



UNIVERSIDAD DE GRANADA

Facultad de Ciencias y Escuela Técnica Superior de
Ingenierías Informática y de Telecomunicación (ETSIIT)

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y
MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Detección y conteo de baobab en imágenes de Google Earth usando Deep Learning

Presentado por:

Cristina de la Torre Villaverde

Tutores:

Siham Tabik

Departamento de Ciencias de la Computación e Inteligencia Artificial

Emilio Guirado

Instituto Multidisciplinar para el Estudio del Medio “Ramon

Margalef”, Universidad de Alicante, San Vicente del Raspeig, Spain

Curso académico 2023-2024

Detección y conteo de baobab en imágenes de Google Earth usando Deep Learning

Cristina de la Torre Villaverde

Cristina de la Torre Villaverde *Detección y conteo de baobab en imágenes de Google Earth usando Deep Learning.*

Trabajo de fin de Grado. Curso académico 2023-2024.

Responsable de tutorización	Siham Tabik <i>Departamento de Ciencias de la Computación e Inteligencia Artificial</i>	Doble grado en Ingeniería Informática y Matemáticas
	Emilio Guirado <i>Instituto Multidisciplinar para el Estudio del Medio “Ramon Margalef”, Universidad de Alicante, San Vicente del Raspeig, Spain</i>	Facultad de Ciencias y Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación (ETSIIT)
		Universidad de Granada

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Cristina de la Torre Villaverde

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 18 de junio de 2024

Fdo: Cristina de la Torre Villaverde

Índice general

Agradecimientos	IX
Resumen	XI
Summary	XIII
1 Introducción	1
1.1 Motivación	1
1.2 Objetivos	6
2 Fundamentos matemáticos	7
2.1 Álgebra lineal	7
2.1.1 Matrices. Operaciones y propiedades	8
2.1.2 Valores y vectores propios de las matrices	9
2.2 Análisis matemático	11
2.2.1 Conceptos básicos	11
2.2.2 Conceptos de diferenciabilidad	13
2.2.3 Descenso del gradiente	17
2.3 Probabilidad	20
2.3.1 Probabilidad condicionada	21
2.3.2 Teorema de Bayes	21
3 Fundamentos informáticos	23
3.1 Aprendizaje automático	23
3.2 Redes neuronales	24
3.2.1 Estructura	24
3.2.2 Modelos de redes neuronales	27
3.3 Aprendizaje profundo	31
3.4 Redes convolucionales	31
3.4.1 Capa convolucional	32

Índice general

3.4.2	Capa Pooling	33
3.4.3	Capas Fully-Connected	34
3.5	Algoritmos de detección en una etapa (One stage)	35
3.5.1	RetinaNet	35
3.5.2	YOLO (You Only Look Once)	37
3.6	Algoritmos de detección en dos etapas (Two stages)	38
3.6.1	R-CNN	38
3.6.2	Fast R-CNN	39
3.6.3	Faster R-CNN	39
3.6.4	Mask R-CNN	40
4	Diseño del proyecto	41
4.1	Workflow del proyecto y planificación temporal	41
4.2	Detron2 para la detección en imágenes	42
4.2.1	Estructura de la red	43
4.2.2	Métricas de evaluación	45
5	Análisis experimental	47
5.1	Preparación de los datos (Pre-procesado)	47
5.2	Creación y entrenamiento del modelo	48
5.3	Testeo del modelo	50
5.4	Comparativa de los resultados	57
6	Conclusiones y trabajo futuro	61
7	Manual de usuario	63
	Bibliografía	65

Agradecimientos

A mis tutores Siham y Emilio, por todo lo enseñado.

A mis padres, por ser apoyo y soporte.

A mi hermano, por ser inspiración.

A mis amigas, por ser compañía y confianza.

A Jesús, por ser compañero de vida.

A Johanna, por los años de carrera y todos los que nos quedan.

Resumen

Actualmente, nos enfrentamos a un problema global: el cambio climático y, como consecuencia, la desaparición de especies. Para intentar frenar este fenómeno, se están desarrollando numerosas herramientas basadas en la inteligencia artificial, y más concretamente en el aprendizaje automático, permitiendo así monitorear las poblaciones y enfocar la atención en aquellas que muestran una disminución en el número de individuos.

Un ejemplo de un árbol que está sufriendo este fenómeno es el Baobab, una especie ancestral con increíbles cualidades de supervivencia que proporciona refugio a numerosas especies. Sin embargo, debido al aumento de las temperaturas, la extrema escasez de agua y la deforestación, los baobabs están experimentando una notable disminución de individuos. Esta tendencia debe ser frenada ya que la desaparición de esta especie implicaría el declive de muchas otras, así como la pérdida de una importante fuente de consumo de carbono.

El objetivo de este proyecto es desarrollar un modelo robusto para la detección de baobabs en imágenes satelitales. Para ello, primero construimos una base de datos compuesta por 464 imágenes de Google Earth de zonas donde habitan los baobabs. Estas imágenes serán etiquetadas y usadas como conjunto de entrenamiento y prueba para nuestro modelo. Existen numerosos proyectos de investigación dedicados a la detección de objetos, desde propósitos más generales hasta otros centrados en conjuntos de datos específicos como aviones y coches, o incluso algunos dedicados a la detección de olivos o cocoteros. Sin embargo, no existen proyectos enfocados en la detección de árboles ancestrales, y más específicamente, en los baobabs.

El principal desafío de este proyecto es la falta de una estructura distintiva en la vista aérea de los baobabs, ya que son árboles con un tronco muy alto y una copa relativamente pequeña. Esta característica complica su detección desde el aire. Sin embargo, observamos que en las imágenes satelitales se

Resumen

puede ver la sombra de estos árboles, la cual es muy característica y puede ayudar a identificar visualmente si se trata de un baobab. Por lo tanto, decidimos entrenar nuestro modelo de detección de dos formas: utilizando solo los datos del baobab y utilizando los datos del baobab junto con su sombra, para determinar de qué manera nuestro modelo obtiene mejores resultados.

Utilizando inteligencia artificial y aprendizaje automático, podemos monitorear de manera más eficiente las poblaciones de baobabs y actuar rápidamente en áreas donde se observe una disminución de individuos. A pesar de los desafíos técnicos, como la identificación desde vistas aéreas, la inclusión de las sombras de los baobabs en el modelo de detección muestra un enfoque innovador y prometedor. La implementación exitosa de este proyecto podría servir de modelo para la conservación de otras especies en riesgo, contribuyendo así a la lucha global contra la pérdida de biodiversidad en un mundo cada vez más seco.

Summary

We are currently facing a global problem: climate change and, as a consequence, the disappearance of species. In an attempt to curb this phenomenon, many tools based on artificial intelligence and, more specifically, on automatic learning are being developed, making it possible to monitor populations and focus attention on those showing a decline in numbers.

An example of a tree affected by this phenomenon is the baobab, an ancient species with incredible survival qualities that provides shelter to many species. However, due to rising temperatures, extreme water shortages and deforestation, baobabs are experiencing a significant decline in numbers. This trend must be halted, as the disappearance of this species would mean the decline of many others, as well as the loss of an important carbon sink.

The aim of this project is to develop a robust model for detecting baobabs in satellite imagery. We will use a database of 464 Google Earth images of areas where baobabs occur. These images will be tagged and used as a training and test set for our model. There are numerous research projects dedicated to object detection, from more general purposes to others focused on specific datasets such as aircraft and cars, or even some dedicated to detecting olive or coconut trees. However, there are no projects dedicated to the detection of ancient trees and, more specifically, baobabs.

The main challenge of this project is the lack of a distinctive structure in the aerial view of baobabs, as they are trees with a very tall trunk and a relatively small crown. This feature makes them difficult to detect from the air. However, we observed that in satellite images, the shadow of these trees can be seen, which is very characteristic and can help to visually identify if it is a baobab. We therefore decided to train our detection model in two ways: using only the baobab data and using the baobab data together with its shadow, to determine which way our model performs best.

Using artificial intelligence and machine learning, we can monitor baobab

Summary

populations more efficiently and act quickly in areas where declines are observed. Despite technical challenges, such as identification from aerial images, the inclusion of baobab shadows in the detection model is an innovative and promising approach. Successful implementation of this project could serve as a model for the conservation of other endangered species, contributing to the global fight against biodiversity loss in an increasingly arid world.

1 Introducción

1.1. Motivación

Gracias a los avances científicos y a las investigaciones sabemos que los árboles ancestrales son un pilar fundamental en la naturaleza, ya que entre otras cosas en ellos crecen otras plantas y viven animales, por lo que, si ellos desaparecen, desaparecería el hogar y hábitat de aquellos que viven en ellos sucediéndose así una gran cadena de pérdida de especies. También son un sumidero de carbono, función que también hacen los monocultivos de gran escala, pero con una capacidad mucho menor. Por toda esta información que se ha ido descubriendo a lo largo de los años hoy en día somos más conscientes de la importancia de preservar estos ejemplares ancestrales. En este trabajo de investigación vamos a explorar una herramienta que puede ser muy útil para la detección de árboles desde el espacio, especialmente nos vamos a centrar en los Baobabs, los árboles más longevos y grandes de África. Esta especie ofrece sombra, cobijo, alimento, medicinas y textiles a la población africana [1].

El baobab africano *Adansonia digitata* es un símbolo de longevidad y resistencia en los paisajes africanos, ya que es la planta con flores viva más antigua conocida hasta el momento. Pueden contener hasta 500 metros cúbicos de madera y pueden superar los 2000 años, esto ha desconcertado a los científicos durante décadas, pero recientes estudios están dando nuevos datos sobre los secretos de su longevidad.



1 Introducción

Uno de los hallazgos más fascinantes es el descubrimiento de que los baobabs no solo producen nuevas ramas, como la mayoría de los árboles, sino que periódicamente generan nuevos tallos. Esta estrategia de crecimiento, aparentemente única en el reino vegetal, les permite conservar energía al producir estructuras más pequeñas en lugar de ramas voluminosas, las cuales necesitan más energía para ser producidas. Sin embargo, a pesar de esta adaptación, los baobabs se están enfrentando a un enemigo: el cambio climático. Estos gigantes del continente africano, que han resistido guerras, sequías y el paso de los siglos, ahora se enfrentan a una amenaza que desafía incluso su capacidad de adaptación. Los investigadores han observado un aumento preocupante en la mortalidad de los baobabs en los últimos años, sin una causa evidente. Aunque inicialmente se sospechaba de enfermedades o epidemias, las investigaciones han revelado que el cambio climático podría estar detrás de estas misteriosas muertes.

El aumento de las temperaturas, la escasez de agua y la intensificación de fenómenos climáticos extremos están poniendo a prueba la resistencia de estos árboles ancestrales. El estrés hídrico y la desertificación están debilitando su sistema inmunológico, haciéndolos más vulnerables a enfermedades y plagas. Además, la pérdida de hábitat y la fragmentación del paisaje están exacerbando aún más su situación.

Para abordar esta crisis, se requiere mucha investigación y es crucial llevar a cabo un monitoreo de la salud de los baobabs y comprender mejor los factores que contribuyen a su desaparición. Es en este punto donde nuestro trabajo de investigación cobra importancia.

Utilizando tecnología, como imágenes de Google Earth y algoritmos de inteligencia artificial, nos proponemos identificar y mapear la ubicación de los baobabs [2].



Si echamos la vista atrás hay un largo recorrido en cuanto a investigaciones sobre detección de objetos con redes convolucionales en imágenes satélite, en muchos de ellos, el propósito ha sido general, es decir, comprobar como de buenos son los resultados al aplicar diferentes modelos de detección de objetos en la búsqueda de objetos en imágenes satélite. Otros estudios se han centrado en comparar distintos modelos sobre un mismo conjunto de datos para así poder decidir con cual se obtenían mejores resultados, y también encontramos investigaciones que han buscado la mejora de resultados a través de convertir estas imágenes satélite en imágenes con mayor calidad [3–9].

Acotando un poco más los datos de entrenamiento, encontramos proyectos que se han centrado en la búsqueda de objetos concretos, como aviones o coches [10, 11].

Centrándonos más en proyectos similares al que vamos a desarrollar encontramos algunos en los que se ha aplicado la detección de objetos sobre árboles como el cocotero o la palma de aceite árboles los cuales tienen una estructura muy característica, y que en estas imágenes satélite vistos desde arriba su copa tiene una forma muy definida, además suelen encontrarse agrupados en poblaciones muy densas de la misma especie. También encontramos estudios que han buscado encontrar árboles como el Olivo, estos árboles no tienen una estructura tan característica, pero sí que las plantaciones de dichos árboles suelen seguir una distribución geométrica con una distancia uniforme de separación entre cada árbol y el suelo que los rodea tiene una textura uniforme [12–15].

A pesar de la gran cantidad de estudios que hay relacionados con el tema no se encuentran investigaciones sobre la detección de árboles ancestrales y más concretamente de los Baobabs.

Nosotros nos vamos a enfocar en la especie *Adansonia grandiflora*, cuyas poblaciones están experimentando una preocupante disminución.

1 Introducción



Figura 1.1: Avenida de los Baobabs en Madagascar

Estos árboles tienen una estructura muy característica, como podemos observar en la imagen, tienen un tronco muy alto y una copa relativamente pequeña en proporción con el tamaño del Baobab. Si observamos estos árboles desde una vista aérea la imagen que tendríamos sería como las que vemos en las siguientes fotografías:



(a) Periodo de floración

(b) Periodo de abscisión

Figura 1.2: Baobab en periodo de floración (con hojas) (a) y en periodo de abscisión (sin hojas) (b)

Podemos ver que estas imágenes no nos aportan nada que a simple vista nos haga detectar si estos árboles son Baobabs u otro tipo de árbol, pero

hemos observado que la proyección de su propia sombra puede aportarnos muchas herramientas para hacer esta detección. Varias veces al día, según la posición del sol, podemos obtener una proyección perfecta de la forma de estos árboles, haciendo muy fácil la identificación de estos árboles, como podemos ver a continuación.



Figura 1.3: Avd de los Baobabs, se aprecia la proyección de su sombra.

Por ello, en este proyecto, vamos a aplicar un algoritmo de detección de objetos, aportándole los datos en el conjunto de entrenamiento con y sin sombra y vamos a observar si realmente, al darle la información sobre la sombra obtenemos mejores resultados y podemos detectar una mayor cantidad de Baobabs. Esta técnica de utilizar la sombra que se observa en imágenes satélites ha sido utilizada en algunos proyectos con el fin de, con ayuda de esta característica, estimar la altura de los edificios, pero sin embargo no hemos encontrado proyectos en los que se utilice para enriquecer el conjunto de datos de entrenamiento [16].

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar un modelo robusto de detección de Baobabs en imágenes de satélite de Google Earth. Para conseguirlo debemos abordar los siguientes objetivos particulares:

1. Estudiar los fundamentos matemáticos del aprendizaje de las redes neuronales convolucionales y de los modelos de detección de objetos en imágenes.
2. Crear una base de datos de calidad que nos ayude a entrenar nuestro modelo.
3. Una vez creado el modelo, analizar dos estrategias de supervisión, una de ellas consistirá en supervisar el entrenamiento con un "bounding-box" que delimita únicamente el Baobab, y una segunda, en la que el "bounding-box" delimita el Baobab junto con su sombra, para decidir cuál de ellas hace nuestro modelo más preciso.

2 Fundamentos matemáticos

La matemática es la base de todas las ciencias, y como preludio al resto de conceptos que vamos a explicar en el resto de capítulos, en este, vamos a presentar algunas definiciones matemáticas que están detrás de los algoritmos y modelos informáticos que se describen en este proyecto.

2.1. Álgebra lineal

El álgebra lineal es una rama de las matemáticas que se enfoca en el estudio de vectores, espacios vectoriales, transformaciones lineales y sistemas de ecuaciones lineales. Su importancia en el aprendizaje automático radica en su capacidad para modelar y resolver problemas con múltiples variables y relaciones lineales. En el contexto de los modelos de detección de objetos, el álgebra lineal desempeña un papel crucial en varios aspectos [17]:

Representación de características: En los modelos de detección de objetos, las imágenes se suelen representar mediante características numéricas. El álgebra lineal ofrece herramientas para manipular y transformar eficientemente estas características.

Modelado de relaciones espaciales: La detección de objetos en imágenes requiere comprender las relaciones espaciales entre los puntos de la imagen. El álgebra lineal se emplea para dar forma a estas relaciones mediante transformaciones geométricas como traslaciones, rotaciones y escalas. Esto facilita la construcción de modelos que sean invariables ante la posición, orientación y tamaño de los objetos en la imagen.

Optimización de modelos: Los modelos de detección de objetos suelen plantearse como problemas de optimización, donde se busca ajustar los parámetros del modelo para maximizar su rendimiento. El álgebra lineal

proporciona técnicas eficientes para resolver estos problemas, como el descenso de gradiente y la descomposición de matrices.

2.1.1. Matrices. Operaciones y propiedades

Vamos a definir las matrices y algunas de sus operaciones y propiedades, ya que como veremos más adelante, son la base de las redes neuronales [18].

Definición 2.1 Una matriz de m filas y n columnas es una caja de nm números a_{ij} , donde a_{ij} está en la fila i -ésima y columna j -ésima. Notaremos $A = (a_{ij})$. Dos matrices A y B con el mismo número de filas y de columnas, son iguales si $a_{ij} = b_{ij}$, para cada i, j . Al conjunto de matrices $m \times n$ se denotará por $M_{m \times n}(\mathbb{R})$.

Definición 2.2 Una matriz cuadrada $A \in M_{n \times n}(\mathbb{R})$ se dice que es regular (o invertible o no degenerada) si existe $B \in M_{n \times n}(\mathbb{R})$ tal que $AB = I_n$. A la matriz B la denotamos por A^{-1} y se llama la matriz inversa de A .

Dadas dos matrices $A, B \in M_{m \times n}(\mathbb{R})$ y $\lambda \in \mathbb{R}$, se define la suma de A y B y el producto de λ por A como

$$A + B = (a_{ij} + b_{ij}), \quad \lambda A = (\lambda a_{ij}).$$

Definición 2.3 Dadas dos matrices $A \in M_{m \times n}(\mathbb{R})$ y $B \in M_{n \times r}(\mathbb{R})$ se define el producto de A por B como la matriz $AB \in M_{m \times r}(\mathbb{R})$ definida por

$$(AB)_{ij} = \sum_{k=1}^n a_{ik}b_{kj}.$$

Proposición 2.4 Se tiene las siguientes propiedades del producto de matrices:

1. $A(B + C) = AB + AC, (A + B)C = AC + BC.$

$$2. (\lambda A)B = \lambda(AB) = A(\lambda B).$$

$$3. A(BC) = (AB)C.$$

Proposición 2.5 Denotamos $Gl(n, \mathbb{R})$ el conjunto de las matrices regulares de orden n . Se tiene las siguientes propiedades:

1. La matriz identidad es regular y su inversa es ella misma.
2. Si A y B son regulares, entonces AB es regular y su inversa es $B^{-1}A^{-1}$.
3. Si A es regular, A^{-1} también es regular y su inversa es A .
4. Si A es regular y $\lambda \neq 0$, entonces λA es regular y su inversa es $\frac{1}{\lambda}A^{-1}$.

