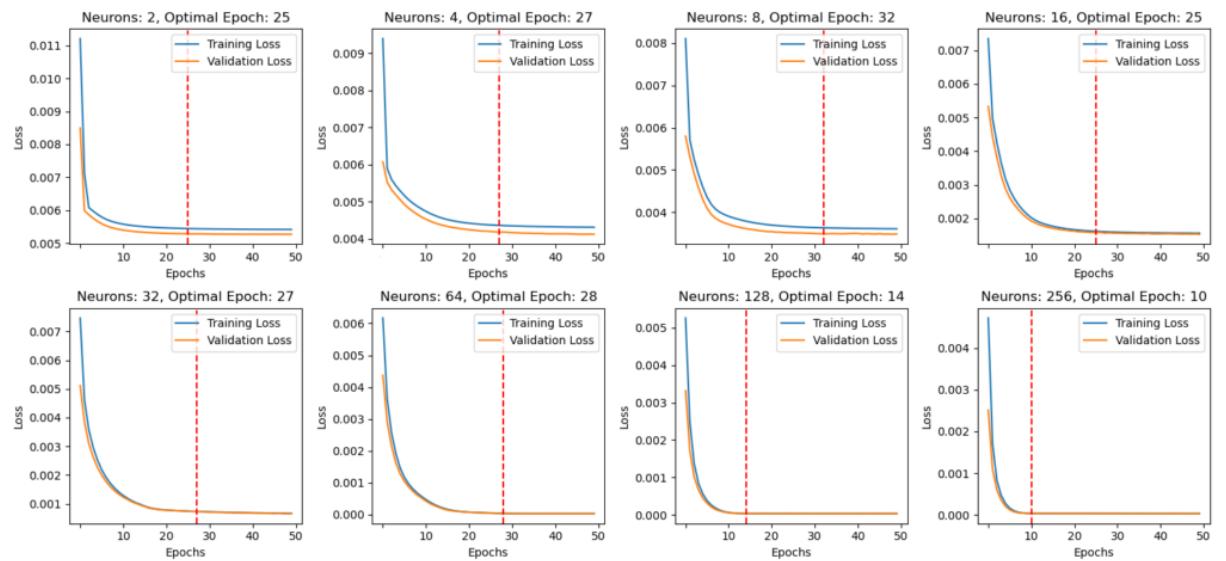


Figura 3.2: Comparativa del error segun el número de épocas y neuronas [9]

Determinar la cantidad óptima de épocas es esencial por varias razones críticas:

- **Aprendizaje Incompleto:** Un número insuficiente de épocas podría no ser suficiente para que el modelo capture y aprenda efectivamente los patrones en el conjunto de datos de entrenamiento.
- **Sobreentrenamiento:** Un número excesivo de épocas puede llevar al modelo a memorizar los datos de entrenamiento, afectando negativamente su habilidad para generalizar a nuevos datos no observados.
- **Gasto Ineficiente de Recursos:** Asignar más épocas de las necesarias resulta en el uso innecesario de recursos computacionales, lo cual puede alargar el tiempo de entrenamiento sin proporcionar mejoras significativas en el rendimiento del modelo.

Tasa de aprendizaje

La tasa de aprendizaje es un hiperparámetro que controla cuánto cambiar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo. Elegir la tasa de aprendizaje es un desafío ya que un valor demasiado pequeño puede resultar en un proceso de entrenamiento largo que podría atascarse, mientras que un valor demasiado grande puede resultar en aprender un conjunto de pesos subóptimo demasiado rápido o un proceso de entrenamiento inestable. [8] Aquí se presentan algunas estrategias para configurarla:

- **Velocidad de Aprendizaje Estática:** Se mantiene constante a lo largo del entrenamiento, lo que puede resultar en un entrenamiento subóptimo.
- **Disminución de la Velocidad de Aprendizaje:** Reduce la velocidad de aprendizaje a lo largo del tiempo, lo cual puede ayudar en la convergencia del modelo.
- **Velocidad de Aprendizaje Adaptativa:** Métodos como Adagrad, RMSprop y Adam ajustan la velocidad individualmente por parámetro, basándose en la historia de los gradientes.
- **Programación de Velocidad de Aprendizaje:** Ajusta la velocidad según un cronograma establecido o en respuesta al rendimiento del modelo durante la validación.
- **Velocidad de Aprendizaje Cíclica:** Oscila entre dos valores, permitiendo tanto la exploración del espacio de pesos como la convergencia durante el entrenamiento.

La selección adecuada y el ajuste de la velocidad de aprendizaje son cruciales para la eficacia del modelo, su rapidez en converger y su habilidad para generalizar tras el entrenamiento.

Funciones de activación

Una vez que se determina una capa de entrada, se asignan ponderaciones. [18] A la hora de determinar la activación de una propia neurona esta utilizará funciones de activación. Hablaremos principalmente de las funciones de activación más utilizadas a la hora de la clasificación, siendo un resultado de positivo(activación) o negativo (no activación). Se destacan las siguientes:

- **Función de activación sigmoidea.** La función sigmoidea se usaba tradicionalmente para problemas de clasificación binaria (sigue la línea de "si $x \leq 0,5$, $y = 0$, si no, $y = 1$ "). Pero tiende a causar un problema de gradientes que desaparecen y, si los valores están demasiado cerca de 0 o +1, la curva o el gradiente es casi plano y, por lo tanto, el aprendizaje sería demasiado lento. [2]

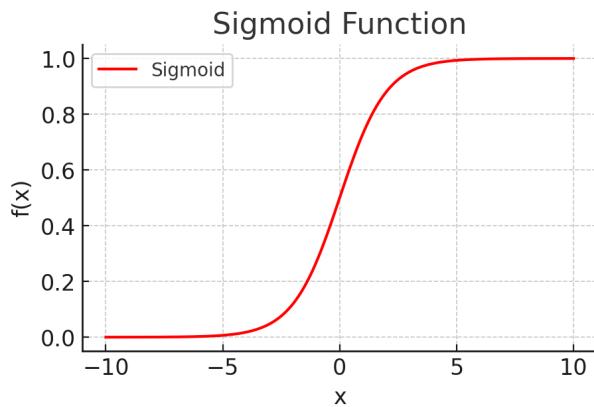


Figura 3.3: Función de activación sigmoidea

- **Función de activación Tanh.** Es diferente del sigmoide en el sentido de que está centrado en cero y, por lo tanto, restringe los valores de entrada entre -1 y +1. Es incluso más costoso desde el punto de vista computacional que sigmoide, ya que implica muchas operaciones matemáticas complejas, que deben realizarse para cada entrada e iteración, repetidamente.[2]

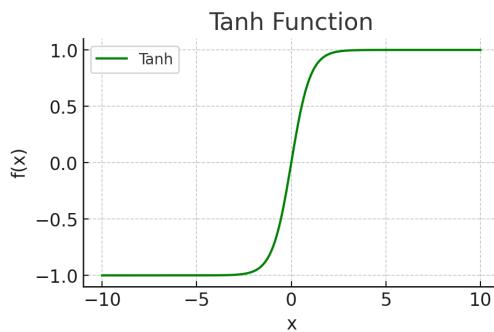


Figura 3.4: Función de activación Tanh

- **Función de activación ReLU.** ReLU es una función de activación no lineal famosa y ampliamente utilizada, que significa Unidad Lineal Rectificada (va más o menos como “si $x \leq 0$, $y = 0$ sino $y = 1$ ”). Por tanto, sólo se activa cuando los valores son positivos. ReLU es menos costoso computacionalmente que tanh y sigmoide porque implica operaciones matemáticas más simples. [2]

