Grado en Ingeniería Informática

Trabajo Fin de Grado

Detección de residuos mediante redes neuronales en entornos de costa y playa.

Leopoldo López Reverón

Tutores: **Agustín Trujillo Pino y Nelson Monzón López**

Las Palmas de Gran Canaria
11 de junio de 2021

Resumen

El océano acumula una enorme cantidad de residuos de distinto tipo que puede permanecer hasta siglos contaminando el mar. Plásticos y otros objetos fabricados por el hombre son en realidad una de las causas más preocupantes de contaminación ambiental del medio marino representando un riesgo para la vida de muchas especies en el mar. Además, y en el mejor de los casos, empobrecen la experiencia y el disfrute de playas y zonas de baño en entornos turísticos provocando incluso, en ocasiones, situaciones peligrosas para la salud de los usuarios.
Canarias, como referente en economía azul, tiene el desafío de mantener sus entornos de playa y costa limpios de residuos y productos contaminantes. En este sentido, debemos ser capaces de identificar el tipo y la cantidad de objetos contaminantes que llegan flotando a nuestras playas y costas facilitando así la organización de rutinas eficientes de recogida los mismos. En este Trabajo Fin de Grado (TFG) se persigue realizar una revisión exhaustiva de la capacidad predictiva en el estado del arte del campo de conocimiento de detección de objetos, específicamente soluciones de redes neuronales en entornos de costa y playa. El objetivo del análisis es identificar aquellos modelos que provean mejores resultados en la difusa decisión de determinar que objeto representa una instancia de un residuo en la línea de playa y la arena.
El trabajo de investigación aquí realizado ha sido en colaboración con la empresa Qualitas Artificial Intelligence And Science[a] S.A.U.(QAISC) que ha apoyado el aprendizaje del estudiante aportando imágenes y vídeos capturados por una cámara PTZ, instalada en el entorno turístico de la playa de Patalavaca que pertenece al municipio de Mogán, localizado en el sur de Gran Canaria, entre las playas de Arguineguín y Anfi del Mar.

[a] www.qaisc.com

Abstract

The ocean accumulates a huge amount of waste of different types that can remain for centuries polluting the sea. Plastics and other objects are actually one of the most worrying causes of environmental pollution of the marine environment, representing a risk to the life of many species in the sea. In addition, and in the best of cases, they impoverish the experience and enjoyment of beaches and bathing areas in tourist environments, sometimes even causing situations that are dangerous to the health of users. The Canary Islands, as a benchmark in the blue economy, has the challenge of keeping its beach and coastline environments clean of waste and polluting products. In this sense, we must be able to identify the type and quantity of polluting objects that float to our beaches and coasts, thus facilitating the organization of efficient collection routines.
This Final Degree Project (TFG) aims to carry out an exhaustive review of the predictive capacity in the state of the art of the field of knowledge of object detection, specifically neural network solutions in environments of coast and beach. The objective of the analysis is to identify those models that provide better results in the fuzzy decision to determine which object represents an instance of a residue on the beach line and the sand.
The research work carried out here has been in collaboration with the company Qualitas Artificial Intelligence And Science [a] SAU (QAISC), which has supported student learning by providing images and videos captured by a PTZ camera, installed in the tourist area of Patalavaca beach that belongs to the municipality of Mogán, located in the south of Gran Canaria, between the beaches of Arguineguín and Anfi del Mar.

[a] www.qaisc.com

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La detección de objetos dentro de una imagen es un área con un especial interés, tanto por la capacidad que le otorga a una máquina, permitiéndole procesar información interpretativa de la escena capturada, como por su aplicación a la resolución de problemas. Tal es así, que la cantidad de artículos científicos, recursos en la red, *frameworks*, es prácticamente inabarcable. Actualmente en esta área la punta de lanza está liderada por sistemas inteligentes, concretamente redes neuronales que cuentan con sistemas internos que permiten la inclusión de información espacial dentro del espacio de inferencia. Entre los cuales destacan, ya las clásicas redes que cuenta parcial o totalmente con capas convolucionales en su arquitectura, y los más recientes *Transformers*, con sus mecanismos de atención que han provocado un cambio disruptivo dentro de la visión por computador.