2.1.2. Valores y vectores propios de las matrices

Como veremos más adelante, algunos algoritmos de los descritos utilizan el método de reducción de la dimensionalidad de imágenes, por lo tanto, es necesario describir el concepto de valores y vectores propios de una matriz [19].

Definición 2.6 Dado un endomorfismo f de un espacio vectorial V , decimos que $\lambda \in \mathbb{R}$ es un valor propio (o autovalor) de f si existe $v \neq 0$ tal que $f(v) = \lambda v$. Al vector v lo llamamos un vector propio (o autovector) de λ .

Si $A \in M_n(\mathbb{R})$, decimos que $\lambda \in \mathbb{R}$ es un valor propio (o autovalor) de A si existe $x \in \mathbb{R}^n, x \neq 0$ tal que $Ax = \lambda x$. Al vector x lo llamamos un vector propio (o autovector) de λ .

Proposición 2.7 Con la notación anterior, tenemos:

1. $f(V_\lambda) \subset V_\lambda$.
2. V_λ es un subespacio propio.
3. λ es valor propio si y sólo si $\dim(V_\lambda) \geq 1$. En tal caso, decimos que V_λ es el subespacio propio del valor λ .
4. $V_\lambda = \text{Ker}(f - \lambda I_V)$.
5. $\dim(V_\lambda) = n - \text{rango}(f - \lambda I_V)$.

6. Si λ es valor propio, entonces $\det(f - \lambda I_V) = 0$.
7. El número de valores propios es menor o igual que la dimensión de V .
8. $\text{Ker}(f) = V_0$ y $\lambda = 0$ es un valor propio si y sólo si la nulidad es al menos 1.

Estas propiedades se extienden inmediatamente para matrices

Definición 2.8 Si f es un endomorfismo de un espacio vectorial, se llama el polinomio característico de f a:

$$P_f(\lambda) = \|f - \lambda I\|$$

A la ecuación $P_f(\lambda) = 0$ se llama ecuación característica de f .

Si A es una matriz cuadrada, el polinomio característico es:

$$P_A(\lambda) = \|A - \lambda I\|$$

Observemos que:

1. $P_f(\lambda)$ es un polinomio de orden n .
2. λ es un valor propio si y solo si es una raíz de $P_f(\lambda)$.
3. Si f es diagonalizable, entonces $P_f(\lambda)$ tiene n raíces (que pueden ser iguales).

2.2. Análisis matemático

El análisis matemático se enfoca en el estudio de conceptos como límites, continuidad, derivadas, integrales y series infinitas, con el objetivo de entender y formalizar el comportamiento de funciones y conjuntos de números reales o complejos. En el contexto de la detección de objetos en imágenes, el análisis matemático desempeña un papel importante en varios aspectos:

Extracción de características: El análisis matemático proporciona técnicas para identificar características relevantes en las imágenes, como bordes, texturas o patrones de color.

Modelado de contextos: En ocasiones, en la detección de objetos es necesario considerar el contexto en el que aparecen los objetos en las imágenes. El análisis matemático ofrece herramientas para modelar y representar este contexto basándose en conceptos de probabilidad y optimización.

Entrenamiento de modelos: Durante el entrenamiento de modelos de detección de objetos, surge el desafío de la optimización. El análisis matemático proporciona técnicas para ajustar los parámetros del modelo de manera óptima.

Evaluación de precisión: El análisis matemático es fundamental para evaluar la precisión y el rendimiento de los modelos de detección de objetos. Métricas como precisión, recall, precisión media promedio (mAP) y la curva de precisión-recall se basan en conceptos de probabilidad y estadística, fundamentos del análisis matemático.

Optimización de la eficiencia computacional: En la implementación de modelos de detección de objetos, la eficiencia computacional es crucial, especialmente en dispositivos con recursos limitados. El análisis matemático puede utilizarse para optimizar la eficiencia computacional de los modelos.

2.2.1. Conceptos básicos

En esta sección vamos a presentar algunos conceptos básicos necesarios para, posteriormente, poder tratar el tema de la diferenciabilidad y finalmente in-

troducir el algoritmo de gradiente descendiente [20].

Definición 2.9 Dados dos vectores $x = (x_1, x_2, \dots, x_n)$, $y = (y_1, y_2, \dots, y_n)$ se define su producto escalar por:

$$\langle x, y \rangle = \sum_{k=1}^n x_k y_k = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

Este producto escalar se llama producto escalar euclídeo.

Observa que el producto escalar de dos vectores no es un vector sino un número real. La notación $x \cdot y$ es frecuentemente usada en los libros de Física para representar el producto escalar de los vectores x e y .

Propiedades del producto escalar.

1. $\langle x, x \rangle > 0$, y $\langle x, x \rangle = 0 \iff x = 0$.
2. Simetría. $\langle x, y \rangle = \langle y, x \rangle$ para todos $x, y \in \mathbb{R}^n$.
3. Linealidad. $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$ para todos $\alpha, \beta \in \mathbb{R}$ y para todos $x, y, z \in \mathbb{R}^n$.

Dichas propiedades del producto escalar se deducen fácilmente de su definición.

Definición 2.10 La norma euclídea de un vector $x = (x_1, x_2, \dots, x_n)$ se define por

$$\|x\|_2 = \sqrt{\langle x, x \rangle} = \sqrt{\sum_{k=1}^n x_k^2}$$

Propiedades de la norma euclídea.

1. $\|x\|_2 > 0$, y $\|x\|_2 = 0 \iff x = 0$.
2. Homogeneidad. $\|\alpha x\|_2 = |\alpha| \|x\|_2$ para todo $x \in \mathbb{R}^n$ y todo $\alpha \in \mathbb{R}$.
3. Desigualdad triangular. Para todos $x, y \in \mathbb{R}^n$ se verifica que

$$\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$$

Además, supuesto que x e y no son nulos, la igualdad $\|x + y\|_2 = \|x\|_2 + \|y\|_2$

equivale a que hay un número $\alpha > 0$ tal que $x = \alpha y$ (es decir, los vectores x e y están en una misma semirrecta que pasa por el origen).

Desigualdad de Cauchy-Schwarz.

Para todos $x, y \in \mathbb{R}^n$ se verifica que

$$|\langle x, y \rangle| \leq \|x\|_2 \|y\|_2$$

Además, supuesto que x e y no son nulos, la igualdad $|\langle x, y \rangle| = \|x\|_2 \|y\|_2$ equivale a que existe un número $\alpha \in \mathbb{R}$ tal que $x = \alpha y$ (es decir, los vectores x e y están en una misma recta que pasa por el origen).

Las definiciones anteriormente vistas para \mathbb{R}^n puede extenderse para cualquier espacio vectorial real.

Definición 2.11 Sea X un espacio vectorial real. Una norma sobre X es una aplicación $\|\cdot\| : X \rightarrow \mathbb{R}_{\geq 0}$ que verifica las siguientes propiedades:

1. $\|x\| = 0 \Leftrightarrow x = 0$.
2. Homogeneidad: $\|\alpha x\| = |\alpha| \|x\|$ para todo $\alpha \in \mathbb{R}$ y todo $x \in X$.
3. Desigualdad triangular: $\|x + y\| \leq \|x\| + \|y\|$ para todos $x, y \in X$.

El par ordenado $(X, \|\cdot\|)$ se llama un espacio normado.

2.2.2. Conceptos de diferenciabilidad

En esta sección vamos a definir conceptos muy importantes en el entrenamiento de redes neuronales, ya que los algoritmos de optimización se basan en el uso de conceptos como el vector gradiente y la regla de la cadena.

Definición 2.12 Toda aplicación lineal de un espacio normado de dimensión finita en otro espacio normado es continua

Reciben el nombre de campos escalares las funciones definidas en subconjuntos de \mathbb{R}^n que toman valores en \mathbb{R} . Un campo escalar es, por tanto, una función real que depende de n variables. A partir de ahora, notaremos como campo escalar a las funciones $f : O \subset \mathbb{R}^n \rightarrow \mathbb{R}$ y el concepto de direcciones

se referirá a los vectores unitarios del espacio normado $(\mathbb{R}^n, \|\cdot\|)$.

Definición 2.13 Sea f un campo escalar definido en un conjunto abierto $\Omega \subset \mathbb{R}^n$, sea $a \in \Omega$ y u una dirección. Se define la derivada de f en a en la dirección u como el límite:

$$D_u f(a) = \lim_{t \rightarrow 0} \frac{f(a + tu) - f(a)}{t}$$

supuesto que dicho límite exista.

La derivada direccional de un campo escalar f en un punto a en la dirección del vector e_k de la base canónica, se llama derivada parcial (de primer orden) de f en a respecto a la variable k -ésima. Está definida por:

$$D_{e_k} f(a) = \lim_{t \rightarrow 0} \frac{f(a + te_k) - f(a)}{t}$$

y se representa con los símbolos $D_k f(a)$ y $\frac{\partial f}{\partial x_k}(a)$.

Definición 2.14 Cuando el campo f es parcialmente derivable en a , el gradiente de f en a es, por definición, el vector $\nabla f(a) \in \mathbb{R}^N$ dado por

$$\nabla f(a) = \left(\frac{\partial f}{\partial x_1}(a), \frac{\partial f}{\partial x_2}(a), \dots, \frac{\partial f}{\partial x_N}(a) \right) = \sum_{k=1}^N \frac{\partial f}{\partial x_k}(a) e_k$$

de modo que, para cada $k \in \{1, 2, \dots, N\}$, la k -ésima componente del vector gradiente de f en a es la derivada parcial de f en a , con respecto a la k -ésima variable [21].

Definición 2.15 Sea f un campo escalar definido en un abierto $\Omega \subset \mathbb{R}^n$ y sea $a \in \Omega$. Supongamos que está definido el vector gradiente $\nabla f(a)$. Se dice que f es diferenciable o derivable en a si se verifica que

$$\lim_{x \rightarrow a} \frac{f(x) - f(a) - \langle \nabla f(a) | x - a \rangle}{\|x - a\|} = 0$$

Definamos

$$R(x, a) = \frac{f(x) - f(a) - \langle \nabla f(a), x - a \rangle}{\|x - a\|}$$

La igualdad anterior dice que

$$\lim_{x \rightarrow a} R(x, a) = 0$$

Con lo que, otra forma equivalente de escribir la igualdad es la siguiente:

$$f(x) = f(a) + \langle \nabla f(a), x - a \rangle + R(x, a) \|x - a\|$$

donde

$$\lim_{x \rightarrow a} R(x, a) = 0$$

Proposición 2.16 Sea f un campo escalar diferenciable en un punto a y sea u una dirección en \mathbb{R}^n . Entonces se verifica que

$$D_u f(a) = \langle \nabla f(a), u \rangle$$

Demostración: En la igualdad (2.1) pongamos $x = a + tu$ con lo que obtenemos

$$f(a + tu) = f(a) + \langle \nabla f(a), tu \rangle + R(a + tu, a) \|tu\|$$

$$= f(a) + t \langle \nabla f(a), u \rangle + R(a + tu, a) |t|$$

Deducimos que

$$\lim_{t \rightarrow 0} \frac{f(a + tu) - f(a)}{t} = \langle \nabla f(a), u \rangle + \lim_{t \rightarrow 0} \frac{R(a + tu, a) |t|}{t} = \langle \nabla f(a), u \rangle$$

□

Para poder definir la regla de la cadena necesitamos extender las definiciones anteriores a funciones que tomen valores en espacios vectoriales reales con dimensión mayor que uno y a partir de ahora cuando nos refiramos a campos vectoriales serán funciones de la forma $f : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$.

Definición 2.17 Sea $F : O \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ un campo vectorial definido en un conjunto abierto O . Se dice que F es diferenciable en $a \in O$ si existe una

aplicación lineal $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ tal que:

$$\lim_{x \rightarrow a} \frac{F(x) - F(a) - T(x - a)}{\|x - a\|} = 0$$

Dicha aplicación lineal, si existe, es única; se llama *diferencial* o *derivada* de F en el punto a y se representa por $DF(a)$.

Proposición 2.18 Sea $F = (f_1, f_2, \dots, f_m) : \Omega \rightarrow \mathbb{R}^m$, donde $\Omega \subset \mathbb{R}^n$ es un abierto, una función vectorial de n variables y m componentes, y sea $a \in \Omega$. Equivalen las siguientes afirmaciones:

- a) F es diferenciable en a .
- b) Los campos escalares f_1, f_2, \dots, f_m componentes de F son diferenciables en a .

Supuesto que se verifican a) y b), la matriz de la aplicación lineal $DF(a)$ en las bases canónicas de \mathbb{R}^n y \mathbb{R}^m es la matriz cuyas filas son los vectores gradiente $\nabla f_i(a)$, esto es, la matriz de m filas y n columnas

$$(D_j f_i(a))_{1 \leq i \leq m, 1 \leq j \leq n}$$

Dicha matriz se llama matriz jacobiana de F en a y se representará por $JF(a)$.

Teorema 2.19 (Regla de la cadena). Sean $G : A \rightarrow \mathbb{R}^q$ y $F : B \rightarrow \mathbb{R}^m$, donde $A \subseteq \mathbb{R}^q$ y $B \subseteq \mathbb{R}^n$ son conjuntos abiertos y $G(A) \subseteq B$. Supongamos que G es diferenciable en un punto $a \in A$ y que F es diferenciable en el punto $G(a) \in B$. Entonces se verifica que la función compuesta $H = F \circ G : A \rightarrow \mathbb{R}^m$ es diferenciable en a , y su diferencial viene dada como la composición de las respectivas diferenciales:

$$DH(a) = DF(G(a)) \circ DG(a)$$

Además, si F y G son de clase C^k , entonces H también es de clase C^k .

Observa que $DG(a) : \mathbb{R}^q \rightarrow \mathbb{R}^n$ y $DF(G(a)) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, por lo que la composición es una aplicación lineal de \mathbb{R}^q a \mathbb{R}^m , como debe ser, pues H es una función vectorial de q variables y m componentes.

La expresión de la igualdad por medio de matrices jacobianas es:

$$JH(a) = JF(G(a))JG(a)$$

Poniendo $H = (h_1, h_2, \dots, h_m)$, $F = (f_1, f_2, \dots, f_m)$, $G = (g_1, g_2, \dots, g_n)$; notando las variables por $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, $y = (y_1, y_2, \dots, y_q) \in \mathbb{R}^q$, y escribiendo $b = G(a)$, tenemos que:

$$\left(\frac{\partial h_i}{\partial y_j}(a) \right)_{1 \leq i \leq m, 1 \leq j \leq q} = \left(\frac{\partial f_i}{\partial x_k}(b) \right)_{1 \leq i \leq m, 1 \leq k \leq n} \left(\frac{\partial g_k}{\partial y_j}(a) \right)_{1 \leq k \leq n, 1 \leq j \leq q}$$

2.2.3. Descenso del gradiente

En esta sección vamos a explicar el algoritmo del gradiente descendiente, que ha sido el utilizado en nuestro modelo como algoritmo de optimización.

Corolario 2.20 Sea f un campo escalar diferenciable en un punto a con vector gradiente no nulo en a .

- (a) La dirección en la que la derivada direccional de f en a es máxima es la dirección dada por el gradiente de f en a , es decir, la dirección $\mathbf{u} = \frac{\nabla f(a)}{\|\nabla f(a)\|_2}$.
- (b) La dirección en la que la derivada direccional de f en a es mínima es la dirección opuesta a la dada por el gradiente de f en a , es decir, la dirección $\mathbf{v} = -\frac{\nabla f(a)}{\|\nabla f(a)\|_2}$.

Demostración. Las afirmaciones hechas son consecuencia de la proposición anterior y de la desigualdad de Cauchy-Schwarz, pues para toda dirección \mathbf{w} se tiene que

$$|\langle \nabla f(a), \mathbf{w} \rangle| \leq \|\nabla f(a)\| \|\mathbf{w}\|_2 = \|\nabla f(a)\|_2$$

Y la igualdad se da si, y solo si, hay un número $\lambda \in \mathbb{R}$ tal que $\mathbf{w} = \lambda \nabla f(a)$. Tomando normas en esta igualdad se deduce que $|\lambda| = \frac{1}{\|\nabla f(a)\|_2}$, es decir, las direcciones \mathbf{w} que hacen máximo $|D_{\mathbf{w}} f(a)| = |\langle \nabla f(a), \mathbf{w} \rangle|$ son $\mathbf{u} = \frac{\nabla f(a)}{\|\nabla f(a)\|_2}$ y $\mathbf{v} = -\frac{\nabla f(a)}{\|\nabla f(a)\|_2}$.

Para la primera se tiene que

$$D_{\mathbf{u}}f(a) = \left\langle \nabla f(a), \frac{\nabla f(a)}{\|\nabla f(a)\|_2} \right\rangle = \frac{1}{\|\nabla f(a)\|_2} \langle \nabla f(a), \nabla f(a) \rangle = \|\nabla f(a)\|_2$$

que es el valor máximo que puede tener una derivada direccional.

Análogamente, para la segunda se tiene que

$$D_{\mathbf{v}}f(a) = -\|\nabla f(a)\|_2$$

que es el valor mínimo que puede tener una derivada direccional. \square

El algoritmo del gradiente descendente es un algoritmo de optimización iterativo que busca encontrar el mínimo de una función convexa y diferenciable para un dominio concreto, basando su funcionamiento en la proposición anterior, y en las redes neuronales es utilizado para reducir el error. Es importante seleccionar correctamente la función de coste para el entrenamiento ya que será la que se intentará minimizar.

Definimos un campo escalar diferenciable $J : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}$, que llamaremos función de coste, como

$$J(\theta) = \frac{1}{n} \sum_{k=1}^n F(f(x_k, \theta), y_k) \quad \text{con } n \in \mathbb{N}$$

siendo $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ la función de pérdida entre el valor predicho y el valor real. Por lo tanto, la función de coste es la media de los valores obtenidos de la función de pérdida para cada n . Ahora buscamos encontrar una distribución de parámetros θ que haga que la función J alcance su valor mínimo.