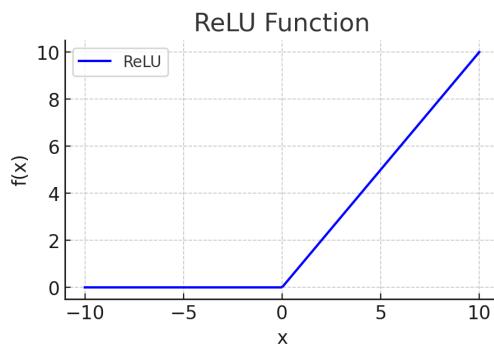


Figura 3.5: Función de activación ReLU

Función de perdida

La función de pérdida ayuda a determinar la eficacia con la que su algoritmo modela el conjunto de datos presentado. De manera similar, la pérdida es la medida que tiene su modelo de previsibilidad, los resultados esperados. Las pérdidas generalmente pueden clasificarse en dos categorías amplias relacionadas con problemas del mundo real: clasificación y regresión. Debemos predecir la probabilidad para cada clase a la que se refiere el problema. Sin embargo, en la regresión tenemos la tarea de pronosticar un valor constante para un grupo específico de características independientes.[29] Viene expresada por la fórmula:

$$CE = - \sum_{i=1}^C t_i \log(s_i)$$

En resumidas evalúa los valores de salida esperados y los valores predichos; Evalúa qué tan efectivamente la red neuronal representa los datos de entrenamiento. Durante el proceso de entrenamiento, nuestro propósito es reducir al mínimo esta discrepancia entre las predicciones y las metas establecidas modificando así sus respectivos pesos.

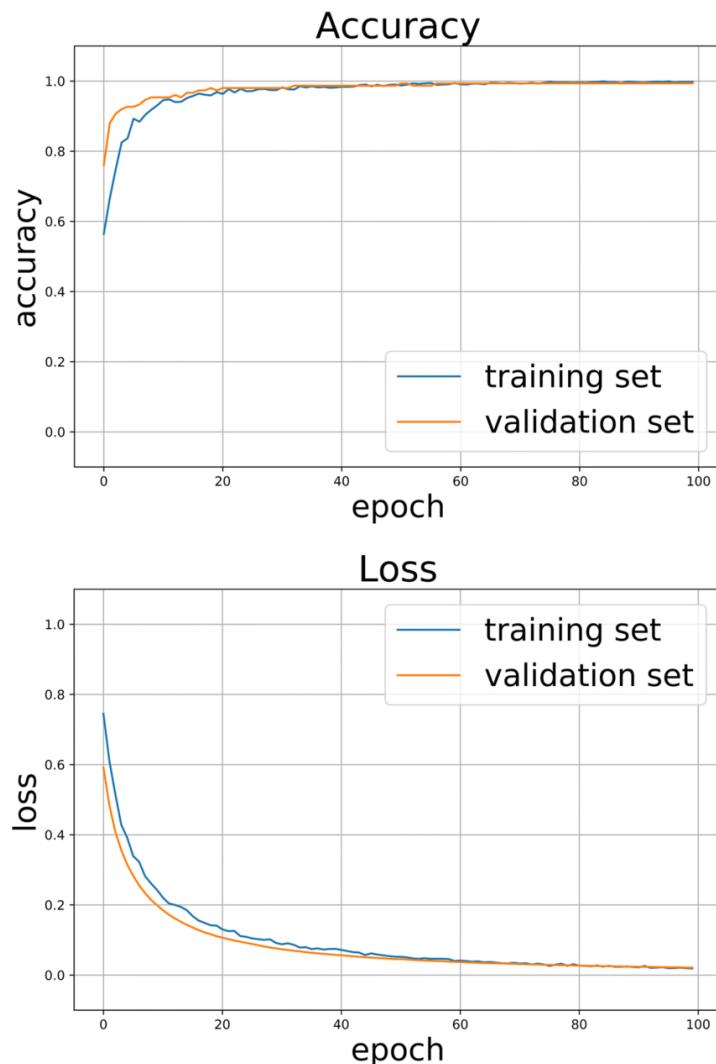


Figura 3.6: Gráfica comparativa precision y perdida

[32]

Accuracy siendo el acierto del modelo en las predicciones, como podemos ver en la gráfica la comparativa entre el acierto con el conjunto de validación y prueba a medida que aumentan las épocas este se hace mas preciso mostrando mejores resultados con un error menor.

3.2. Redes Neuronales Convolucionales - CNNs

Una red neuronal convolucional (CNN), también conocida como ConvNet, es un tipo especializado de algoritmo de aprendizaje profundo diseñado principalmente para tareas que requieren el reconocimiento de objetos, incluida la clasificación, detección y segmentación de imágenes. Las CNN se emplean en una variedad de escenarios prácticos, como vehículos autónomos, sistemas de cámaras de seguridad y otros. [10]

En la estrategia inicial de clasificación de imágenes, se solía emplear el uso directo de los píxeles para llevar a cabo la clasificación. Por ejemplo, para diferenciar entre diversas razas de perros, se podría utilizar la información de color a través del histograma de colores de los píxeles en la imagen, así como la forma de las orejas mediante la detección de bordes. A pesar de que este método puede arrojar resultados aceptables en situaciones simples, la extracción de características más complejas permite abordar desafíos más complicados.

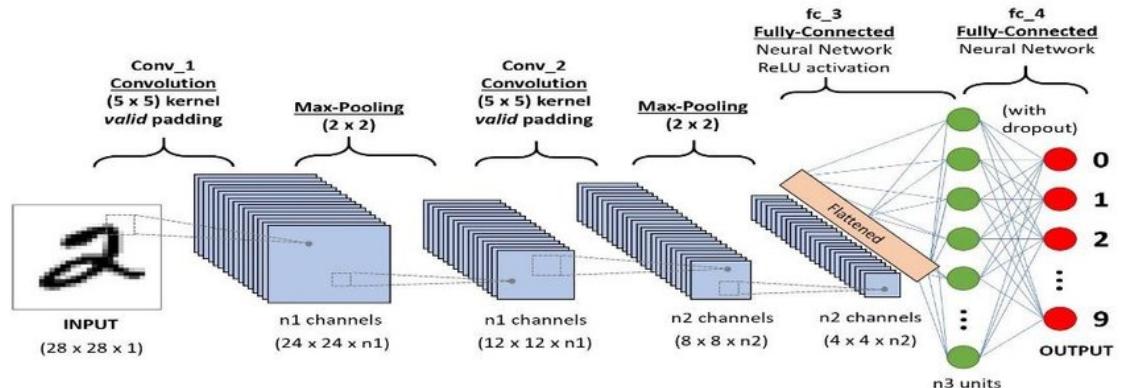


Figura 3.7: Secuencia de una CNN paso a paso [26]

Como podemos ver en la figura 3.7 corresponde a una CNN capaz de determinar a partir de una imagen/dibujo de un numero que numero es este teniendo unas posibles respuestas del 0 al 9 en la salida.

En nuestro caso tendremos como entrada una imagen RGB por lo tanto una imagen con 3 matrices, en las que en cada una de estas se representaran valores del 0 al 255 para las intensidades de los colores rojo, verde y azul RGB. De la siguiente forma: En cuando a las CNN es muy importante ser

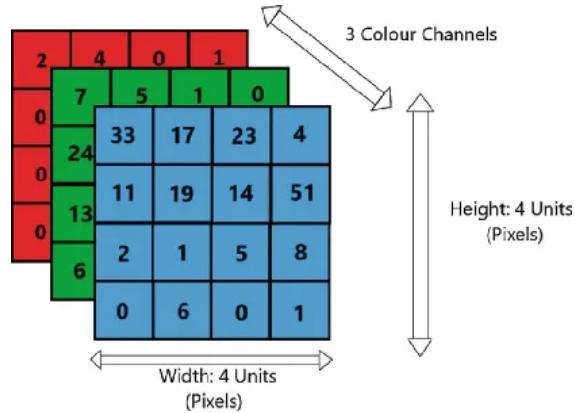


Figura 3.8: Secuencia de entrada RGB en CNN [24]

conocedor de 2 acciones fundamentalas que realizan estas. La aplicación de la *convolución* y el *pooling*

- Convolución. Obtendrá información relevante a partir de la matriz de entrada multiplicando por los valores del filtro K

Como podemos ver en la figura 3.10 el kernel es de 3x3 mientras que

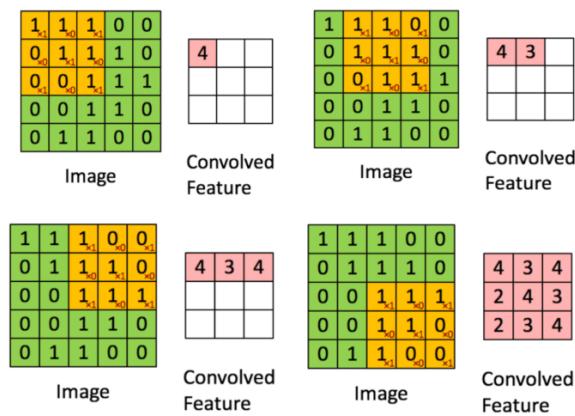


Figura 3.9: Aplicación de convolución[11]

la matri de entrada es de 5x5. A la hora de moverse el kernel por

la matriz de entrada se moverá en función del Stride Longitud, en este caso siendo de 1 moviéndose un total de nueve veces a la hora de recorrer la matriz de entrada. Con los siguientes valores del filtro K :

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

- Pooling. A partir del kernel podremos hacer 2 tipos de agrupaciones:
 1. Average Pooling. Agrupando conjunto de celdas de la matriz es capaz de calcular el promedio para cada uno de los determinados conjunto de celdas obteniendo así un número reducido conformado por el promedio de estas
 2. Max Pooling. Al igual que en el Average Pooling pero seleccionando las celdas de los grupos de la matriz mayores

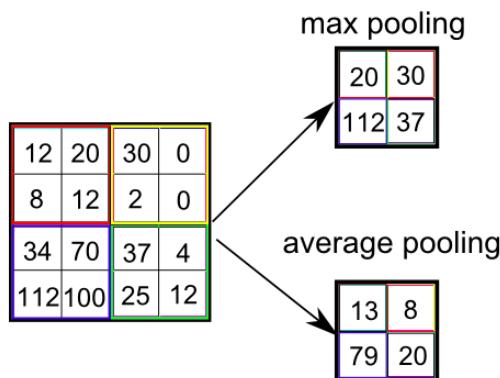


Figura 3.10: Demostración gráfica de Max Pooling y Average Pooling

Como podemos ver en la figura 3.10 el tamaño de la formación de los grupos estará definido por el valor Stride que en este caso tendrá un valor de 2.

Convertido nuestra imagen de entrada a una forma adecuada para nuestro perceptrón multinivel, aplanaremos la imagen en un vector de columna. La salida aplanada se envía a una red neuronal de retroalimentación y se aplica retropropagación a cada iteración del entrenamiento. A lo largo de una serie de épocas, el modelo es capaz de distinguir entre características dominantes y ciertas de bajo nivel [27]

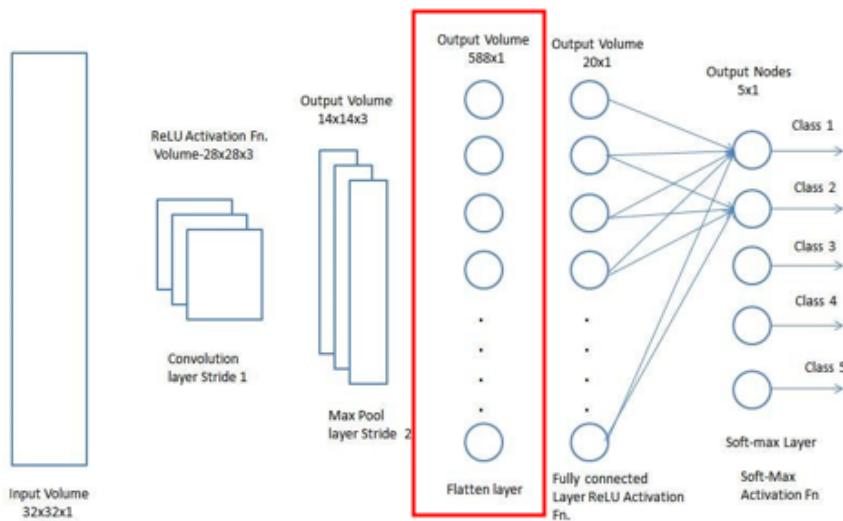


Figura 3.11: Capa flatten en la estructura de red CNN[25]

A partir de las características obtenidas en las diferentes capas de la red CNN, será capaz de obtener mediante la Red neuronal el resultado final, activándose la neurona oportuna mediante su función de activación como puede ser la función ReLu (1 activa , 0 desactivada). Teniendo tantas neuronas en la última capa como clases a detectar en la imagen.

3.3. R-CNN – región basada en una red convolucional

Estas a diferencia de la anterior CNN, utilizan tambien la identificacion de regiones, agrupando asi las caracteristicas en una misma region y luego sacando informacion de estas. Por lo tanto tendremos una CNN por cada una de las regiones propuestas. Teniendo asi las siguientes fases:

1. Obtendra la imagen de entrada.
2. Generacion en la imagen de regiones propuestas
3. Red CNN la cual permitira la obtencion de caracteristicas
4. Clasificacion de las regiones, mediante RNA y la activacion de las neuronas

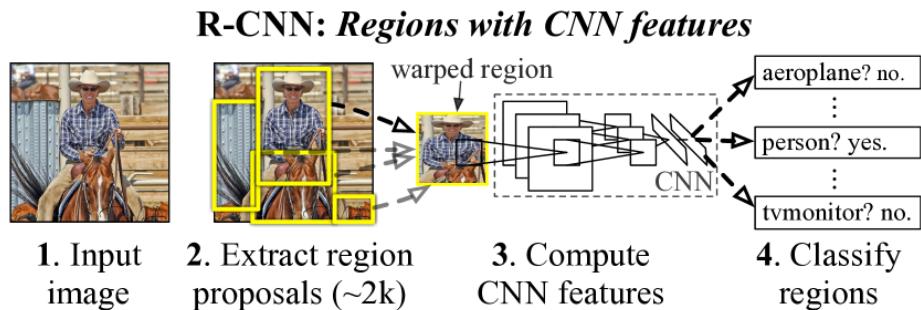


Figura 3.12: Descripción general de la estructura de red CNN[15]