Partimos de un valor aleatorio x , que irá variando. Antes vimos que, considerando un punto de un campo escalar, el valor opuesto del gradiente en dicho punto, es la dirección del máximo decrecimiento, entonces, en cada iteración se calculará el gradiente correspondiente y se actualizarán los valores siguiendo su dirección opuesta, entonces si consideramos $\theta_0 = a$, tenemos:

$$\theta_{n+1} = \theta_n - \alpha \nabla J(\theta_n) \quad \text{con } n \in \mathbb{N} \cup \{0\}$$

, siendo α la tasa de aprendizaje. La elección de la tasa de aprendizaje es

también un factor determinante, ya que un valor muy bajo hará que la convergencia sea demasiado lenta, y uno muy alto hará que el algoritmo diverja, al igual que la elección del punto inicial θ_0 , ya que este tiene una gran influencia en la convergencia del algoritmo.

Con el paso del tiempo han surgido diferentes variantes del algoritmo de gradiente descendiente, pero vamos a centrarnos en el Gradiente descendiente estocástico ya que ha sido el utilizado en nuestro modelo.

Este es uno de los algoritmos más usados y nació como solución al problema del algoritmo de gradiente descendiente en lotes, el tiempo que tardaba en ejecutarse. En este caso, se toma una muestra de los datos del conjunto de entrenamiento y se calcula el gradiente iterando sobre este subconjunto. Así, se reducen los recursos necesarios para la optimización.

Algorithm 1 Descenso Estocástico del Gradiente

```

1: Inicializar los parámetros  $\theta$  aleatoriamente.
2: Definir la tasa de aprendizaje  $\alpha$ .
3: repeat
4:   for cada muestra  $(x_k, y_k)$  en el conjunto de datos do
5:     Calcular la predicción:  $\hat{y}_k = f(x_k, \theta)$ .
6:     Calcular la pérdida:  $F(\hat{y}_k, y_k)$ .
7:     Calcular el gradiente:  $\nabla_{\theta}F(\hat{y}_k, y_k)$ .
8:     Actualizar los parámetros:  $\theta \leftarrow \theta - \alpha \nabla_{\theta}F(\hat{y}_k, y_k)$ .
9:   end for
10:  until la convergencia

```

2.3. Probabilidad

La probabilidad se define como una medida numérica que cuantifica la incertidumbre asociada con eventos aleatorios. En el ámbito de los modelos de detección de objetos, la probabilidad desempeña un papel crucial en varios aspectos:

Modelado de incertidumbre: Los modelos de detección de objetos están diseñados para abordar la incertidumbre inherente a la detección de objetos en imágenes. La probabilidad se emplea para medir esta incertidumbre, proporcionando una medida de confianza o certeza respecto a la presencia y ubicación de un objeto en una escena.

Clasificación de detecciones: En muchas ocasiones, los modelos de detección de objetos generan múltiples detecciones potenciales en una imagen, cada una con una puntuación de confianza asociada. La probabilidad se utiliza para asignar una probabilidad de clase a cada detección, lo que facilita la distinción entre objetos genuinos y posibles falsos positivos.

Fusión de información: Algunos modelos de detección de objetos emplean múltiples fuentes de información, como características visuales, contexto espacial y temporal, y conocimiento previo, para mejorar la precisión de la detección. La probabilidad se utiliza para integrar esta información de manera coherente y combinar las diversas fuentes de evidencia de forma ponderada.

Estimación de la pose y la orientación: En el caso de detectar objetos en imágenes tridimensionales, la probabilidad se emplea para estimar la posición y orientación del objeto en el espacio tridimensional. Esto implica modelar la incertidumbre asociada con la estimación de la pose y utilizar técnicas probabilísticas para lograr una estimación precisa.

A continuación, vamos a presentar dos conceptos de probabilidad estrechamente relacionados con los algoritmos de detección de objetos.

2.3.1. Probabilidad condicionada

La probabilidad condicionada es la probabilidad de que ocurra un evento cuando ya ha ocurrido otro concreto. En la detección de objetos, sirve para obtener la probabilidad de que un objeto sea de una clase específica cuando tiene ciertas características [22].

Definición 2.21 Sea (Ω, \mathcal{A}, P) un espacio probabilístico arbitrario y A un suceso ($A \in \mathcal{A}$) tal que $P(A) > 0$. Para cualquier otro suceso $B \in \mathcal{A}$, se define la probabilidad condicionada de B dado A o probabilidad de B condicionada a A como

$$P(B/A) = \frac{P(B \cap A)}{P(A)}.$$

Observemos que la condición $P(A) > 0$ es necesaria para que la definición tenga sentido. Por otra parte, la idea intuitiva de probabilidad condicionada hace lógica esta restricción ya que si $P(A) = 0$, A es un suceso imposible y no tiene sentido condicionar a él.

2.3.2. Teorema de Bayes

Este teorema es especialmente útil en los algoritmos de detección de objetos ya que se usa en los modelos probabilísticos para actualizar la probabilidad de que haya un objeto en la región seleccionada según las observaciones que va realizando.

Teorema de Bayes o de la probabilidad inversa

Sea (Ω, \mathcal{A}, P) un espacio de probabilidad y sea $\{A_n\}_{n \in \mathbb{N}} \subset \mathcal{A}$ un sistema completo de sucesos o partición de Ω con $P(A_n) > 0, \forall n \in \mathbb{N}$. Sea B un suceso cualquiera de \mathcal{A} , entonces

$$P(A_n/B) = \frac{P(B/A_n)P(A_n)}{\sum_{n \in N} P(B/A_n)P(A_n)}$$

Demostración

Por la definición de probabilidad condicionada y aplicando el Teorema de la probabilidad compuesta, tenemos:

$$P(A_n/B) = \frac{P(B/A_n)P(A_n)}{P(B)}$$

y aplicando el Teorema de la probabilidad total en el denominador se obtiene el resultado deseado.

El razonamiento lógico que subyace en el cálculo de estas probabilidades es el siguiente: Interpretar, de nuevo, el suceso B como el resultado obtenido al realizar un experimento y los sucesos A_n como el conjunto de todas las “causas” que pueden producir la aparición del suceso B ; entonces, si para cada “causa” conocemos su probabilidad a priori $P(A_n)$ y la verosimilitud $P(B/A_n)$ de que el suceso B haya sido causado por A_n , la ocurrencia de B , nos permite asignar, mediante la aplicación del Teorema de Bayes, una “probabilidad a posteriori” $P(A_n/B)$ al suceso de que la verdadera causa haya sido A_n .

3 Fundamentos informáticos

En este capítulo vamos a explicar algunos conceptos informáticos, que es necesario entender, para poder comprender el modelo creado en este proyecto con el fin de obtener un algoritmo que nos ayude a realizar el conteo de Baobabs.

3.1. Aprendizaje automático

Podemos definir el aprendizaje automático como una rama de la informática y la inteligencia artificial que imita el aprendizaje de los humanos haciendo uso de datos y algoritmos. Los algoritmos de aprendizaje automático construyen un modelo basándose en un conjunto de observaciones (datos de entrenamiento) con el objetivo de realizar estimaciones o tomar decisiones sin estar concretamente construidos para ello. Actualmente, podemos encontrar el aprendizaje automático en todo tipo de tareas automatizadas que se utilizan a diario en distintos sectores. Existen varios tipos de aprendizajes automáticos, dependiendo de la clase de problema que buscamos resolver y de los datos que se poseen [23, 24]:

Aprendizaje supervisado: está caracterizado por utilizar un conjunto de datos etiquetados para entrenar los algoritmos, con el objetivo de utilizarlos para asignar nuevos ejemplos. Principalmente, es usado en problemas de clasificación y regresión. En la fase de entrenamiento de modelos de este aprendizaje se va ajustando los pesos de las características en función de los valores de las etiquetas, así la situación deseable será conseguir un modelo capaz de hacer predicciones acertadas sobre instancias vistas por primera vez.

Aprendizaje no supervisado: en este caso, los datos no están etiquetados. Por lo tanto, lo que se hace es analizar patrones y agrupar en conjuntos

los datos no etiquetados.

Aprendizaje semi-supervisado: este aprendizaje combina los dos anteriores, toma un gran conjunto de datos no etiquetados y una pequeña muestra de datos etiquetados.

Aprendizaje por refuerzo: hace uso del método de ensayo y error para entrenar los modelos. Se define un objetivo y unas acciones permitidas, así se van realizando modificaciones en función del estado del sistema.

3.2. Redes neuronales

Las redes neuronales son un modelo computacional inspirado en el funcionamiento y la estructura de las neuronas y el cerebro humano. Se constituyen de conjuntos de nodos, o neuronas, conectados entre sí formando capas. Las redes neuronales son capaces de aprender y reconocer patrones a partir de datos, ya que ajustan sus conexiones ponderadas para realizar tareas complejas. Se han convertido en una herramienta fundamental en el campo de la inteligencia artificial, debido a su estructura y capacidad de adaptación [25–27].

3.2.1. Estructura

La neurona es la unidad básica de procesamiento, esta imita el comportamiento de las neuronas biológicas del cerebro humano. Estas neuronas tienen un estado interno, llamado nivel de activación, el cual puede cambiar cuando la neurona recibe señales.

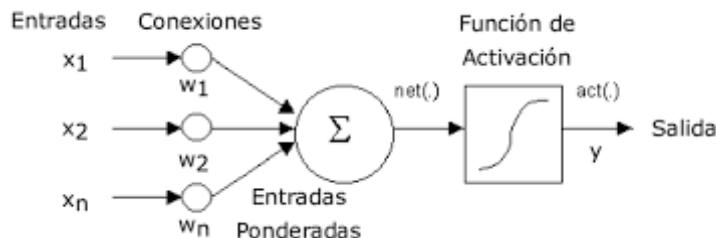


Figura 3.1: Estructura de una neurona artificial

En la imagen podemos ver las diferentes partes que forman la neurona [28, 29]:

Entradas (x_1, x_2, \dots, x_n) entrada recibida por la neurona.

Pesos (w_1, w_2, \dots, w_n) importancia de cada una de las entradas. Así se obtienen las entradas ponderadas, multiplicando cada entrada por su peso correspondiente.

Umbral o bias (w_0) constante que se agrega a las entradas ponderadas.

Sumatorio ponderado (\sum) suma de las entradas ponderadas.

Función de activación ($act()$) esta función añade no linealidad y determina la salida de la neurona evaluando el sumatorio y el bias.

La salida de la neurona artificial sería:

$$y = act(\sum x_i w_i + w_0)$$

Según la función de activación elegida obtendremos mejores o peores resultados. Podemos encontrar algunas funciones no lineales simples como la función escalón o la función signo, pero existen también otras más complejas como la sigmoide, la función ReLU (Rectified Linear Unit) o la tangente hiperbólica, las cuales ofrecen mejores resultados [28, 30].

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \text{ [1]}$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$

Figura 3.2: Tabla con distintos ejemplos de funciones de activación

Las redes neuronales multicapa se forman agrupando neuronas en capas en cascada. En estas capas, las neuronas se encuentran conectadas entre sí,

para procesar la información conforme esta avanza a través de la red. La entrada de una capa es la salida de la anterior y cada capa tiene una función específica dentro de la red. Las redes neuronales están formadas por tres tipos de capas [29, 30]:

Capa de entrada capa que recibe los datos de entrada.

Capas ocultas estas capas procesan y transforman la información extrayendo características complejas de los datos. El número de capas ocultas varían según las necesidades de la red. A mayor número de capas, mayor profundidad y capacidad. En el caso de que la red contenga varias capas ocultas reciben el nombre de Perceptrón multicapa ((Multilayer Perceptron, MLP) [26, 27].

Capa de salida esta capa se encarga de trasladar la información de salida al exterior.

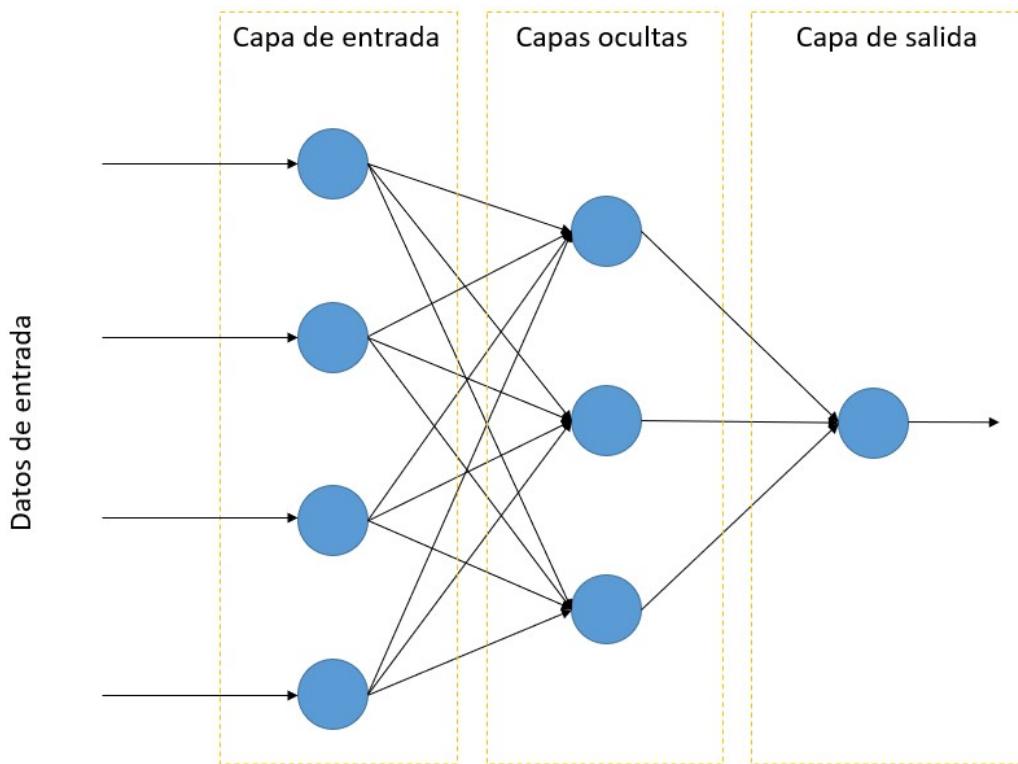


Figura 3.3: Estructura de una red neuronal multicapa

3.2.2. Modelos de redes neuronales

A continuación, vamos a estudiar los modelos de redes neuronales más relevantes, centrándonos más adelante en el modelo de red convolucional que ha sido el utilizado en este proyecto.

3.2.2.1. Redes neuronales Feed-Forward

El modelo de redes neuronales Feed-Forward (feed-forward neural network, FNN), es un modelo básico de red neuronal en el que las neuronas se agrupan formando capas. Están caracterizadas principalmente por tener una única dirección del flujo de información, es decir, la información va desde la capa de entrada a la de salida sin realimentaciones de capas posteriores a capas anteriores.

Se componen de capas de neuronas interconectadas e incluyen una capa de entrada, varias capas ocultas y una capa de salida.

Destacaremos en este tipo de redes el algoritmo de entrenamiento back-propagation o de retropropagación, el cual consiste en ajustar los pesos de las conexiones entre neuronas con el objetivo de minimizar el error entre las salidas predichas por la red y las salidas reales del entrenamiento. En este proceso se propaga el error hacia atrás en cada capa ajustando los pesos, repitiendo este proceso en varias iteraciones, la red ajusta gradualmente los pesos mejorando así sus resultados.

3.2.2.2. Redes neuronales Recurrentes

Las redes neuronales recurrentes (recurrent neural network, RNN) es otra de las arquitecturas básicas de redes neuronales artificiales. Estas, al contrario que las redes Feed-forward, son bidireccionales. Una de sus principales características es que existen conexiones recurrentes, o realimentaciones entre neuronas que hace que la salida de la red se pueda utilizar como entrada en el siguiente paso, formando así un bucle de realimentación [31].

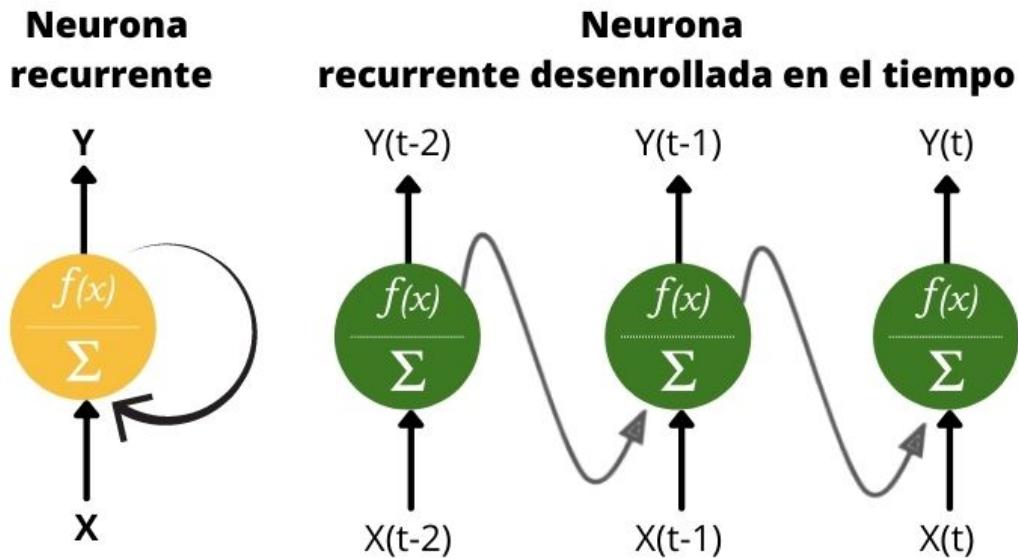


Figura 3.4: Estructura de una red neuronal recurrente

Las realimentaciones pueden darse entre una neurona y ella misma, entre neuronas de una misma capa y entre neuronas de diferentes capas. Gracias a esto, la red neuronal puede tener una memoria interna y ser consciente del tiempo. Por esto, este tipo de redes son especialmente útiles cuando se trabaja con secuencias de datos, ya que la red puede modelar y comprender como cada elemento de la secuencia está relacionado con el resto a lo largo del tiempo [26].

3.2.2.3. Redes neuronales en Base Radial

Las redes neuronales de base radial (Radial Basis Function Networks, RBF) utilizan como función de activación funciones de base radial, estas son funciones matemáticas las cuales sus valores solo dependen de la distancia entre el punto de entrada y un punto fijo (centro) [32, 33].