3.3. R-CNN – REGIÓN BASADA EN UNA RED CONVOLUCIONAL21

Para la generación de las regiones hay varios algoritmos que permiten la definición de estas regiones propuestas:

- **Algoritmo de búsqueda selectiva.** El cual Combina recursivamente las regiones similares más pequeñas en otras más grandes. Usando el algoritmo Greedy para combinar regiones similares para hacer regiones más grandes [12]

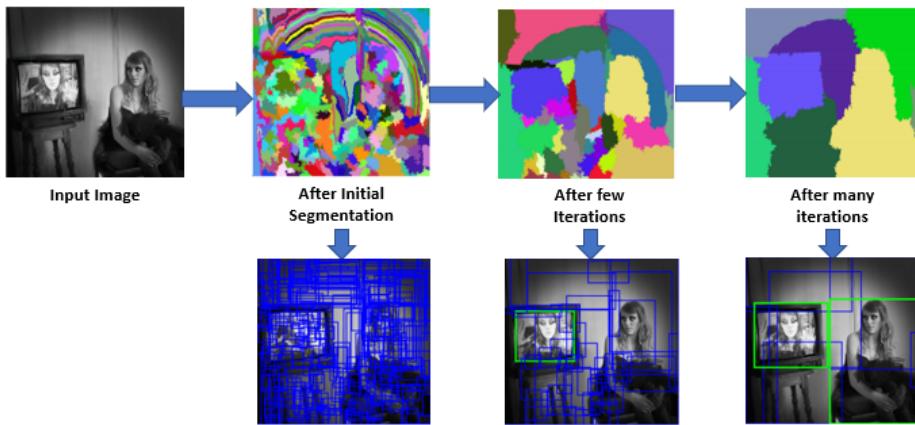


Figura 3.13: Representación gráfica de el algoritmo Greedy en búsqueda selectiva[12]

- **Region Proposal Network.** utilizan cajas de anclaje como referencias en múltiples escalas y relaciones de aspecto, lo que permite predecir eficientemente propuestas de región con un amplio rango de escalas y proporciones. [22]

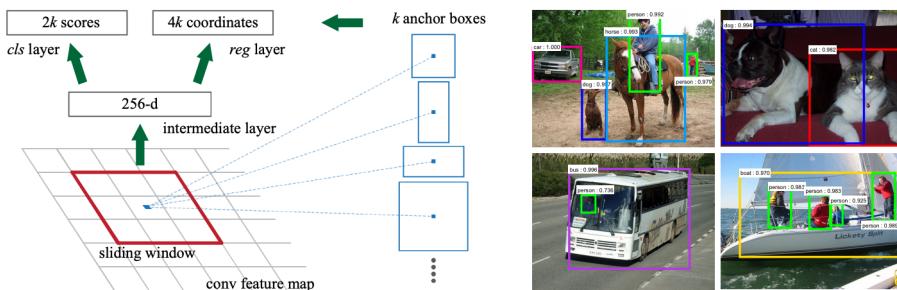


Figura 3.14: A la izquierda, la Red de Propuestas de Región (RPN). A la derecha, se presentan ejemplos de detecciones utilizando propuestas RPN[22]

3.4. Fast R-CNN – Fast Region based convolutional network

Tras la aparición de R-CNN se llegaron a producir mejoras sobre este, surgiendo así el modelo Fast R-CNN. En lugar de extraer características de CNN de forma independiente para cada región de interés, Fast R-CNN las agrega en un único paso hacia adelante sobre la imagen; es decir, las regiones de interés de la misma imagen comparten cálculo y memoria en los pases hacia adelante y hacia atrás.

En lugar de extraer características de CNN de forma independiente para cada región de interés, Fast R-CNN las agrega en un único paso hacia adelante sobre la imagen; es decir, las regiones de interés de la misma imagen comparten cálculo y memoria en los pases hacia adelante y hacia atrás.[14]

El sistema está compuesto por dos módulos. El primer módulo es una red convolucional profunda totalmente que propone regiones, y el segundo módulo es el detector Fast R-CNN que utiliza las regiones propuestas. Todo el sistema es una única red unificada para la detección de objetos. Utilizando la terminología recientemente popular de redes neuronales con mecanismos de "atención", el módulo RPN le indica al módulo Fast R-CNN dónde mirar.

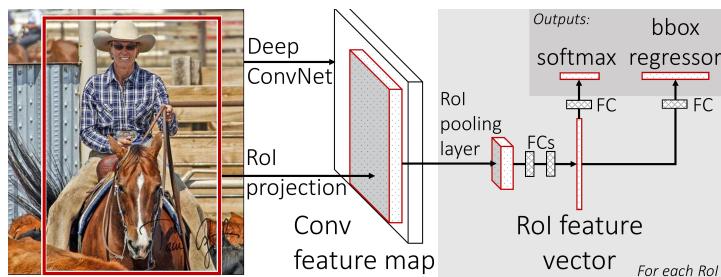


Figura 3.15: modelo Fast R-CNN[13]

Explicación de los elementos para entender su mejor funcionamiento:

- **Arquitectura ConvNet Profunda:** La esencia de este enfoque radica en el uso de una red convolucional de múltiples capas que procesa de manera exhaustiva la imagen entera, destilando un conjunto de mapas de características. Estos mapas actúan como representaciones detalladas que capturan los atributos visuales cruciales dispersos por toda la imagen.
- **Proyección de Región de Interés (RoI):** Cada región demarcada como de interés es mapeada hacia el espacio de características convolucional previamente obtenido. La ubicación precisa de cada RoI es esencial y se determina dentro del contexto del mapa de características.
- **Capa de Pooling RoI:** Posterior a la proyección, sigue la aplicación de la capa de pooling sobre las RoIs. El propósito de esta operación es condensar las dimensiones de los atributos de las RoIs a una escala uniforme, simplificando el tratamiento subsecuente. Este procedimiento se replica para cada RoI identificada en el mapa.
- **Vector de Características RoI:** Los atributos de cada RoI, una vez comprimidos, se transforman en un vector único de características. Este vector funciona como un compendio eficiente de la información visual inherente a cada región de interés seleccionada.
- **Salidas del Modelo:** El flujo de información se conduce hacia una serie de capas densamente conectadas (FC), culminando en la producción de dos resultados por cada RoI:
 - **Regresor de Cajas Delimitadoras:** Funciona ajustando los contornos de las cajas delimitadoras proyectadas para coincidir con la localización exacta de objetos dentro de la RoI.
 - **Clasificador Softmax:** Este clasificador estima la probabilidad de que cada RoI contenga un objeto y, en caso afirmativo, procede con su clasificación.

3.5. Mask R-CNN

Una vez comprendidos los términos y modelos anteriores podemos dar paso al aprendizaje sobre Mask R-CNN la cual es la que utilizará este proyecto.

Mask R-CNN es una arquitectura avanzada de red neuronal diseñada para llevar a cabo la identificación y delimitación precisa de elementos individuales en imágenes, a través del proceso conocido como segmentación de instancias. Esta técnica es esencial para diferenciar y segmentar múltiples elementos de una misma categoría dentro de una imagen, proporcionando una comprensión detallada de su contexto y relaciones espaciales. [30]

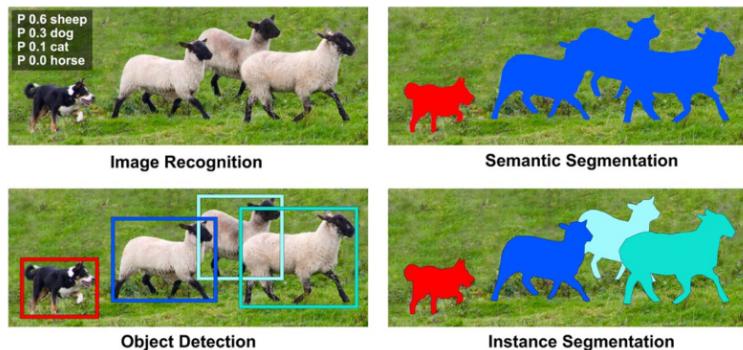


Figura 3.16: Comparativas entre *Image recognition* , *Semantic segmentation* ,*Object detection* y *Instance segmentation* [28]

La Figura 3.1 muestra las distinciones entre la detección de objetos, la segmentación semántica y la segmentación de instancias. La segmentación de instancias fusiona los dos últimos enfoques para ofrecer una identificación y separación minuciosa de cada elemento individual.

Basicamente Mask R-CNN es una extensión de Fast R-CNN pero prediciendo una máscara binaria para cada clase de forma independiente, sin competencia entre clases, basandose en la rama de clasificación RoI de la red para predecir la categoría.

Máscaras

A la hora de recibir un input (imagen) se generarán un *bounding box* y una *máscara* dentro de este *bounding box* para cada una de las regiones propuesta *RPN*.



Figura 3.17: Primera máscara sobre un objeto a detectar

La máscara generada por primera vez tendrá valor de 0 conformando todo el espacio del *bounding box* hasta el límite de este como podemos ver en la figura 3.17

Una vez que el modelo es entrenado será capaz de establecer la máscara a los bordes de la figura a detectar dando un valor a la forma en la máscara de 0 y al resto del espacio entre la figura y el *bounding box* de 1.

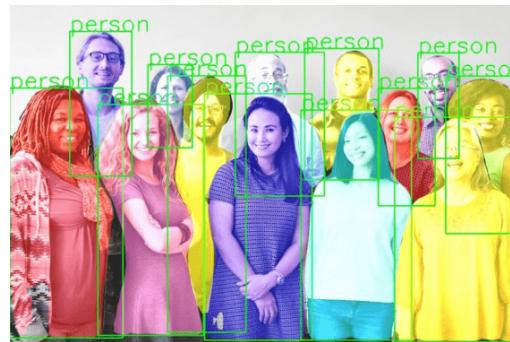


Figura 3.18: Bounding boxes y máscaras generadas por un modelo Mask R-CNN [20]

Como podemos ver en la figura el modelo guardará el *bounding box* y la máscara de cada uno de los elementos detectados, en este caso personas.

ROiAlign

RoIAlign es particularmente beneficioso porque evita la cuantización de las coordenadas de la región de interés durante el proceso de pooling. Como podemos ver en el siguiente caso perderíamos información a la hora de hacer el *Pooling*

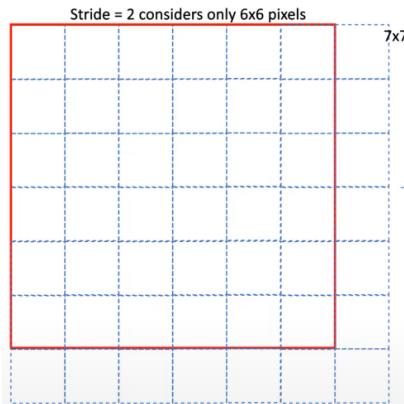


Figura 3.19: Perdida al hacer
Max Pooling

Como podemos ver en el siguiente caso se toman cuatro muestras dentro de cada bin utilizando interpolación bilineal, lo que resulta en un alineamiento exacto y conservación de detalles. Esto se traduce en una segmentación más precisa a nivel de píxel.

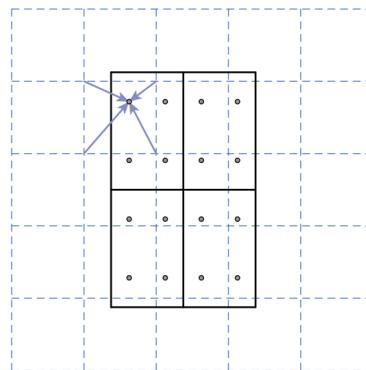


Figura 3.20: Ejemplo de stride utilizando interpolación bilineal[16]

Arquitectura

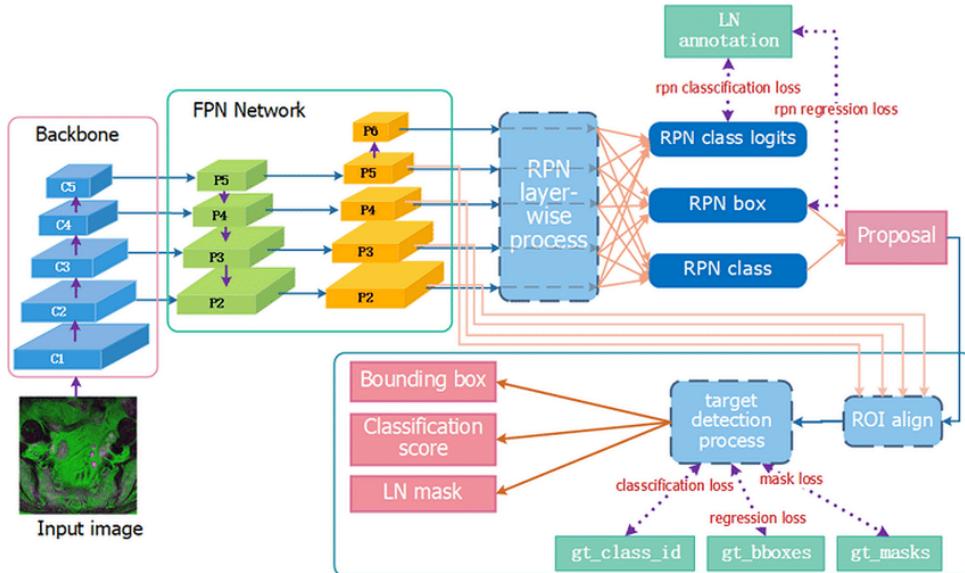


Figura 3.21: Arquitectura Mask R-CNN

Esta arquitectura de Mask R-CNN se compone de varias partes clave:

1. **Backbone**: Es la base de la red neuronal que procesa la imagen de entrada y extrae las características.
2. **FPN Network**: Toma las características extraídas del backbone y las procesa a diferentes escalas para capturar detalles a varios niveles.
3. **RPN Layer-wise Process**: La Red de Propuestas de Región (RPN) utiliza las características de FPN para proponer candidatos a objetos que podrían estar presentes en la imagen.
4. **Proposal**: Las propuestas generadas se pasan para su procesamiento.
5. **ROI Align**: Esta etapa alinea precisamente las regiones de interés con las características de la imagen.
6. **Bounding Box, Classification Score, LN Mask**: Estos son los outputs finales que incluyen las cajas delimitadoras para cada detección, los puntajes de clasificación para cada objeto detectado y las máscaras que segmentan los objetos a nivel de píxel.