Están formadas por una capa de entrada, una sola capa oculta y la capa de salida. En la capa oculta, que es la que la distingue del perceptrón multicapa, están las funciones con base radial y así se añaden no linealidades.

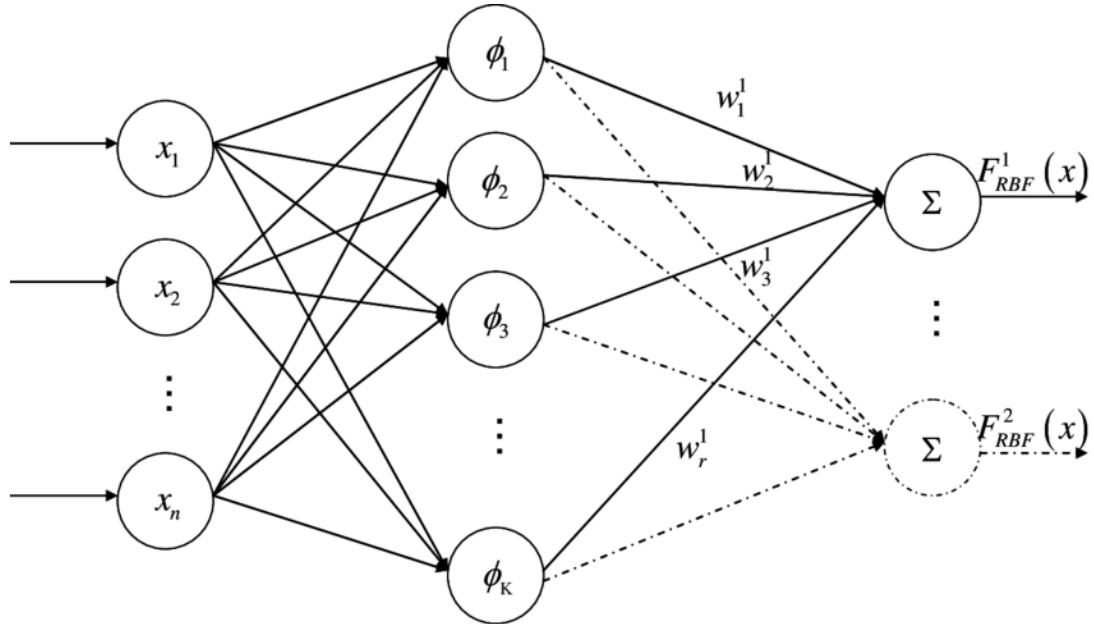


Figura 3.5: Estructura de una red neuronal de base radial

La expresión que define la salida de una red neuronal de base radial con n neuronas de entrada, k neuronas en la capa oculta y r neuronas de salida sería [26]:

$$y_p(t) = \sum_{i=1}^k \omega_{ip} \phi_i(t) + u_p$$

con $p=1 \dots r$.

Donde ω_{ip} son los pesos de cada conexión de cada neurona, u_p el umbral de cada neurona y ϕ la función de activación, estas determinan la activación de las neuronas las cuales se expresan como [26, 34]:

$$\phi_i(t) = \phi(\|X(t) - C_i\| / d_i)$$

Siendo ϕ la función radial, $X(t)$ el vector de entrada, C_i el vector con los centros de la función y d_i la desviación de la función, por tanto, la activación depende de la distancia euclídea entre las entradas y los centros. Algunas de las funciones en base radial más utilizadas son [26]:

Función Gaussiana:

$$\phi(r) = e^{-\frac{r^2}{2}}$$

Función inversa cuadrática:

$$\phi(r) = \frac{1}{1+r^2}$$

Función inversa multicuadrática:

$$\phi(r) = \frac{1}{\sqrt{1+r^2}}$$

3.2.2.4. Otros modelos

Autoencoder: red neuronal empleada para aprender representaciones compactas y eficientes de datos no etiquetados. Está formado por dos partes principales, un codificador que transforma los datos de entrada en una representación codificada, y un decodificador que los reconstruye a partir de esta representación. Ambos son redes neuronales que se construyen utilizando cualquier arquitectura [34].

En el entrenamiento, el Autoencoder intenta que la diferencia entre la entrada y su reconstrucción (función de pérdida) sea mínima, consiguiendo así capturar características relevantes de los datos.

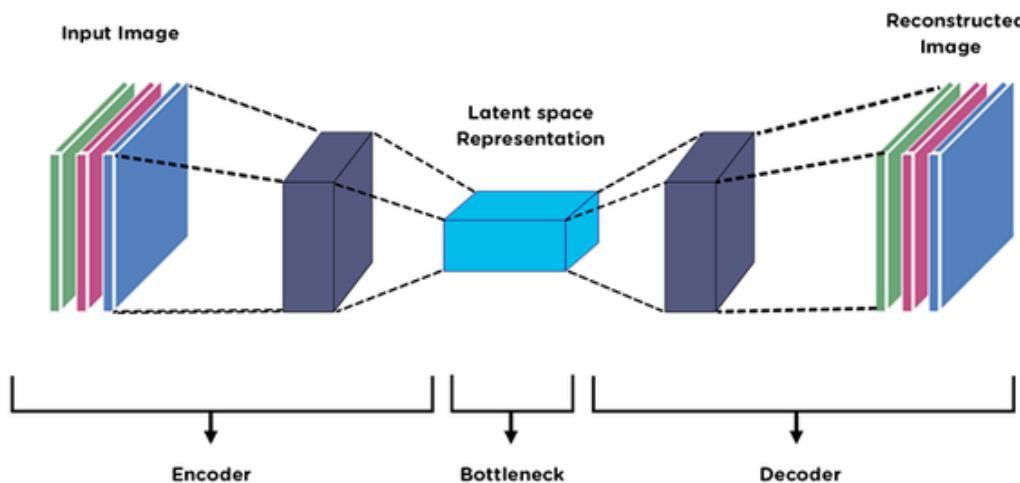


Figura 3.6: Estructura del modelo Autoencoder

Red Neuronal de Impulsos: (Spiking Neural Networks, SNN), en estas redes las neuronas se comunican con impulsos discretos en lugar de con valores continuos, simulando así el funcionamiento de las neuronas biológicas. Estos impulsos imitan el proceso de activación y propagación de señales en el cerebro humano [35, 36].

3.3. Aprendizaje profundo

El aprendizaje profundo, o Deep Learning, está relacionado con las redes neuronales profundas. Este tipo de redes surge para ponerle una solución al principal inconveniente de las redes superficiales, el crecimiento del tamaño de las capas conforme la complejidad del problema aumentaba, esto afectaba a la capacidad de los recursos computacionales, así como a la capacidad de aprendizaje y generalización de la red.

Como solución se optó el sustituir la estructura de pocas capas muy grandes por una con mayor número de capas pero que contuvieran menos neuronas y se observó que el número total de parámetros disminuía y que la capacidad de aprendizaje y generalización de la red en mucho casos superaba a la estructura anteriormente usada.

Actualmente, una de las aplicaciones más destacadas de las redes neuronales profundas es la Visión por computador, es decir, el tratamiento de imágenes y videos con diferentes objetivos. Con ayuda de herramientas matemáticas provenientes de la geometría y la estadística estas redes extraen infinidad de características de los píxeles de una imagen.

3.4. Redes convolucionales

Las redes convolucionales, o redes neuronales convolucionales (CNN, Convolutional Neural Networks), son un tipo de arquitectura de redes neuronales diseñada para procesar datos que tienen una estructura de cuadrícula, como las imágenes. Se crearon inspirándose en el sistema visual de los animales y se utilizan, mayoritariamente, para el reconocimiento de imágenes y el procesamiento de videos, aunque tiene múltiples aplicaciones y, actualmente, es el método más utilizado para la extracción de características en imágenes.

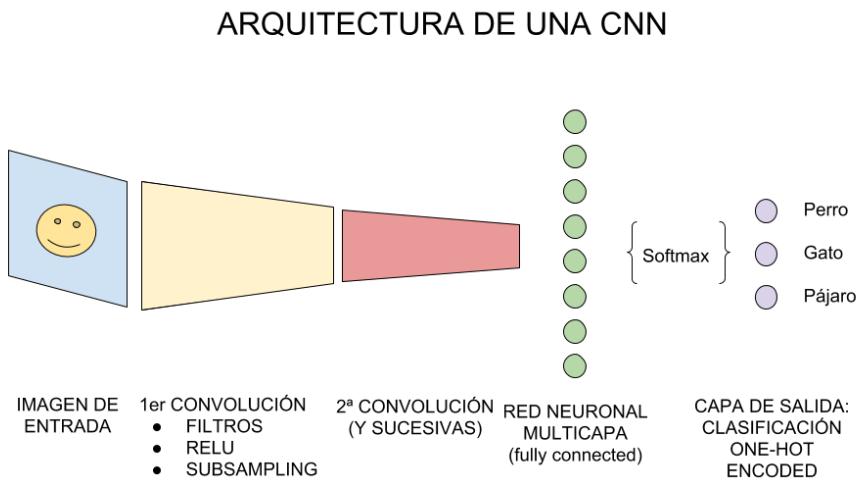


Figura 3.7: Arquitectura de una red Convolutacional

Estas redes están formadas por varias capas que vamos a detallar a continuación.

3.4.1. Capa convolucional

Esta capa tiene como función principal la extracción de características. El proceso llevado acabo consiste en la aplicación de distintos filtros para extraer propiedades concretas y distintivas de la imagen.

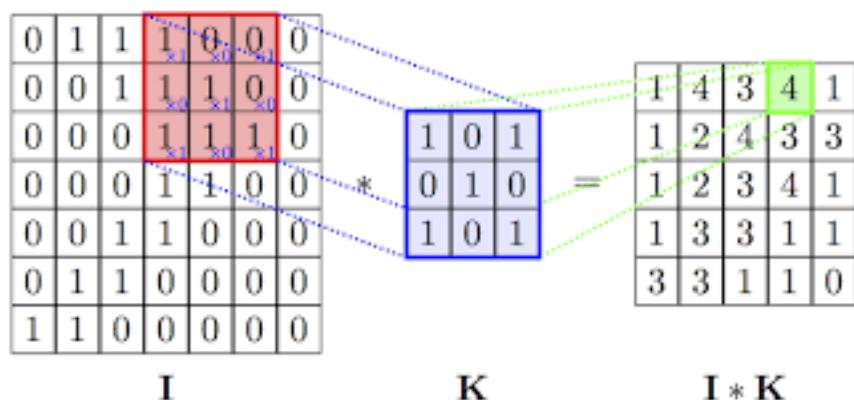


Figura 3.8: Estructura de la capa convolucional

Para ello se aplica una matriz kernel a cada pixel de la imagen, pudiendo generar distintos tamaños de salida, dependiendo del tipo de aplicación:

Valid: El filtro solo se aplica sobre píxeles de la imagen original, retornando una salida de menor tamaño que la original dependiendo del tamaño del filtro.

Same: El filtro puede ser aplicado sobre píxeles que sobrepasan el margen de la imagen, los cuales se rellenan con o's (padding), con el objetivo de devolver una dimensión igual a la original.

Full: El filtro se aplica a todos los píxeles de la imagen original en todas las posiciones posibles, rellenando con o's los márgenes exteriores. Retornando así una salida de tamaño mayor del de la imagen original.

Los valores de los filtros serán las variables que el modelo ajustará para minimizar el valor de la función de pérdida durante la etapa de entrenamiento.

3.4.2. Capa Pooling

Esta capa es la encargada de reducir la dimensionalidad y como consecuencia el coste computacional.

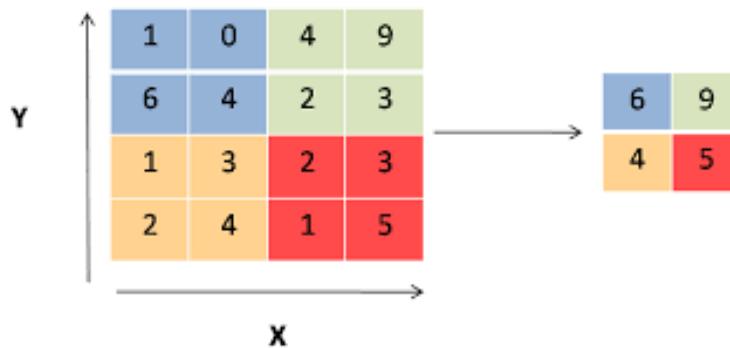


Figura 3.9: Estructura de la capa Pooling

Para ello esta capa recibe la salida proporcionada por la capa convolucional, divide la imagen en subregiones en forma de cuadrado y de cada una se extrae un único valor. Para seleccionar este valor existen diferentes técnicas, entre ellas, Max-Pooling, la cual consiste en seleccionar el máximo

valor, Average-Pooling, la cual selecciona el valor medio, o Min-Pooling, la cual toma el mínimo valor.

3.4.3. Capas Fully-Connected

Estas capas son las encargadas de aplicar el aplanado (Flatten), a la salida obtenida de las capas anteriores, para obtener así un vector de dimensión única y así con ayuda de la función Soft-max se obtiene la probabilidad de pertenencia a las distintas clases posibles. La función Soft-max se define como:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

donde:

- \mathbf{z} es el vector de entradas a la función *softmax*.
- z_i es el i -ésimo componente del vector \mathbf{z} .
- K es el número total de clases.
- $\sigma(\mathbf{z})_i$ es la salida de la función *softmax* correspondiente a la i -ésima clase.

Estas capas tienen la cualidad de que, todas las neuronas de cada capa están conectadas a cada neurona de la siguiente.

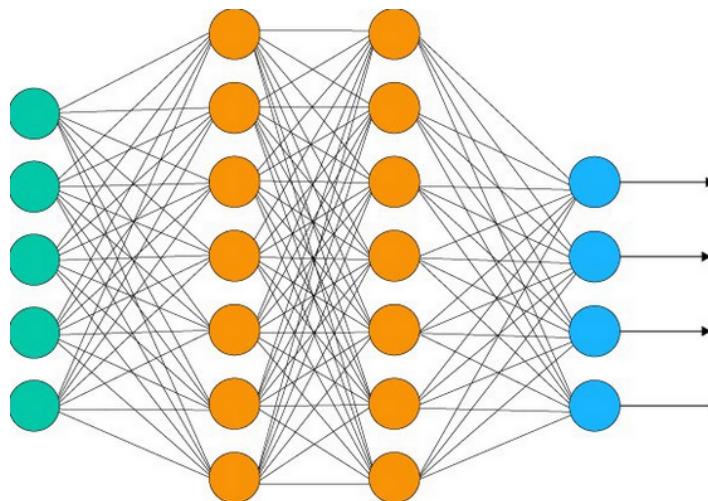


Figura 3.10: Estructura de la capa Fully-connected

3.5 Algoritmos de detección en una etapa (One stage)

Este método es muy útil, pero al aplicarlo en la detección de imágenes es necesario tener información espacial, por lo que es necesario adaptar el método de extracción de características a este requisito. Además, las redes convolucionales clásicas devuelven un tamaño de salida fijado, pero en este caso se necesita que el tamaño de salida sea variable, ya que no se conoce con exactitud el número de apariciones que va a tener cada objeto en la imagen. Por esto es necesario encontrar un método para determinar en qué subregiones conviene buscar estos objetos, es decir, en qué subregiones de la imagen se debe aplicar la red convolucional. De aquí surgen los algoritmos de detección, divididos principalmente en dos grupos que explicaremos a continuación.

3.5. Algoritmos de detección en una etapa (One stage)

Estos algoritmos realizan la detección de objetos en imágenes en una sola pasada a través de la red neuronal. Estos algoritmos combinan la localización y la clasificación de los objetos en un solo paso, lo que los hace más rápidos en comparación con los algoritmos de dos etapas, aunque a veces con una ligera reducción en la precisión.

3.5.1. RetinaNet

Este modelo utiliza las propiedades de Feature Piramidel Network (FPN) el cuál extrae mapas de características con múltiples escalas, permitiendo así la detección con precisión a distintas escalas, y las propiedades de Focal Loss, que permite manejar el gran desbalance entre las clases en primer plano y en el fondo durante el entrenamiento [37,38].

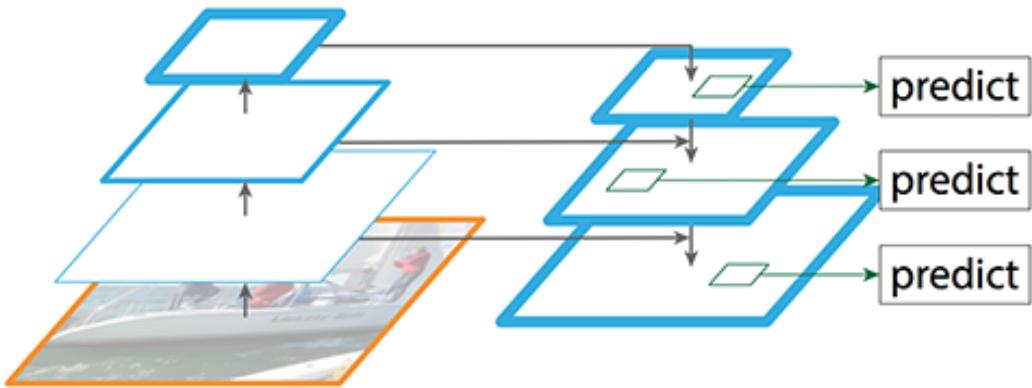


Figura 3.11: Estructura de la red FPN

El modelo FPN tiene subsecciones donde se aplican alternadamente capas convolucionales de diferentes tamaños y parámetros. Esto nos deja en la salida de cada subcapa mapas de características con 256 canales cada uno con diferentes tamaños de submuestreo, permitiendo la detección de objetos a distintas escalas dentro de la imagen.

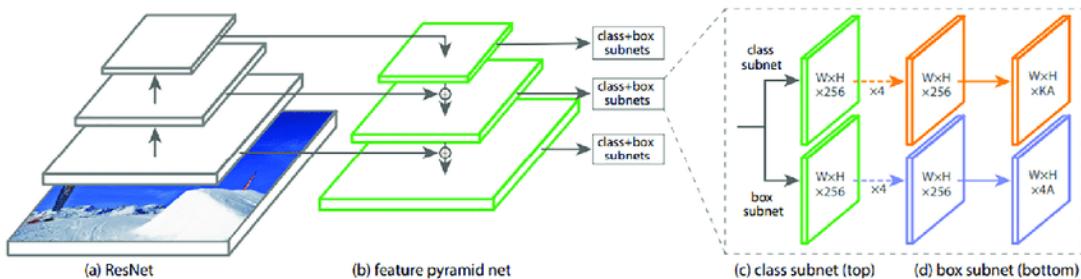


Figura 3.12: Estructura del modelo Retinanet

Como vemos en la imagen, en las zonas donde se va a evaluar si existen objetos se parte de unos anclajes (Anchors), que son cajas con distintas áreas, cubriendo las escalas de la FPN, y de diferentes ratios (1:2, 1:1, 2:1), que mapean toda la imagen. Existe una capa principal encargada de las predicciones, esta obtiene el mapa de características para una imagen de entrada y dos 'subnets':

Classification Subnet: es la encargada de predecir la probabilidad de encontrar un objeto en una posición espacial para cada Anclaje y cada clase a

3.5 Algoritmos de detección en una etapa (One stage)

considerar.