3.6. Cuatro fases fundamentales

1. **Pre-procesamiento:** Esta fase involucra la preparación inicial de las imágenes recolectadas(dataset). Se realizan ajustes y mejoras, como la normalización del tamaño de las imágenes.
2. **Codificación:** En esta fase, las imágenes son transformadas de su forma original a un formato que el modelo de IA puede procesar eficientemente. Esto generalmente implica la conversión de imágenes a un conjunto de características o tensores que sirven como entrada al modelo de redes neuronales convolucionales (CNN).
3. **Detección y Clasificación:** Aquí es donde el modelo de IA entra en acción. Utilizando los datos procesados, el modelo realiza la identificación y clasificación de los insectos presentes en las imágenes. Esta fase es crucial y es el núcleo del proceso de detección de insectos.
4. **Post-procesamiento:** Finalmente, los resultados generados por el modelo de IA pueden necesitar ajustes adicionales para mejorar la clasificación de las especies detectadas. En esta etapa, se corrigen posibles errores y se perfeccionan los resultados antes de su presentación final.

En las subsecciones siguientes, discutiremos con mayor detalle cada una de estas etapas y su importancia en el proceso de detección de insectos mediante técnicas avanzadas de IA.

Pre-procesamiento

Pre-procesamiento

Detección y Clasificación

Post-procesamiento

Herramientas	App	Python	API REST	BD	Memoria
HTML		X			
CSS		X			
FLASK		X			
Python		X			
AngularJS		X			
Bower		X			
PHP			X		
Karma + Jasmine		X			
Slim framework			X		
Idiorm			X		
Composer			X		
JSON		X	X		
PhpStorm		X	X		
MySQL				X	
PhpMyAdmin				X	
Git + BitBucket	X		X	X	X
MikT _E X					X
T _E XMaker					X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

3.7. Herramientas y tecnologías

A continuación se muestra una tabla con las herramientas en las diferentes partes del proyecto AppPython ,API REST, BD y Memoria

4. Técnicas y herramientas

técnicas, metodologías, y herramientas de desarrollo

Dentro del espectro de metodologías a disposición, sobresalen las Redes Neuronales Convolucionales (CNN), Mask R-CNN, YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector), cada una distinguiéndose por sus características específicas y sus respectivas áreas de aplicación y restricciones. Este apendice procede a realizar un análisis comparativo de dichas técnicas, culminando en la explicación de por qué se prefirió Mask R-CNN para este estudio en particular. H

- **CNN.** En el ámbito de la segmentación de imágenes han cumplido con los propósitos para los que se utilizaban sin embargo en determinados casos se las considera "antiquadas" debido a una serie de razones como son:
 1. Gran Conjuntos de DataSet. Para un buen funcionamiento de las CNN necesitaremos gran cantidad de datos para las diferentes fases del entrenamiento y validación , esto en algunos casos puede suponer un gran coste de recursos y tiempo.
 2. Alta Complejidad Computacional. Es común una gran cantidad de capas e interconexiones entre estas esto implica una alta complejidad computacional limitándose a equipos con grandes recursos de procesamiento , siendo estos no disponibles para dispositivos como pueden ser sistemas embebidos.
 3. Dificultades para adaptarse a nuevas situaciones no vistas durante el entrenamiento, lo cual limita su eficacia en la localización visual en espacios no previamente mapeados.[23]

4. Evolucion de otros arquitecturas. A lo largo de los últimos años han surgido nuevas arquitecturas que han demostrado mejores resultados respecto a las CNN quedando estas en segundo plano
5. Confusión con el fondo. confunde los parches de fondo de una imagen con objetos porque no puede ver el contexto más amplio

Por lo general CNN cumple con el objetivo pero a día de hoy puede ser mejorable habiendo nuevas arquitectura que dan un mejor rendimiento.

- **YOLO** se entrena con imágenes completas y optimiza directamente el rendimiento de la detección. Este modelo unificado tiene varias ventajas sobre los métodos tradicionales de detección de objetos como son:[21]

1. Extremadamente rápido. Simplemente ejecutamos nuestra red neuronal en una nueva imagen en el momento de la prueba para predecir las detecciones
2. Razona globalmente sobre la imagen a la hora de hacer predicciones. Ve la imagen completa durante el entrenamiento y el tiempo de prueba, por lo que codifica implícitamente la información contextual sobre las clases, así como su apariencia.
3. Altamente generalizable, es menos probable que se rompa cuando se aplica a nuevos dominios o entradas inesperadas.

YOLO todavía está por detrás de los sistemas de detección de última generación en cuanto a precisión. Presentando las siguientes desventajas:[21]

1. Dificultad en datos nuevos o inusuales.
2. Dificultan en predicción de objetos pequeños.
3. limitaciones significativas en el espacio de las predicciones de los marcos delimitadores identificando solo una categoría por celda.

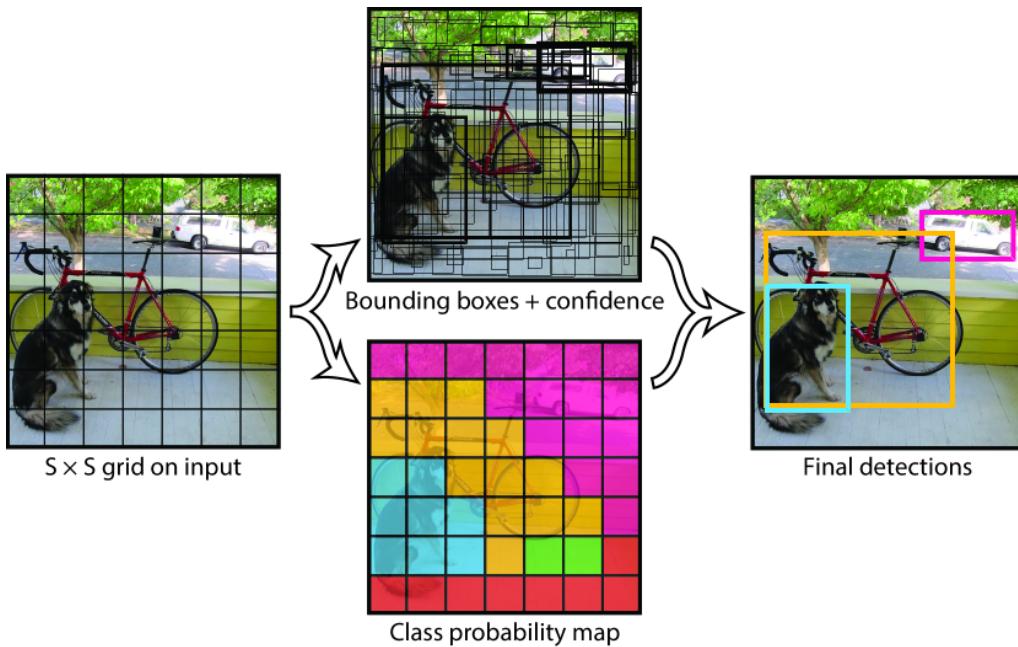


Figura 4.1: proceso de detección de objetos utilizado por la arquitectura de red neuronal YOLO[21]

- **SSD** Otra método de detección de objetos que estubimos planteando es la SSD (Single Shot MultiBox Detector), este es el segundo mas conveniente ya que es mejor que YOLO por motivos como los siguientes: [17]
 1. Mayor rapidez. un detector de disparo único para múltiples categorías que es más rápido que el estado del arte anterior para detectores de disparo único.
 2. Mayor precisión. Para la rapidez que tiene de detección de objetos tiene bastante buena precisión guardando así una cierta calidad rapidez/precisión.
 3. Entrenamiento simple de un extremo. Incluso en imágenes de entrada de baja resolución, lo que mejora aún más la relación entre velocidad y precisión.
 4. Mayor portabilidad. Al tratarse de un sistema tan rápido y que necesita pocos recursos este es mas portable llegando así a poder implementarse en sistemas embebidos o dispositivos móviles.

Una vez destacados los puntos positivos destacaremos los negativos que nos llevaron a la conclusión de mejor utilizar Mask R-CNN.

1. Detección de objetos pequeños. Al igual que le pasa a YOLO, tiene dificultades a la hora de la detección de elementos pequeños.
 2. Mal asignación de las bounding boxes. Un mal establecimiento de las cajas delimitadoras puede suponer gran perdida de información
 3. Baja sensibilidad en variación de datos. La eficaz de este sistema puede verse comprometido en caso de tener un dato de entrada real en el que la iluminación o texturas sean diferentes.
 4. Implementación óptima. Para que SSD sea eficaz y merezca la pena esta requerirá una implementación compleja.
- **Mask R-CNN** Tras la realización de un estudio previo para la elección de el sistema a utilizar nos decantamos por Mask R-CNN ya que este permite una mejor precisión a la hora de la detección de los elementos que son de tamaño reducido como es nuestro caso, ya que estamos tratando de identificar los insectos. El entrenamiento y proceso de detección es mayor porque requiere un análisis pixel a pixel, por lo tanto nos hemos decatado mas en eficiencia que en rapidez.

5. Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Para el desarrollo de el proyecto se ha de revisar antes el proceso de instalación de *python* , *CUDA toolkit*(ficheros dll para utilización de la GPU), *Mask R-CNN*, *IDE* , etc ... Para ello acudir al anexo *Manual de Usuarios*.

ETIQUETADO HERRAMIENTA INFORMACION DE CADA OBJETO

5.1. Datos

A la hora de buscar los datos con los que entrenaremos nuestro modelo, tuvimos en cuenta principalmente que tendría que ser un dataset de insectos cuya presencia es común en plagas para ello tuvimos varias opciones:

1. IP102 dataset. Dataset muy grande con 7500 imágenes contando con 102 categorías diferentes (insectos distintos) [31].
Este fue un posible dataset ya que tenía gran cantidad de imágenes, al final no fue elegido por el hecho de que las imágenes tenían distintos fondos y elementos que podrían ser incovenientes a la hora del entrenamiento y detección de insectos. Para acceder al dataset acceder a:
<https://github.com/xpwu95/IP102>

2. 4tu.ResearchData. Es una reconocida infraestructura integral dedicada al manejo de datos de investigación en las áreas de ciencia, ingeniería y diseño. [1]

Gracias a 4tu.ResearchData encontramos el dataset ideal para nuestro caso ya que las imágenes están tomadas sobre un adhesivo amarillo el cual hace distintivos a las formas de los insectos. Este además nos facilita el etiquetado con su correspondiente image.

Para acceder al dataset podremos hacerlo mediante el enlace:

https://data.4tu.nl/articles/dataset/Raw_data_from_Yellow_Sticky_Traps_with_insects_for_training_of_deep_learning_Convolutional_Neural_Network_for_object_detection/12707066

Adecuación de los datos

Antes de realizar el entrenamiento de nuestro modelo necesitamos adecuar los nombres de los datos ya que estos en nuestro código corresponderá con el ID de imagen yendo así desde el 0 al 284, teniendo así la imagen (en .jpg) y su extensión con el mismo renombre (en.xml)

Para ello utilizamos un script llamado "`rename_file_to_numbers`". El cual además de darles el renombre correcto las cambia de posiciones aleatorias para que nuestro modelo no coja siempre los mismos datos en mismas posiciones durante el entrenamiento evitando así el Overfitting¹



Figura 5.1: Ejemplo de la imagen 001.jpg y su etiquetado 001.xml

Como podemos ver en la imagen 5.1 tendremos una etiqueta por cada imagen y en la propia etiqueta los objetos presentes en esta, como podemos

¹El overfitting es el proceso en el que el modelo aprende predicciones de memoria luego siendo inadecuado para datos reales.

ver cada objeto tiene cuatro parámetros, xmmin,xmax,ymin,ymax. Esto es debido a que el etiqueta se hizo formando rectángulos, por lo tanto este será su caja delimitadora.

Uno de los problemas que presenta al dataset es que no están correctamente definidos las características de ancho y alto de las imágenes con su correspondiente etiqueta. Este problema nos producía desde un primer momento una carga incorrecta de las cajas delimitadoras que identifican a cada uno de los elementos por lo tanto no definiendo correctamente las máscaras.

Para solucionar este problema se ha realizado un script "ajustarTamanosXML".

Etiquetado con Makesense

Aunque el dataset es apropiado pudimos ampliar las etiquetas con la herramienta *makesense*, pudiendo realizar el etiquetado de más insectos de la imagen que no estaban etiquetadas incluso dándole un etiquetado *poligonal* el cual al tener más puntos este llegará a realizar predicciones más exactas.² Para utilizar la herramienta acudiremos a:

<https://www.makesense.ai/>

²El proyecto está programado para cojer los 4 puntos de las cajas delimitadoras (etiquetado rectangular), en caso de querer realizarse con etiquetado poligonal ha de adecuarse.

5.2. COCO - Common Objects in Context

A la hora de realizar el entrenamiento sobre un modelo no crearemos un modelo desde 0 ya que este nos llevaría mas tiempo y recursos si no que utilizaremos un modelo pre-entrenado con una gran cantidad de datos y con unos pesos ya establecidos previamente.

Para poder descargar este modelo lo haremos a traves del siguiente enlace: [mask r-cnn.h5](#).

Una vez que lo hemos descargado nos tendremos que asegurar que cuenta con la extion .h5 la cual es la que utilizan todos los modelos y nos indicará que es un archivo que utiliza una gran cantidad de datos.

5.3. Objetos

Atributos A lo largo de el entrenamiento y identificación de los insectos tendremos varios atributos que hacen clave la ejecución del programa, estos son los siguientes:

- **class_ids.** Este es un array que se utiliza para guardar los identificadores de cada uno de los insectos detectados. Guardara n elementos comprendidos entre el 1 y el 3 siendo estos:
 1. Mosca Blanca. WF
 2. Nesidicoris. NS
 3. Macrolophus. MC
- **bbox - Bounding Boxes.** Correspondiente a las esquinas de las cajas delimitadoras siendo 2 puntos en el *plano(x,y)*:
 - xmin. Vertice inferior izquierda
 - xmax. Vertice superior derecha
 - ymin. Vertice inferior izquierda
 - ymax. Vertice superior derecha
- **image_id** Correspondiente a el identificador de la imagen, tendrá valores posibles del 1 al n (n = número de imágenes).
- **mask.**Corresponde al array de las máscaras de los objetos habiendo tantas máscaras como insectos,estas en la iteración 0 será todo 0 luego irán rellenándose el fondo de la caja delimitadora con valor de 1 al contorno.
- **class_id_counter.** Corresponde el numero de elementos identificados durante la predicción.
- **score.** Este atributo correspondiente a cada objeto siendo la probabilidad de que tan seguro esta nuestro modelo de haber identificado a ese objeto,es un varol en coma flotante del 0 al 1. A continuación mostraremos el ejemplo de dos insectos detectados:



Figura 5.2: Ejemplo de una imagen de entrada a Mask R-CNN

Como podemos ver en la figura 5.2 es una entrada correspondiente al dataset en la que a cada uno de los objetos etiquetados se le asigna la mascara y un color único además de ser representada las máscaras de todos ellos.