Box Regression Subnet: es la encargada de ajustar las cajas de los anclajes de cada localización espacial a un posible objeto en esa zona. Para ello considera 4 parámetros por cada anclaje que intentan ajustar el anclaje a los valores reales de las cajas existentes.

3.5.2. YOLO (You Only Look Once)

Una de las principales propiedades de este algoritmo es que utiliza características de toda la imagen para realizar la detección. Para ellos se divide la imagen en una matriz de celdas AxA en la que se predicen N cajas con una puntuación de confianza. La arquitectura de la red para hacer estas predicciones sigue un procedimiento análogo al del resto de modelos, usando capas convolucionales anidadas y Max Pooling para la extracción de características, y capas Fully-Connected para las probabilidades de pertenencia a cada clase y las coordenadas de estas [39].

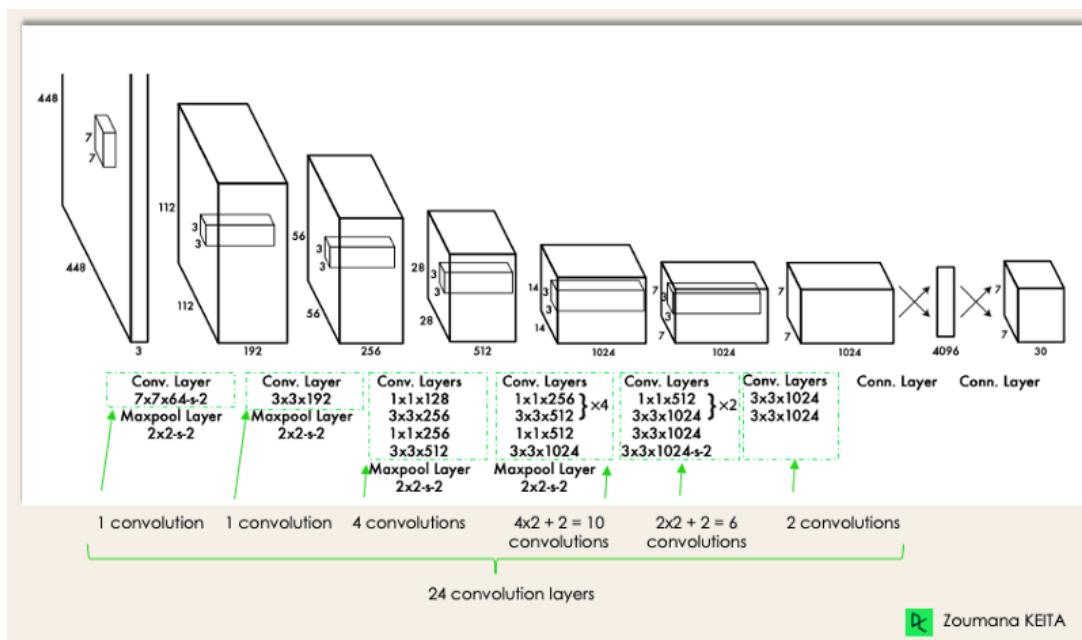


Figura 3.13: Esquema del algoritmo YOLO

Generalmente, los algoritmos de dos etapas logran una mayor precisión en la detección. Sin embargo, si buscamos una detección en tiempo real con un alto número de imágenes por segundo, los algoritmos de una sola etapa pueden ofrecer una mayor velocidad de procesamiento. Por lo tanto, no existe un tipo de algoritmo que sea superior en todos los casos.

3.6. Algoritmos de detección en dos etapas (Two stages)

Estos algoritmos dividen el proceso de detección en dos etapas, añadiendo a la capa básica de detección de objetos de distintas clases en las diferentes regiones una etapa previa donde se extraen las regiones potenciales de contener un objeto.

3.6.1. R-CNN

El principal problema de los algoritmos en dos etapas es el gran número de regiones generadas a explorar. Para solucionar esto, se busca dividir la imagen en zonas candidatas a contener objetos a detectar. Para encontrar estas regiones existen diferentes técnicas, una de las más usadas es el método de búsqueda selectiva, este combina la búsqueda exhaustiva y la segmentación, seleccionando unas 2000 regiones con forma rectangular, basándose en las propiedades similares de ellas [40, 41].

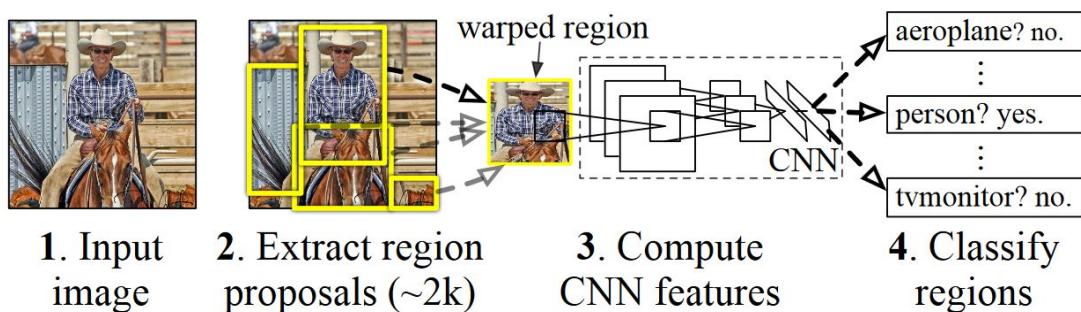


Figura 3.14: Esquema del algoritmo R-CNN

La capa CNN obtiene las características de las diferentes regiones rectangulares, las cuales se utilizan como parámetros para un algoritmo Suport Vector Machine (SVM), el cual devuelve la probabilidad de que un objeto se

encuentre en cada región. A continuación, para acotar la región seleccionada y disminuir posibles errores de segmentación, se optimizan 4 parámetros de desplazamiento que indican si mejora el ajuste de la caja al objeto en las 4 direcciones. Este algoritmo tenía como desventaja el tiempo de procesamiento necesario para procesar las aproximadamente 2000 regiones generadas, para solventar este problema el propio creador del algoritmo R-CNN sugirió la siguiente solución que vamos a estudiar [42,43].

3.6.2. Fast R-CNN

En este algoritmo toda la imagen es transformada mediante una red convolucional. Tras obtener el mapa de atributos, se extraen las regiones de interés correspondientes a las regiones propuestas por el algoritmo de búsqueda selectiva. Tras esto, con ayuda de una capa RoI-Pooling se les da la misma dimensión a todas las regiones, para después hacer la clasificación de estas y predecir así la clase y ajustar los parámetros de desplazamiento al igual que en el algoritmo R-CNN.

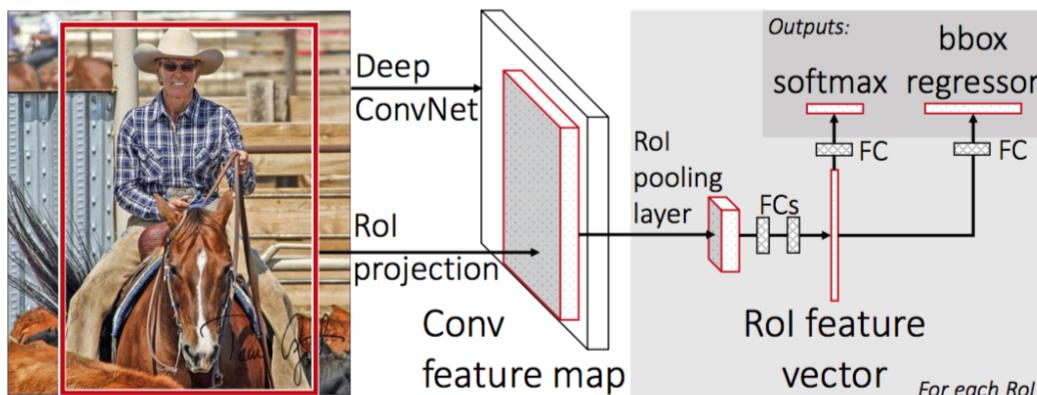


Figura 3.15: Esquema del algoritmo Fast R-CNN

3.6.3. Faster R-CNN

El siguiente paso en la evolución de estos modelos fue utilizar una red separada para predecir directamente las regiones propuestas sobre el mapa de

características generado por la capa convolucional, en lugar de emplear la búsqueda selectiva. Posteriormente, estas regiones se transforman mediante una capa ROI pooling, cuya salida se utiliza para clasificar la imagen de manera similar a los demás casos.

3.6.4. Mask R-CNN

Este modelo es una extensión del algoritmo Faster R-CNN que incluye, en paralelo, un método para predecir la máscara del objeto, es decir, su contorno específico dentro de la región delimitada por la caja (bounding box). Esto se consigue añadiendo un término a la función de pérdida que codifica K (número de clases) máscaras binarias [44].

Mask R-CNN → Faster R-CNN + FCN

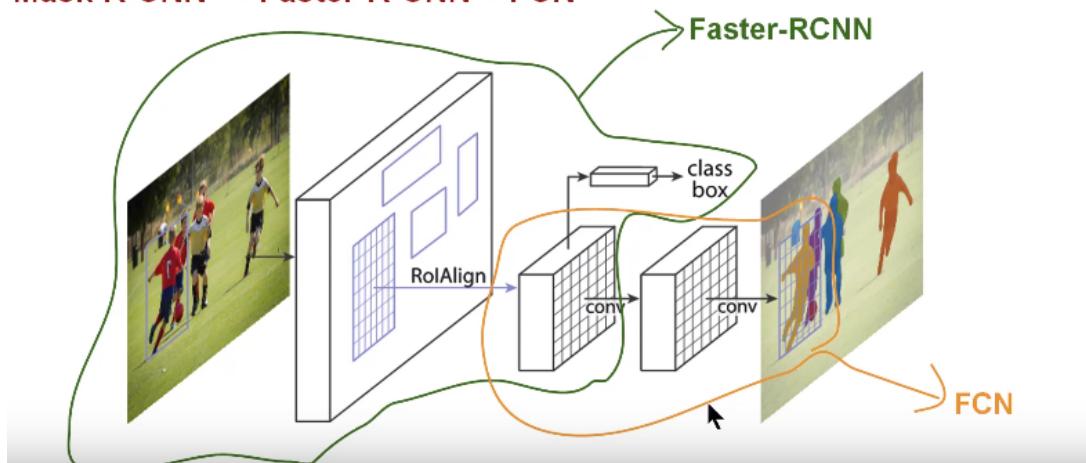


Figura 3.16: Esquema del algoritmo Mask R-CNN

En primer lugar, el algoritmo localiza el objeto y después aplica una máscara binaria pixel a pixel que determina todas las características que le pertenecen y, por consecuencia, su perfil.

4 Diseño del proyecto

En este capítulo, ilustramos y explicamos el proceso seguido para llevar a cabo este proyecto y presentamos la principal librería utilizada para desarrollarlo.

4.1. Workflow del proyecto y planificación temporal

En este apartado vamos a ver cómo hemos distribuido el trabajo a realizar en distintas tareas a lo largo del año.

En primer lugar, observamos un *Workflow*, que consiste en un diagrama ilustrativo, en el que englobamos en grandes bloques las principales tareas que hemos realizado en este proyecto.

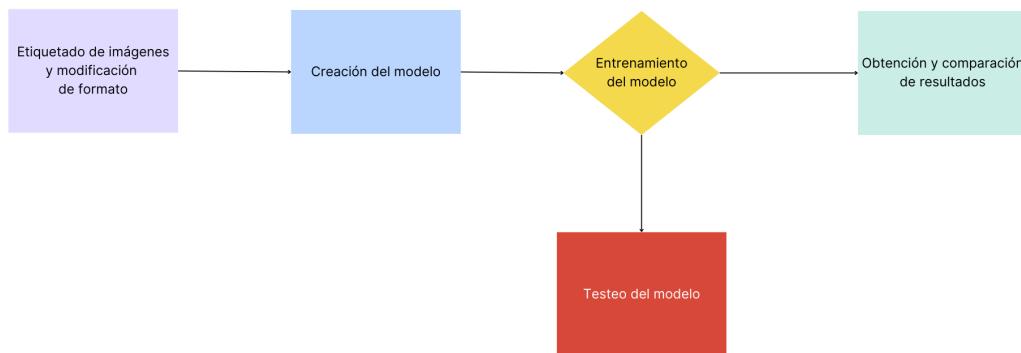


Figura 4.1: Workflow del proyecto

Destacamos, entre todo el trabajo realizado, las tareas relacionadas con el etiquetado de imágenes, la creación del modelo, el entrenamiento y posterior

testeo del mismo y la creación de un mapa de distribución de Baobabs.

A continuación, se muestra un diagrama de Gantt, en el que podemos ver la distribución temporal de las tareas descritas.

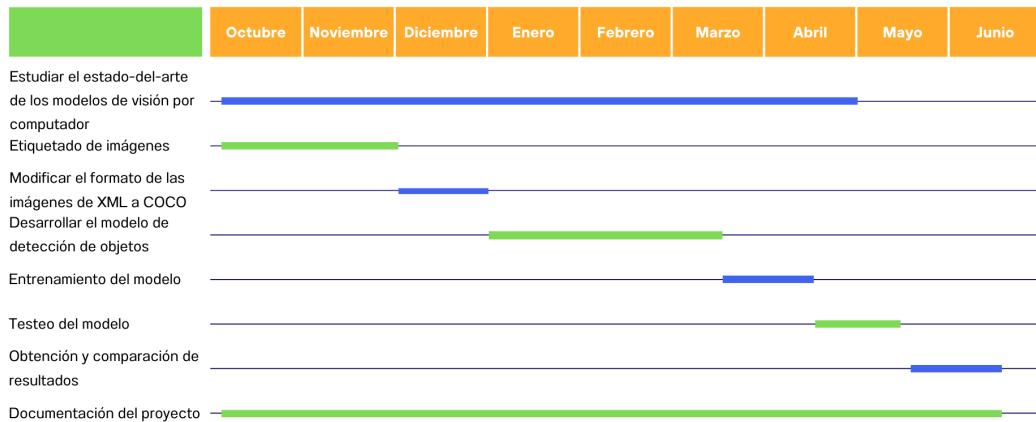


Figura 4.2: Diagrama de Gantt, que distribuye en el tiempo las tareas realizadas

4.2. Detectron2 para la detección en imágenes

Actualmente, los modelos de detección basados en redes neuronales convolucionales constituyen el estado-del-arte. En particular, Yolo presenta el modelo de detección más rápido mientras Faster R-CNN se considera como el modelo de detección de objetos más preciso [45].

En este trabajo hemos utilizado la librería de código abierto Detectron2 creada por Facebook AI Research sobre PyTorch [46]. Esta librería permite, en función del fichero de configuración que se seleccione, utilizar varios algoritmos de detección de objetos estudiados anteriormente: Mask R-CNN, RetinaNet, Faster R-CNN, Fast R-CNN, junto a otros que no hemos detallado, pero resultan también muy interesantes como PointRend o DensePose. Nosotros finalmente nos hemos decantado por utilizar Faster R-CNN, ya que hemos priorizado la precisión frente a la velocidad.

4.2 Detectron2 para la detección en imágenes

En la imagen podemos ver las diferentes funcionalidades que tiene Detectron2, aunque nosotros nos centraremos en el uso de la detección de objetos con cajas (Bounding-Box).

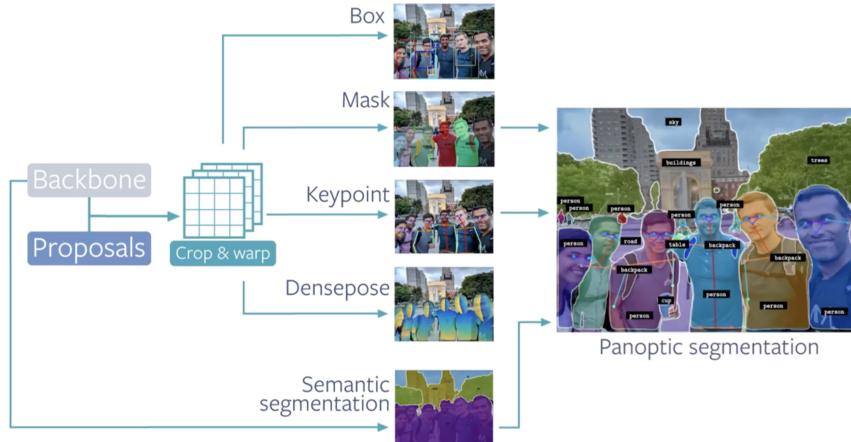


Figura 4.3: Funcionalidades de la librería Detectron2

4.2.1. Estructura de la red

Vamos a ver más en detalle la estructura de red generada por esta librería [47].

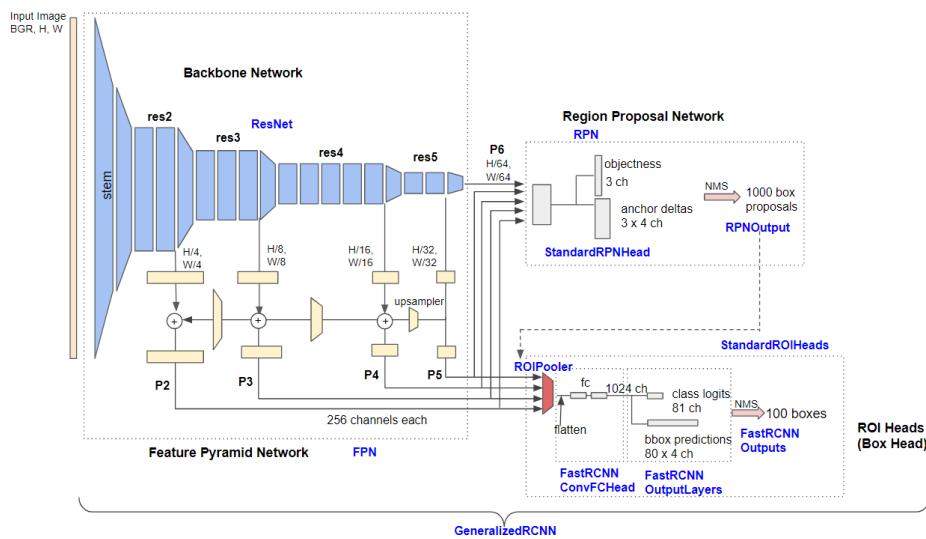


Figura 4.4: Estructura de la red usada en la librería Detectron2

Esta está formada por tres partes principales:

Backbone Network: es la parte encargada de extraer los mapas de características de las imágenes de entrada, usando el algoritmo visto anteriormente FPN (Feature Piramide Network), el cual extrae los mapas con diferentes escalas. Esta parte de la red está dividida en subcapas, cuyas salidas son mapas de características con 256 canales, cada uno con diferentes tamaños de submuestreo, permitiendo así la detección de objetos de distintos tamaños dentro de la imagen [48].