Mientras que en la figura 5.3 corresponde a una imagen tras haber hecho predicciones sobre ella



Figura 5.3: Ejemplo de una imagen tras haber realizado predicciones

Para tener una mejor comprensión sobre cada uno de los insectos detectados y sus correspondientes atributos durante el proceso de predicción, se muestra un pequeño diagrama con sus correspondientes atributos a cada uno de ellos.

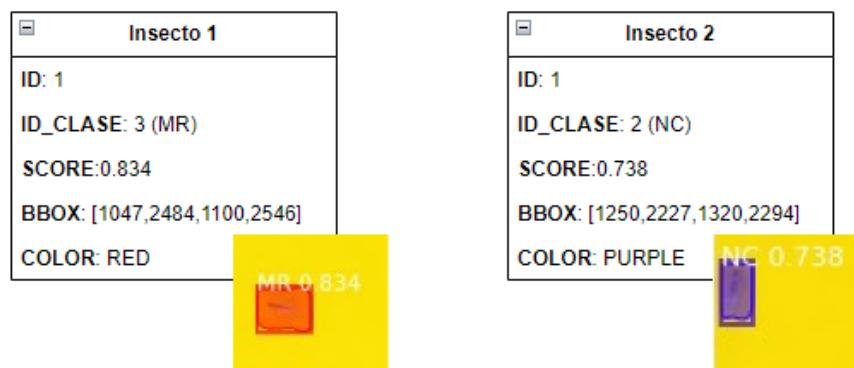


Figura 5.4: Figura correspondiente a los atributos de 2 insectos detectados

5.4. Algoritmos

SGD - Stochastic Gradient Descent

Adam

RMSprop

5.5. Resultados

5.6. Mejoras

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

- [1] 4TU.ResearchData Community. 4tu.researchdata community portal. <https://community.data.4tu.nl/>, 2024. Último acceso: [Fecha de tu último acceso].
- [2] Fritz AI. Exploring activation and loss functions in machine learning. <https://fritz.ai/exploring-activation-and-loss-functions-in-machine-learning/>, 2023. Accedido en 2024.
- [3] Luis Amador Hidalgo. *Inteligencia artificial y sistemas expertos*. Universidad de Córdoba, Servicio de Publicaciones, 1996.
- [4] Appen. Recent developments in neural networks. <https://www.appen.com/blog/recent-developments-neural-networks>, 2023. <https://www.appen.com/blog/recent-developments-neural-networks>.
- [5] Varios Autores. An Introduction to Convolutional Neural Networks. *arXiv:1511.08458*, 2015.
- [6] Baeldung. Epoch in neural networks, 2023. Último acceso: 2024.
- [7] Julio Berbel. La inteligencia artificial en la agricultura: perspectivas de los sistemas expertos. 1989.
- [8] Jason Brownlee. Understand the dynamics of learning rate on deep learning neural networks. <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>, 2019. Último acceso: 2024.
- [9] Cibebuchi. Optimizing epochs in neural networks: A practical approach to stability and performance. <https://blog.cibebuchi.com/>

- [com/optimizing-epochs-in-neural-networks-a-practical-approach-to-stability-and-performance/](https://www.tensorflow.org/tutorials/optimization/optimizing_epochs), 2023. Último acceso: 2024.
- [10] DataCamp. Introduction to convolutional neural networks (cnns), 2021. Último acceso: [Fecha de tu último acceso].
 - [11] Fouriering. Procesamiento digital de imagenes, 2020. Available online: <https://www.fouriering.com/post/procesamiento-digital-de-imagenes> (Accessed on 21 October 2021).
 - [12] Barcelona Geeks. Búsqueda selectiva para la detección de objetos: R-cnn. <https://barcelonageeks.com/busqueda-selectiva-para-la-deteccion-de-objetos-r-cnn/>, 2023.
 - [13] Ross Girshick. Fast r-cnn. <https://paperswithcode.com/method/fast-r-cnn>, 2015. Acceso: 9 de Abril de 2024.
 - [14] Ross Girshick. Fast r-cnn. <https://paperswithcode.com/method/fast-r-cnn>, 2015. Acceso: 9 de Abril de 2024.
 - [15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2014.
 - [16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017. Accessed: [Insert Date Here].
 - [17] Christian Heipke. *Crowdsourcing of Geospatial Data*, volume XLI-B6, pages 83–90. Springer International Publishing, Cham, 2016.
 - [18] IBM. Neural networks. <https://www.ibm.com/es-es/topics/neural-networks>, 2023. Último acceso en 2024.
 - [19] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
 - [20] LearnOpenCV. Mask r-cnn. <https://learnopencv.com/tag/mask-r-cnn-2/>, 2022. Último acceso: [Fecha de acceso].
 - [21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2016.

- [22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. <https://paperswithcode.com/method/rpn>, 2015.
- [23] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3302–3312, 2019.
- [24] Prince Kumar Singh. Pytorch: Tensors and its operations. <https://www.analyticsvidhya.com/blog/2023/03/pytorch-tensors-and-its-operations/>, March 2023.
- [25] Ljubiša Stanković and Danilo Mandic. Convolutional neural networks demystified: A matched filtering perspective-based tutorial. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.
- [26] Kazheen Ismael Taher and Adnan Mohsin Abdulazeez. Deep learning convolutional neural network for speech recognition: a review. *International Journal of Science and Business*, 5(3):1–14, 2021.
- [27] Towards Data Science. A comprehensive guide to convolutional neural networks — the eli5 way. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a5>, 2018. Último acceso: 9 de Abril de 2024.
- [28] User:X93ma. User contributions - statwiki, 2022. [Online; accessed 7-April-2024].
- [29] Analytics Vidhya. Understanding loss function in deep learning. *Analytics Vidhya Blog*, June 2022.
- [30] Mrinal Walia. Semantic segmentation vs. instance segmentation: Explained. <https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/>, 2022. Accessed: 2024-04-07.
- [31] Xiaoping Wu, Chi Zhan, Yukun Lai, Ming-Ming Cheng, and Jufeng Yang. Ip102: A large-scale benchmark dataset for insect pest recognition. In *IEEE CVPR*, pages 8787–8796, 2019.
- [32] Ce Zheng, Xiaolin Xie, Longtao Huang, Binyao Chen, Jianling Yang, Jiewei Lu, Tong Qiao, Zhun Fan, and Mingzhi Zhang. Detecting glaucoma based on spectral domain optical coherence tomography

imaging of peripapillary retinal nerve fiber layer: a comparison study between hand-crafted features and deep learning model. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 258:577–585, 2020.