Region proposal Network: La entrada de esta capa es la salida de la anteriormente descrita, es decir, los mapas de características con distintas escalas y con ellos genera más de 1000 regiones en las que potencialmente podría haber objetos a detectar. Para lograr esto, cuenta con una red neuronal y varias funcionalidades complementarias la cual requiere de un entrenamiento independiente.

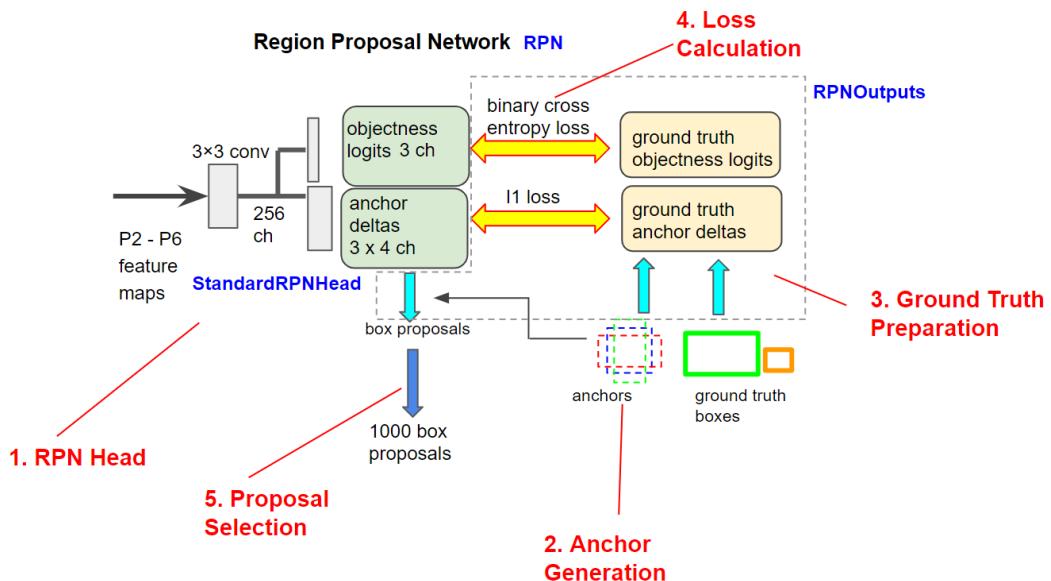


Figura 4.5: Estructura de la capa Region proposal Network

En la imagen podemos observar las distintas partes por las que está formada. En el primer paso son introducidos como entrada los mapas de características generados por la Backbone Network y se genera un

mapa de probabilidad de existencia de objetos y la forma de las cajas relativas a estas.

A continuación, se generan una serie de anclajes y para cada nivel del modelo se fija un tamaño de celda (32, 64, 128, 326, 512) y unas proporciones (1:2, 1:1, 2:1). Después, se divide la imagen en una matriz con la dimensión del mapa de características de cada nivel y en cada esquina se sitúan los 3 Anclajes.

El siguiente paso, es el cálculo de métrica de Intersección sobre la Unión (IoU), la cual explicaremos más adelante, para obtener los anclajes más representativos y se implementan gradientes en las 4 direcciones para ajustarlos lo máximo posible al ground Truth.

Por último, se seleccionan 2000 Anclajes optimizados por cada subnivel ordenados según las métricas calculadas y después, se seleccionan 1000 cajas, cada una con una puntuación(score), con ayuda de la técnica de 'Non-Maximum Supresion' [49].

ROI Heads: En esta última capa, se extraen las propuestas de caja de los diferentes mapas de características procedentes de la primera capa. Para decidir de qué nivel de submuestreo extraer la caja se aplica la siguiente fórmula:

$$\text{feature_level} = \text{floor}\left(4 + \log_2\left(\frac{\sqrt{\text{box area}}}{224}\right)\right)$$

A continuación, estas regiones son normalizadas a una dimensión 7x7 para los 256 canales y se pasan como entrada a las redes ROI (box) Head, en las cuales se aplican dos funciones de pérdida una para la localización de objetos y otra para su clasificación.

4.2.2. Métricas de evaluación

Intersection Over Union: Esta métrica calcula el área de intersección de la detección y el ground truth normalizada a la unión de ambas.

viso.ai

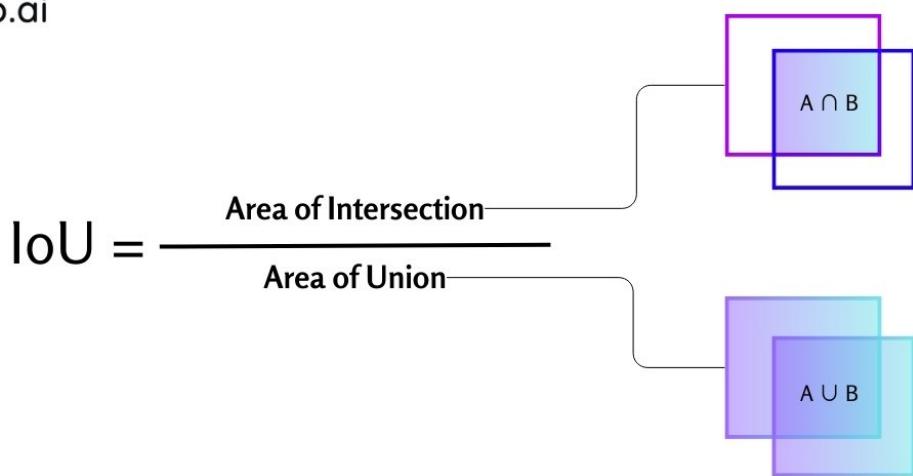


Figura 4.6: Fórmula matemática de la métrica Intersection over Union (IoU)

Por lo tanto, si el resultado de la métrica es 1 significa que la detección es exactamente igual al ground truth, y si es 0, significa que no ha habido ninguna coincidencia. Para penalizar los objetos no detectados, el valor medio de esta métrica se calcula utilizando todos los objetos existentes. En el caso de objetos detectados que no se corresponden con ningún objeto existente, no se incluye ninguna consideración aquí.

Non-Maximum Supresion: Esta técnica sigue los siguientes pasos para filtrar las regiones propuestas:

1. Selecciona la región con la puntuación(score) más alta y por lo tanto mayor confianza en la clasificación.
2. Calcula la métrica IoU de la región seleccionada con todas las restantes y elimina aquellas que superen un umbral predefinido.
3. Repite los pasos anteriores hasta llegar al número de regiones deseadas.

5 Análisis experimental

En este capítulo se explica el proceso seguido hasta obtener los resultados objeto de estudio, que ha consistido en un preprocesamiento del conjunto de datos, la creación del modelo, el testeo del mismo y la obtención de unos datos cualitativos y cuantitativos que nos permiten comparar las distintas configuraciones utilizadas.

5.1. Preparación de los datos (Pre-procesado)

El conjunto de datos brutos proporcionado por los tutores, consistente en 464 imágenes, de resolución 1 metro por pixel, de Google Earth de zonas de Madagascar en las que hay Baobabs, junto a un mapa de Google Maps en el que estaban marcadas las ubicaciones de algunos de estos árboles.

Para entrenar el modelo de detección bajo el paradigma supervisado ha sido necesario anotar las imágenes indicando dónde se encuentra cada Baobab con un Bounding Box. Para ello se usó el programa LabelImg, una herramienta gráfica de código abierto para el etiquetado de imágenes, que es habitualmente utilizado para construir conjuntos de datos de anotaciones para tareas de visión por computador, como la detección de objetos y la segmentación de imágenes. Su código está escrito en Python y se apoya en el framework Qt para construir la interfaz gráfica.

LabelImg devuelve las imágenes etiquetadas con formato YOLO o Pascal VOC XML y necesitábamos tenerla en formato COCO para poder entregárselas al modelo. El formato COCO (Common Objects in Context) es un estándar utilizado para anotaciones en conjuntos de datos de visión por computadora, principalmente en tareas de detección de objetos, segmentación de instancias, y segmentación de personas. Es conocido por su detallada estructura de anotaciones, que lo hace muy útil para la evaluación precisa de algoritmos de visión por computador complejos. Para hacer este paso entre formato de an-

taciones creamos un Script, en el cual aprovechamos para dividir el conjunto de datos en tres subconjuntos: uno de entrenamiento, que utilizamos el 70 % de los datos (324 imágenes), uno de validación y uno de test, repartiendo el 30 % de imágenes restantes entre ellos (70 imágenes en cada uno). El conjunto de test es el utilizado para probar el funcionamiento del modelo [50]. (Ver resumen en Tabla 5.1).

	Train	Validation	Test	Total
Nº Imágenes	324	70	70	464

Tabla 5.1: Tabla de distribución de imágenes

Etiquetamos objetos de dos clases: Baobabs con sombra y Baobabs sin sombra, ya que buscamos comprobar si el uso de la sombra del baobab aporta más exactitud al modelo, para ello le pediremos al algoritmo que haga la búsqueda solo entrenando el modelo con los datos sin sombra y luego con sombra para poder hacer una mejor comparativa.

5.2. Creación y entrenamiento del modelo

Antes de construir nuestro modelo hubo que hacer una preparación previa del entorno, ya que se ha hecho uso de los servidores de la Universidad de Granada, por la necesidad de disponer de una tarjeta gráfica NVidia CUDA-compatible y con sistema operativo Linux y Python ≥ 3.7 , ya que detectron2 lo exigía y no disponíamos de un equipo con estas propiedades. Creamos un entorno virtual con Python 3.9 para que fuera compatible con la instalación de Pytorch y Torchvision que también eran un requisito obligatorio antes de finalmente poder instalar el paquete Detectron2.

Para la creación del modelo se hizo una primera prueba con la configuración Faster R-CNN R50 FPN 3x, pero los resultados obtenidos no fueron nada buenos, por lo tanto, hicimos una segunda prueba con Faster R-CNN X101 32x8d FPN 3x y en este caso sí que obtuvimos buenos resultados por lo que decidimos quedarnos con este modelo ya que lo que buscábamos era comparar los modelos utilizando la sombra de los baobabs y sin utilizarla, no comparar los diferentes algoritmos. Entre estos dos modelos la diferencia más reseñable es la profundidad de Resnet utilizada, en Faster R-CNN

5.2 Creación y entrenamiento del modelo

R₅₀ FPN 3x es de 50 capas y en Faster R-CNN X₁₀₁ 32x8d FPN 3x es de 101 capas. El modelo elegido se trata de un algoritmo Faster R-CNN con una Backbone Network de tipo FPN y con el programa de entrenamiento 3x (aproximadamente 37 COCO epochs).

Fijamos los siguientes valores en la configuración del modelo:

Workers: 4

Ejemplos de entrenamiento utilizados en cada iteración: 4

Tasa de aprendizaje: 0.001

Iteraciones de calentamiento: 1000

Número máximo de iteraciones: 1501

Número de regiones por imagen utilizadas para entrenar RPN: 64

Como hemos usado el Default Trainer, la función optimizadora por defecto es el descenso de gradiente estocástico (Stochastic gradient descent, SGD), cuyo objetivo es reducir al mínimo la función de costo al ajustar los parámetros del modelo de forma iterativa, utilizando la retroalimentación del conjunto de datos de entrenamiento. A diferencia de otros modelos anteriores en lugar de actualizar los parámetros del modelo con todos los datos de entrenamiento en cada iteración, el descenso de gradiente estocástico usa muestras pequeñas y aleatorias de los datos de entrenamiento ("minilotes.^º "batches"), en cada paso, para actualizar los parámetros del modelo.

Con esta configuración y el conjunto de datos de entrenamiento y validación se entrenó el modelo y con el de test se hizo el testeo del modelo obteniendo las imágenes y tablas que mostraremos en los siguientes apartados.

Ejecutamos el modelo de dos formas diferentes, pidiéndole que buscara solo los Baobabs sin hacer uso de su sombra e indicándole que si la utilizará para así poder obtener resultados y poder comparar.

5.3. Testeo del modelo

Para poder analizar el resultado de nuestro modelo, tras entrenar ejecutamos nuestro algoritmo sobre un conjunto de imágenes de test y obtuvimos gráficamente los resultados que veremos a continuación.

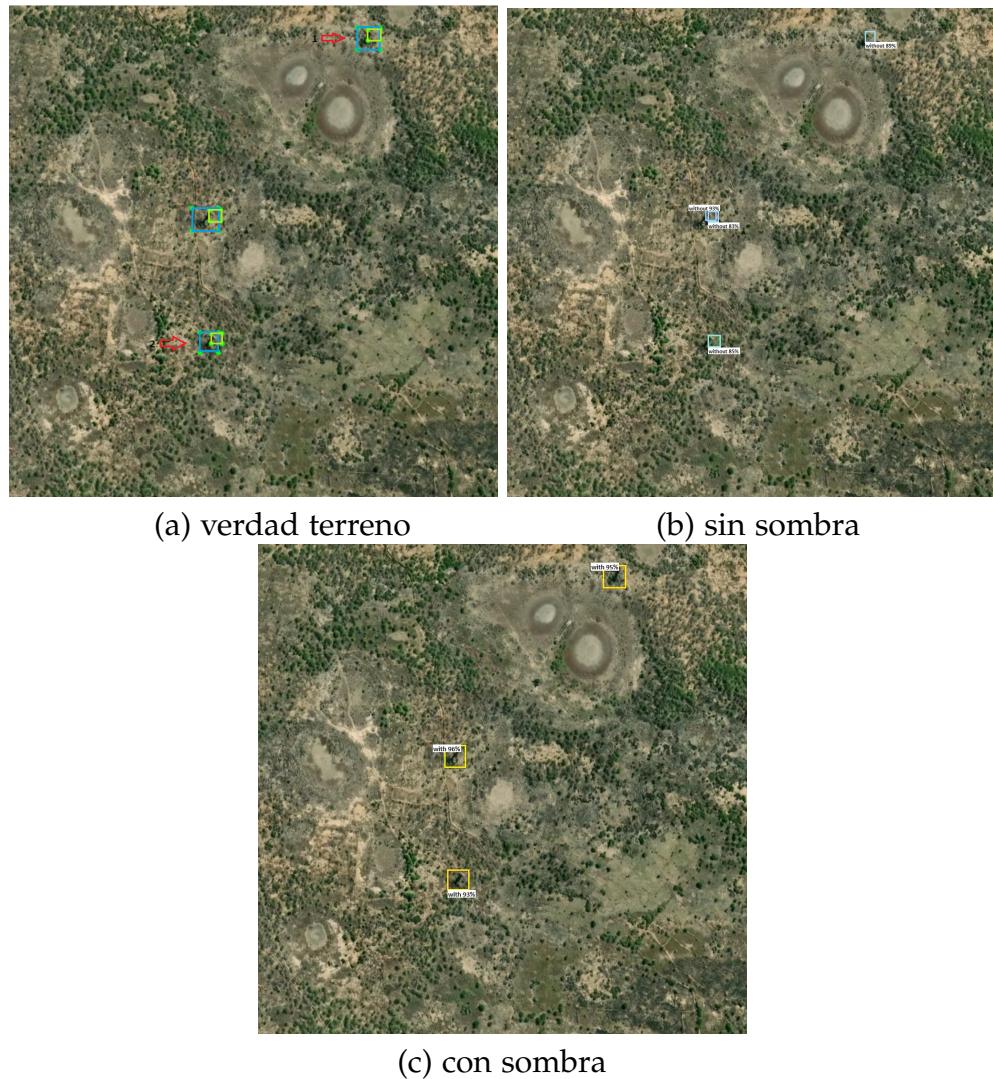


Figura 5.1: Imagen del lugar con coordenadas (44.39556708, -20.24249308) con verdad terreno, resultados de la detección sin (b) y con sombra (c)

En este primer análisis podemos ver como los dos algoritmos han encontrado los tres árboles, aunque podemos observar como en el algoritmo con sombra el grado de fiabilidad es mayor.

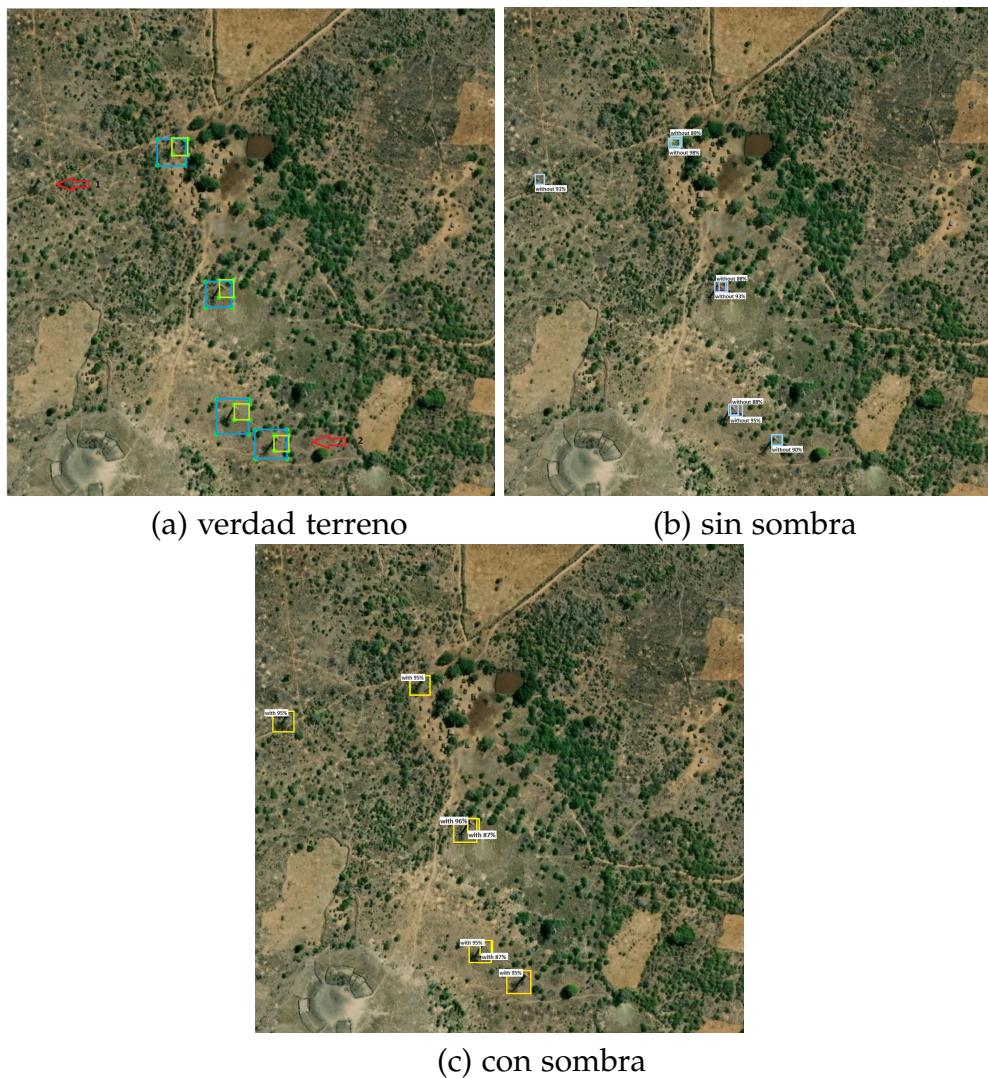


Figura 5.2: Imagen del lugar con coordenadas (44.42125889, -20.23153587) con verdad terreno, resultados de la detección sin (b) y con sombra (c)

En este caso, podemos ver como ha habido un error en el etiquetado (flecha 1) que el algoritmo ha sabido solventar. En los dos algoritmos han sido encontrados los cuatro baobabs que podemos ver en la imagen. Tanto en el algoritmo de solo con sombra como en el solo sin sombra vemos como a pesar de detectar correctamente los objetos, algunos los ha detectado varias veces con distinto grado de fiabilidad.

5 Análisis experimental

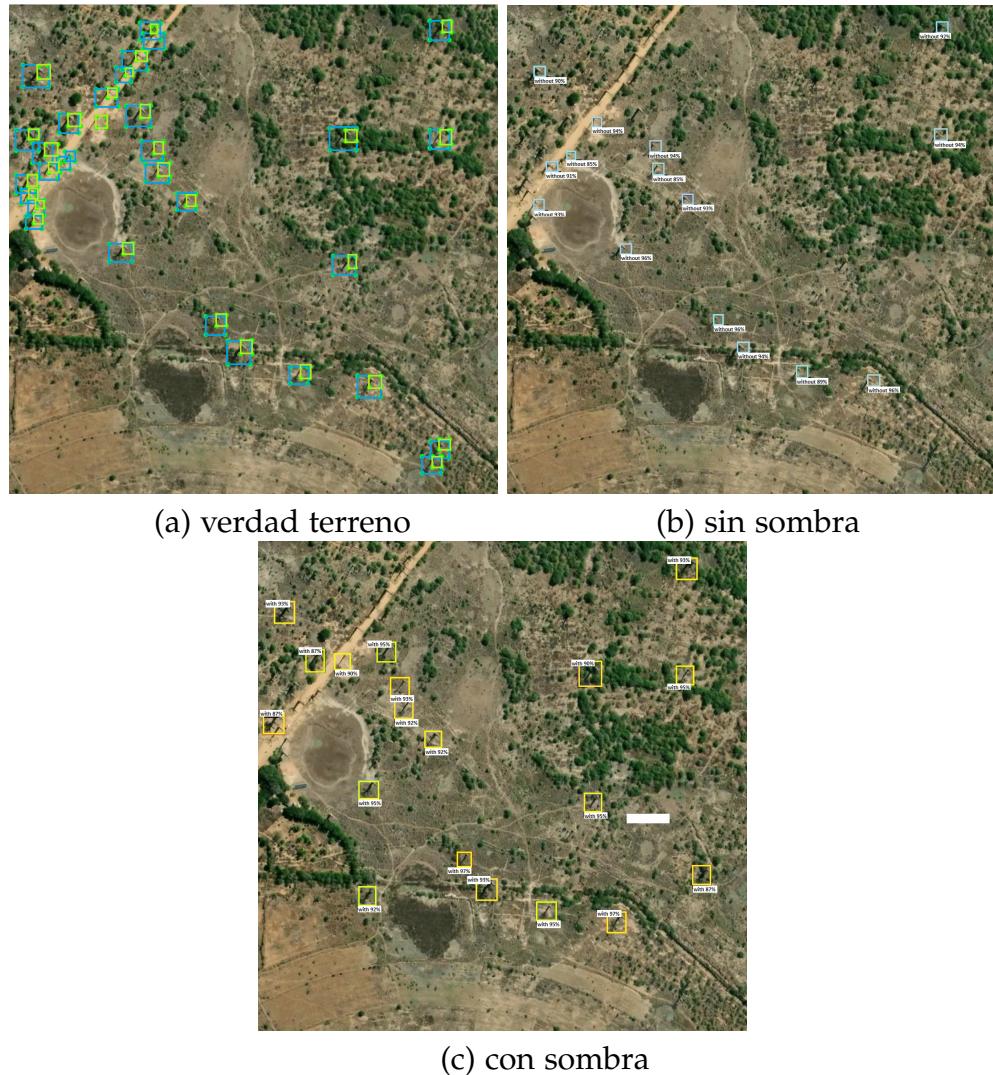


Figura 5.3: Imagen del lugar con coordenadas (44.42125889, -20.25125829) con verdad terreno, resultados de la detección sin (b), con sombra (c)

En esta tercera imagen, la complejidad del terreno afecta a los algoritmos. Había 33 árboles etiquetados, pero ninguno de los algoritmos detectó todos los baobabs, aunque sí encontraron árboles no etiquetados. El algoritmo sin sombra detectó 15 baobabs y el con sombra 19. Los árboles en el borde del camino no fueron encontrados, ya que la sombra coincide con el borde y el color del tronco del baobab es similar al del camino, dificultando su detección incluso para el ojo humano.

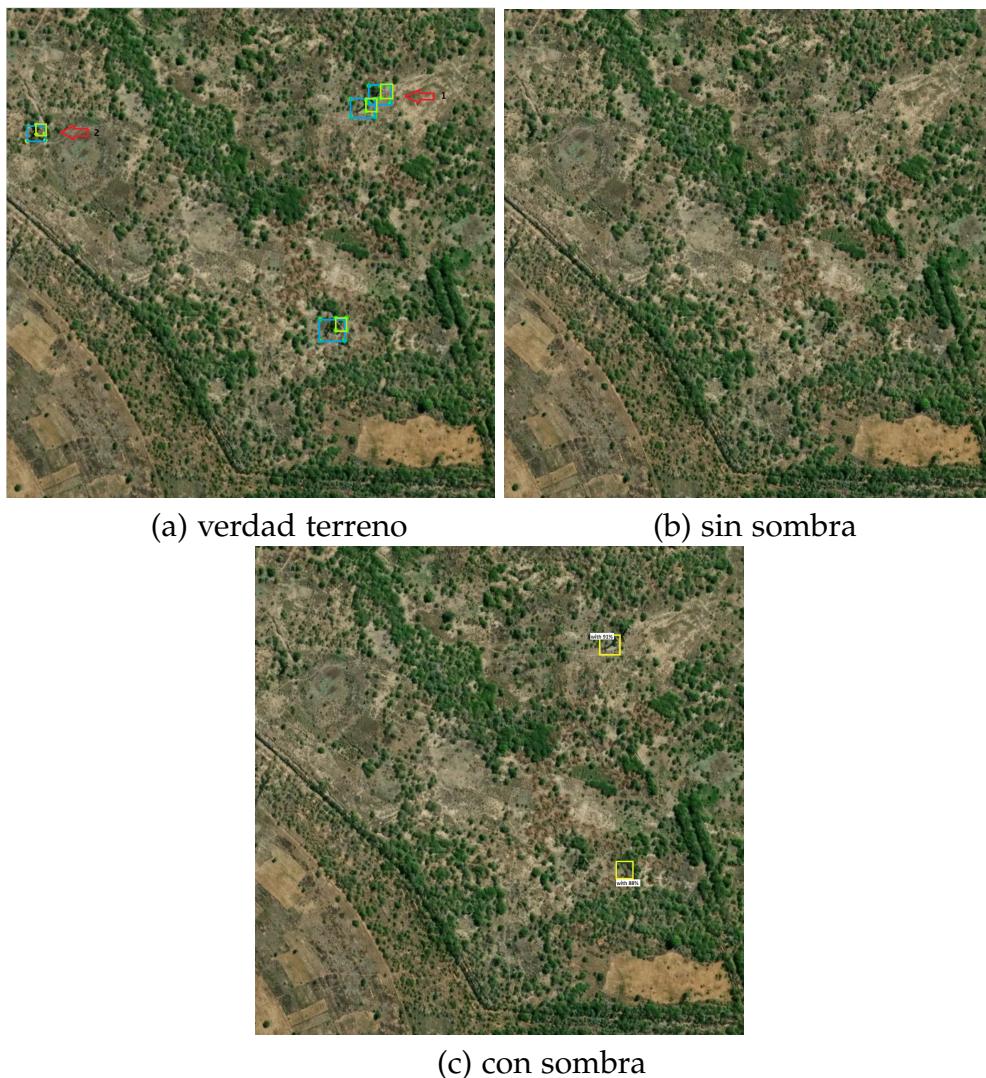


Figura 5.4: Imagen del lugar con coordenadas (44.42593013, -20.25344952) con verdad terreno, resultados de la detección sin (b) y con sombra (c)

En este análisis lo más destacable es que el sin sombra no ha detectado ningún baobab, sin embargo, vemos como si han sido encontrados los árboles utilizando la sombra, aunque el baobab marcado con la flecha 1 no ha sido detectado en ninguno de los dos algoritmos.

5 Análisis experimental

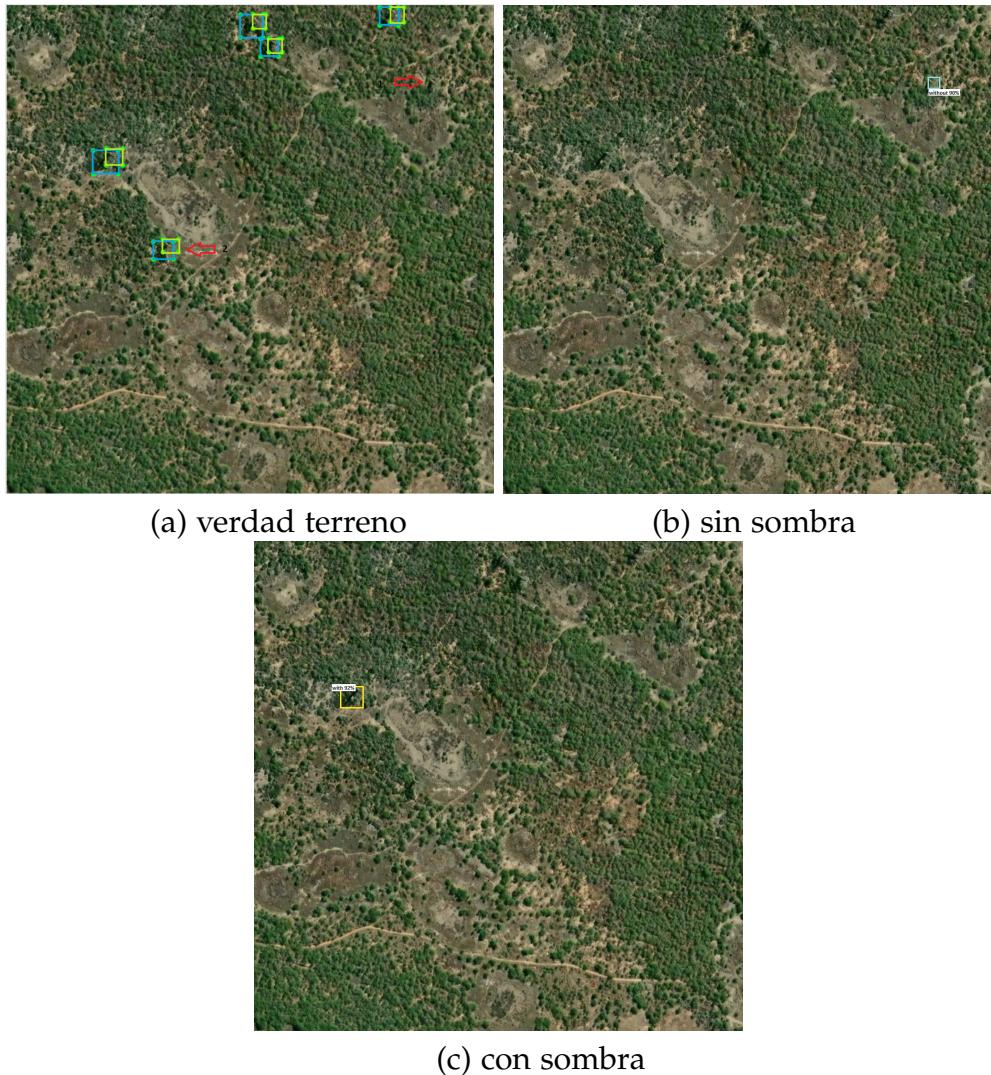


Figura 5.5: Imagen del lugar con coordenadas (44.42826575, -20.24687575) con verdad terreno, resultados de la detección sin (b) y con sombra (c)

En esta quinta imagen, el ruido del terreno es notable. El algoritmo con sombra detectó un solo baobab, mientras que el sin sombra no encontró todos los árboles etiquetados, pero descubrió uno no etiquetado originalmente (marcado con la flecha 1).

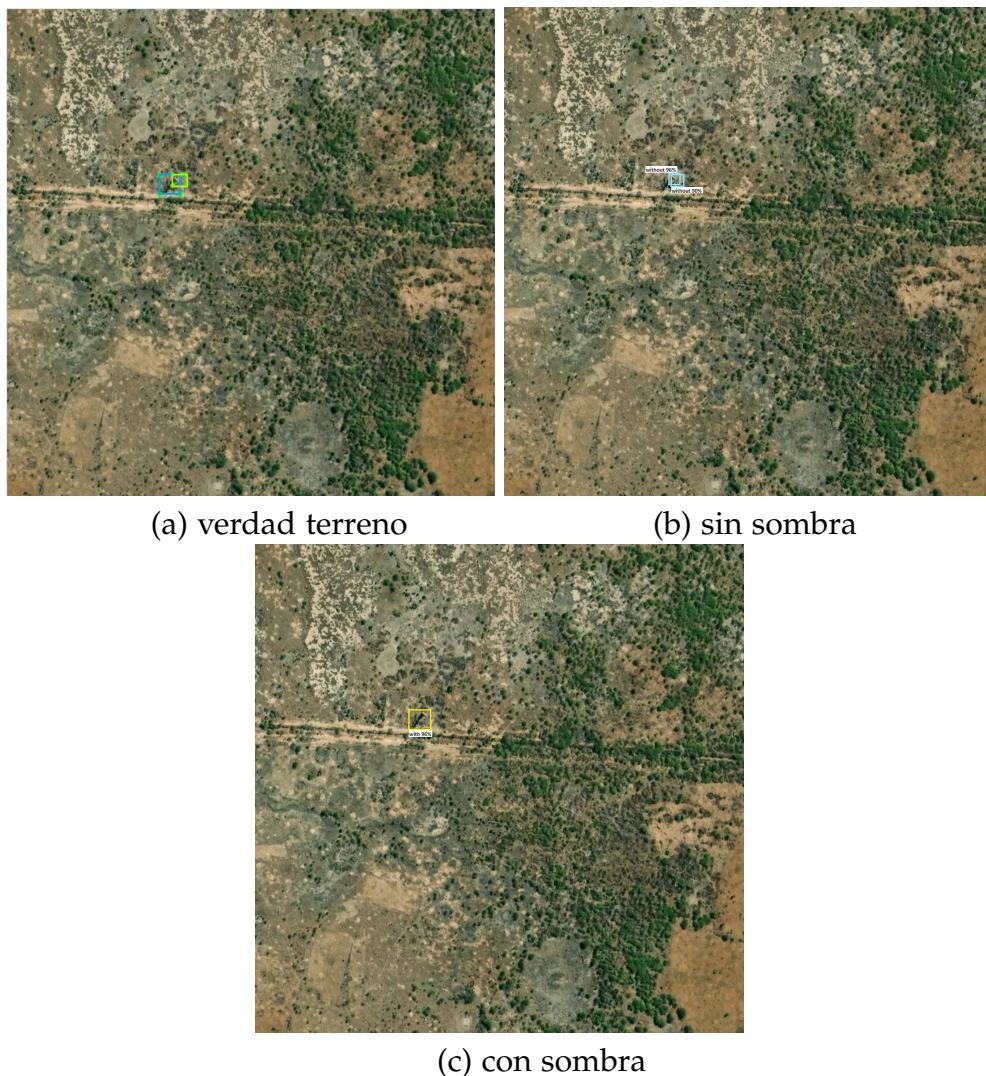


Figura 5.6: Imagen del lugar con coordenadas (44.45862881, -20.26221411) con verdad terreno, resultados de la detección sin (b) y con sombra (c)

Este sexto caso ha sido muy exitoso, ya que los dos algoritmos han detectado el único baobab que se observa en la imagen original, con la única notación de que en algoritmo sin sombra se ha detectado dos veces el mismo árbol con distintos niveles de fiabilidad. Aquí vemos como el terreno en el que se encuentra el baobab es más uniforme y no está rodeado de otro tipo de árboles.

5 Análisis experimental

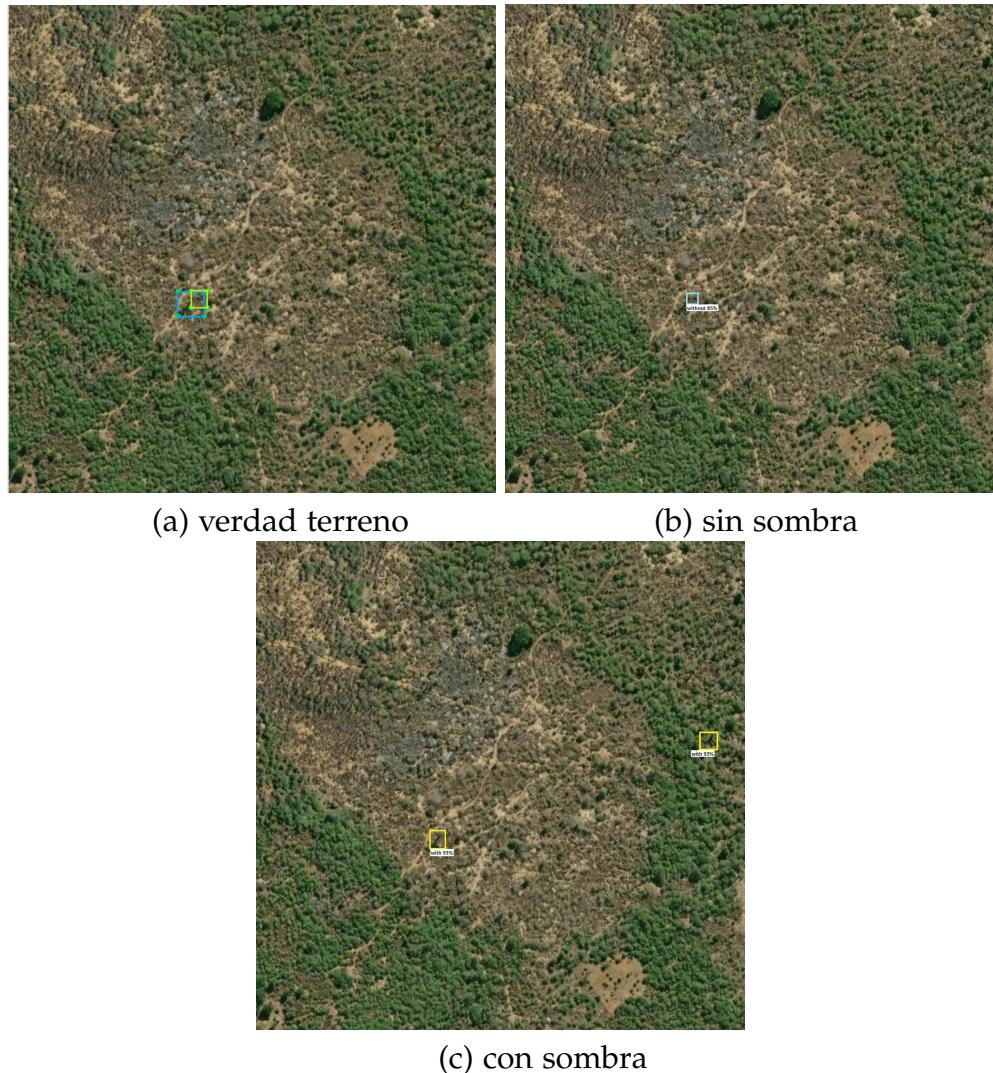


Figura 5.7: Imagen del lugar con coordenadas (44.50066997, -20.2403017) con verdad terreno(a), resultados de la detección sin (b), con sombra (c)

En este último análisis destacamos la aparición de falsos positivos, en la imagen original se observa que solo hay un árbol, pero el algoritmo que utiliza la sombra ha detectado más de uno, a simple vista los objetos detectados guardan cierta similitud con la sombra de un baobab, pero no lo son. El algoritmo que no utiliza la sombra ha detectado el objeto.

5.4. Comparativa de los resultados

A continuación, vamos a ver unas tablas con los resultados obtenidos para cada algoritmo. Vamos a utilizar la precisión media (Average Precision, AP). Para entender esta unidad de medida primero debemos definir dos conceptos:

Precisión: mide el porcentaje de verdaderos positivos predichos al hacer predicciones sobre un conjunto de datos de test.

Recall: mide el porcentaje de verdaderos positivos predichos sobre todos los casos positivos reales.

Entendiendo esto podemos definir la Precisión media, que es el área bajo la curva precision-recall (PR curve), en esta curva se sitúan en el eje X los valores de Recall y en el eje Y los de precision.

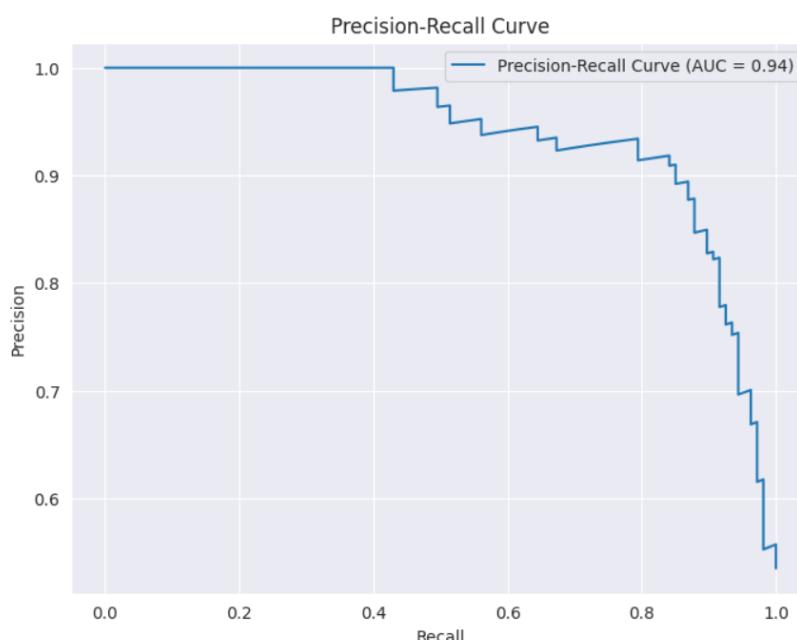


Figura 5.8: Ejemplo de gráfica de la curva Precision-Recall

También tenemos valores como AP₅₀ y AP₇₅ que sería la precisión media con un IoU (Intersection over Union) fijado a 50 % y 75 % respectivamente, así como AP_s y AP_m que se refiere al AP en objetos pequeños y medianos.

5 Análisis experimental

También existe API para objetos grandes, que en este caso no aplica ya que el tamaño de los objetos respecto de la imagen es pequeño.

En la tabla 5.2, podemos observar los resultados obtenidos en términos de AP al aplicar los algoritmos Sin sombra y Con sombra por separado y vemos como la precisión media para el algoritmo Con sombra es significativamente mejor.

Tipo de anotación	AP	AP ₅₀	AP ₇₅	AP _S	AP _M
Sin sombra	15.382	49.988	3.864	15.934	5.487
Con sombra	36.424	80.833	27.204	23.717	40.077

Tabla 5.2: Resultados de la detección del modelo entrenado sin y con sombra en término de AP con diferentes umbrales de IoU.

Viendo estos resultados y apoyándonos en las imágenes obtenidas podemos ver objetivamente que algoritmo tiene mejores resultados. Comparando el algoritmo que hace uso de la sombra y el que no vemos una diferencia de un 21.042 % en la precisión media, siendo el algoritmo con mejores resultados el que utiliza la sombra de los Baobabs para detectarlos.

A pesar de los buenos resultados obtenidos en el modelo con sombra, hemos encontrado algunos puntos a mejorar.



Figura 5.9: Árboles no detectados que se encuentran al borde de un camino.

5.4 Comparativa de los resultados

En la imagen podemos ver un ejemplo de los puntos débiles del algoritmo, en este caso el modelo no ha sido capaz de detectar estos árboles, ya que a simple vista podemos ver que la sombra del Baobab puede confundirse fácilmente con el borde del camino y el tronco tiene prácticamente la misma tonalidad que el asfalto del carril.



Figura 5.10: Sombra confundida por el modelo con un Baobab.

También hemos observado algunos falsos positivos, en la mayoría de casos sombras que pueden parecer la sombra correspondiente al Baobab, pero que en realidad son sombras generadas por otros árboles al estar muy cercanos.

Ambos casos probablemente podrían ser solventados si se obtuvieran imágenes con mayor calidad.

6 Conclusiones y trabajo futuro

Tras analizar los resultados obtenidos llegamos a la conclusión de que añadir la característica de la sombra al algoritmo si que nos aporta un extra a la hora de detectar los Baobabs y nos ayuda a poder detectar mayor cantidad de ellos que sin dar este dato extra al modelo.

Este proyecto puede tener una larga trayectoria y opciones de mejoras para trabajos futuros, algunas de ellas podrían ser las siguientes:

1. Aplicar técnicas de mejora de calidad de las imágenes para solventar los problemas detectados en nuestro modelo.
2. Hacer una comparativa con todos los algoritmos disponibles tanto de una como de dos etapas, para así ver cuál de ellos aporta mayor precisión.
3. Realizar la detección de objetos con otro método distinto al utilizado en este proyecto, la detección con Bounding Box, por ejemplo, la segmentación de imágenes. Para ello habría que hacer un preprocesamiento del conjunto de datos diferente y más exhaustivo, ya que en lugar de etiquetar las imágenes con Bounding Boxes, habría que marcar la silueta de cada Baobab, pero quizá así podría obtenerse mejores resultados.
4. Construir un Mapa de distribución de Baobabs. Una vez se tuviera un modelo definitivo, podría aplicarse el modelo a imágenes satélites de todas las zonas del planeta en las que existan Baobabs y con los resultados obtenidos construir un mapa en el que estuviera localizada la ubicación de cada uno de ellos, de este modo podría tenerse la población totalmente controlada.

7 Manual de usuario

En esta sección vamos a describir brevemente la estructura de archivos que encontramos en [el repositorio de Github](#) en el que podemos encontrar el código utilizado para este proyecto, con el fin de facilitar el trabajo futuro de investigación a partir de este trabajo.

- **tococo.py** En este fichero encontramos el código utilizado para pasar la notación de las imágenes etiquetadas de formato XML a COCO format.
- **train.py** Aquí encontramos el código utilizado para entrenar el modelo dándole como entrada las imágenes de los baobabs con las notaciones en formato COCO, este tiene como salida las imágenes de test con los baobabs encontrados.
- **README.md** En este archivo encontraremos las indicaciones necesarias para poder ejecutar el código.

Bibliografía

- [1] J. Glausiusz, "Old trees have much to teach us," <https://www.nature.com/articles/d41586-022-03393-1>, 2022.
- [2] S. Wild, "Africa's majestic baobab trees are mysteriously dying," <https://www.nature.com/articles/d41586-018-05411-7>, 2018.
- [3] H. Miyamoto, K. Uehara, M. Murakawa, H. Sakanashi, H. Nosato, T. Kouyama, and R. Nakamura, "Object detection in satellite imagery using 2-step convolutional neural networks," 2018. [Online]. Available: <https://deepai.org/publication/object-detection-in-satellite-imagery-using-2-step-convolutional-neural-networks>
- [4] M. Radovic, O. Adarkwa, and Q. Wang, "Object recognition in aerial images using convolutional neural networks," 2017. [Online]. Available: <https://www.mdpi.com/2313-433X/3/2/21>
- [5] J. Mabon, M. Ortner, and J. Zerubia, "Learning point processes and convolutional neural networks for object detection in satellite images," 2024. [Online]. Available: <https://www.mdpi.com/2072-4292/16/6/1019>
- [6] K. Bhil, R. Shindihatti, S. Mirza, S. Latkar, Y. S. Ingle, N. F. Shaikh, I. Prabu, and S. N. Pardeshi, "Recent progress in object detection in satellite imagery: A review," Singapore, pp. 209–218, 2022.
- [7] A. Groener, G. Chern, and M. Pritt, "A comparison of deep learning object detection models for satellite imagery," pp. 1–10, 2019.
- [8] W. Guo, W. Yang, H. Zhang, and G. Hua, "Geospatial object detection in high resolution satellite images based on multi-scale convolutional neural network," 2018. [Online]. Available: <https://www.mdpi.com/2072-4292/10/1/131>

Bibliografía

- [9] J. Shermeyer and A. Van Etten, "The effects of super-resolution on object detection performance in satellite imagery," June 2019.
- [10] B. Azam, M. J. Khan, F. A. Bhatti, A. R. M. Maud, S. F. Hussain, A. J. Hashmi, and K. Khurshid, "Aircraft detection in satellite imagery using deep learning-based object detectors," p. 104630, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0141933122001673>
- [11] Y. Koga, H. Miyazaki, and R. Shibasaki, "A method for vehicle detection in high-resolution satellite images that uses a region-based object detector and unsupervised domain adaptation," 2020. [Online]. Available: <https://www.mdpi.com/2072-4292/12/3/575>
- [12] J. Zheng, W. Wu, L. Yu, and H. Fu, "Coconut trees detection on the tenarunga using high-resolution satellite images and deep learning," pp. 6512–6515, 2021.
- [13] J. Zheng, S. Yuan, W. Wu, W. Li, L. Yu, H. Fu, and D. Coomes, "Surveying coconut trees using high-resolution satellite imagery in remote atolls of the pacific ocean," p. 113485, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0034425723000366>
- [14] W. Li, R. Dong, H. Fu, and L. Yu, "Large-scale oil palm tree detection from high-resolution satellite images using two-stage convolutional neural networks," 2019. [Online]. Available: <https://www.mdpi.com/2072-4292/11/1/11>
- [15] A. Khan, U. Khan, M. Waleed, A. Khan, T. Kamal, S. N. K. Marwat, M. Maqsood, and F. Aadil, "Remote sensing: An automated methodology for olive tree detection and counting in satellite images," pp. 77816–77828, 2018.
- [16] G. Liassis and S. Stavrou, "Satellite images analysis for shadow detection and building height estimation," pp. 437–450, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0924271616301939>

- [17] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” 2016.
- [18] R. L. Camino, “Tema 3: Matrices,” <https://www.ugr.es/~rcamino/docencia/geo1-03/g1tema3.pdf>, 2003.
- [19] ——, “Tema 1: Diagonalización de endomorfismos,” <https://www.ugr.es/~7Ercamino/docencia/geometriaII15-16/t-tema1.pdf>, 2016.
- [20] F. J. P. González, “Cálculo diferencial en \mathbb{R}^n ,” Universidad de Granada, Departamento de Análisis matemático, https://www.ugr.es/~fjperez/textos/Calculo_Diferencial_Varias_Variables.pdf, 2016.
- [21] R. P. Albert, “Vector gradiente,” Universidad de Granada, Departamento de Análisis matemático, https://www.ugr.es/~rpaya/documentos/AnalisisI/2022_23/Apuntes_08.pdf, 2022.
- [22] P. R. Román, “Tema 4. probabilidad condicionada: teoremas básicos. independencia de sucesos,” Universidad de Granada, Departamento de Estadística e Investigación Operativa, <https://www.ugr.es/~proman/EDIP/2013-2014/PDF/Tema4.pdf>, 2013.
- [23] E. Bernard, “Introduction to machine learning,” Wolfram Media, Incorporated, <https://books.google.es/books?id=x6K7zgEACAAJ>, 2021.
- [24] S. Russell and P. Norvig, “Artificial intelligence: A modern approach,” Prentice Hall, 3 edition, 2010.
- [25] F. Izaurieta and C. Saavedra, “Redes neuronales artificiales,” Departamento de Física, Universidad de Concepción Chile, 2000.
- [26] P. Isasi-Viñuela, “Redes de neuronas artificiales: un enfoque práctico,” Pearson Education, 2003, ISBN 8420540250.
- [27] F. Berzal, “Redes neuronales and deep learning,” Fernando Berzal, 2018, ISBN 9781731265388.
- [28] R. Salas, “Redes neuronales artificiales,” Universidad de Valparaíso. Departamento de Computación, 1, 2004.
- [29] X. B. Olabe, “Redes neuronales artificiales y sus aplicaciones,” 2008.

Bibliografía

- [30] D. J. Matich, "Ayuda al diagnóstico de tdah en la infancia mediante técnicas de procesado de señal y aprendizaje," 2001.
- [31] S. Grossberg, "Recurrent neural networks," Scholarpedia, 8(2):1888, // www.scholarpedia.org/article/Recurrent_neural_networks, 2013.
- [32] A. J. Ghosh and L. C. Jain, "An overview of radial basis function networks," Physica-Verlag HD, 2001, ISBN 978-3-7908-1826-0.
- [33] J. Mark, "Introduction to radial basis function networks," <https://api.semanticscholar.org/CorpusID:31438158>, 1996.
- [34] C. C. Aggarwal, "Neural networks and deep learning," Springer, 2018, ISBN 978-3-319- 94462-3.
- [35] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications," Acta neurobiologiae experimentalis, 71(4):409–433, 2011.
- [36] A. Grüning and S. M. Bohte, "Spiking neural networks: Principles and challenges," ESANN, 2014.
- [37] K. Y. Trong Huy Phan, "Resolving class imbalance in object detection with weighted cross entropy losses," <https://arxiv.org/abs/2006.01413>, 2020.
- [38] R. G. K. H. P. D. Tsung-Yi Lin, Priya Goyal, "Focal loss for dense object detection," <https://arxiv.org/abs/1708.02002>, 2017.
- [39] R. G. A. F. Joseph Redmon, Santosh Divvala, "You only look once: Unified, real-time object detection," <https://arxiv.org/abs/1506.02640>, 2015.
- [40] T. D. R. Girshick, J. Donahue and J. Malik., "Rich feature hierarchies for accurate object detection and semantic segmentation," CVPR, <https://arxiv.org/abs/1311.2524>, 2014.
- [41] T. G. J. R. Uijlings, K. E. van de Sande and A. W. Smeulders, "Selective search for object recognition," IJCV, <http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>, 2013.

- [42] V. Cortes, Corinna; Vapnik, "Support-vector networks," Machine Learning. 20 (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478, 1995.
- [43] R. Girshick, "Fast r-cnn," <https://arxiv.org/abs/1504.08083>, 2015.
- [44] P. D. R. G. Kaiming He, Georgia Gkioxari, "Mask r-cnn," <https://arxiv.org/abs/1703.06870>, 2017.
- [45] J. of Big Data, "Comparative analysis of deep learning image detection algorithms," 2021. [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00406-8>
- [46] <https://detectron2.readthedocs.io/en/latest/index.html>.
- [47] <https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd>.
- [48] R. G. K. H. B. H. S. B. Tsung-Yi Lin, Piotr Dollar, "Feature pyramid networks for object detection," <https://arxiv.org/abs/1612.03144>, 2016.
- [49] N. B. B. S. R. C. L. S. Davis, "Improving object detection with one line of code," <https://arxiv.org/abs/1704.04503>, 2017.
- [50] <https://cocodataset.org>.