Las redes neuronales convolucionales no son recientes y ya en 1980 apareció un primer trabajo conocido como *Neocognitrón* de la mano de Fukushima [6] cuyo esquema contaba con varias capas para el reconocimiento de letras. Sin embargo, las redes convolucionales tal y como las conocemos no aparecerían hasta 1998 en el trabajo presentado en LeCun *et al.* [12] bautizada como *LeNet-5* que exponía una arquitectura es sencilla que permite reconocer números escritos a mano.

Las limitaciones de las redes convolucionales son bien conocidas. Por ejemplo, que estas carecen de mecanismos para absorber la información diferenciadora entre las distintas características de los objetos y las características de la imagen, además del problema intrínseco de su funcionamiento, que impide abstraer características aplicables independientemente de la orientación de la imagen (*invariancia rotacional*).

Los problemas anteriormente descritos quedan perfectamente claros con un ejemplo, si tenemos una red convolucional aprendiendo las características de un objeto complejo que se compone a su vez de objetos diferenciables, una cara que contaría con boca, nariz y ojos, la red aprendería cada uno de ellos de forma separada, dando como resultado deducciones erróneas, tal como se aprecia en 1.1.

Aunque la red intentara asociar que si aparecen estos elementos en la imagen seguramente representa una cara, se habrá perdido la información de la orientación y disposición de los elementos básicos que conformarían la cara, pudiendo generar un falso positivo al carecer de las relaciones entre los elementos.

Figura 1.1: Invariancia Rotacional y Jerarquía de Objetos, a la izquierda la imagen original y a la derecha la representación de los posibles aspectos que generan falsos positivos en las redes convolucionales. Imagen extraída de Pinterest [5].

Por otro lado, la aparición de los *Transformers* se fechan en 2017 por la mano de Ashish Vaswan *et. al* [22] y actualmente están a la vanguardia del paradigma de detección de objetos, ocupando los dos primeros puestos en la clasificación del conjunto de datos de *COCO* Tsung-Yi Lin *et al.* [14]. Originalmente estas arquitecturas fueron diseñadas para el campo de conocimiento del procesamiento del lenguaje natural, intentando resolver el problema relacional entre las señales de entrada y salida de las redes convolucionales, dado que estas presentan dificultades a la hora de aprender las dependencias entre posiciones alejadas. Debido a lo cual agregaron una representación de la información que permitiera evaluar el error con el mismo peso para posiciones alejadas, además de mecanismos de atención encargados de representar la información independiente del marco representativo problemático de la secuenciación alineada.

1.1. Objetivos del proyecto

El objetivo de este Trabajo de Fin de Titulo es evaluar las capacidades de distintas redes neuronales para la detección de objetos en entornos de costa y playa. A partir de lo cual, se fijan los siguientes objetivos:

1. Profundizar en los conocimientos adquiridos por el alumno en lenguajes Python y C++.

2. Comprobar la tasa de acierto de redes neuronales para la detección de residuos.

3. Familiarizar al estudiante con estrategias de IA y con sistemas NVIDIA.

4. 4. Elaboración de toda la documentación mínima necesaria en cada una de las fases para su posterior uso incluyendo las fases de pruebas y puesta en producción.

5. Análisis del estado del arte en el campo especifico de la detección de residuos.

1.2. Competencias específicas y aportaciones del trabajo

Durante la realización de este TFG se han cubierto varias competencias estipuladas en el documento oficial del grado en Ingeniería Informática, las cuales se procede a enunciar y justificar.

- CP01, Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para valorar desarrollos tecnológicos relacionados con la informática.

  Cubierta por el análisis del estado del arte de la detección de residuos en ámbitos de costa y playa.

- CP04, Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas.

  Cubierta en el desarrollo del framework para el análisis de los modelos inteligentes del estado del arte

- CP05, Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.

  Cubierta en el uso de modelos de aprendizaje profundo para la detección de residuos.

- CP07, Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen,

incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

Cubierta en la experimentación e implementación de modelos de aprendizaje profundo.

Además, el presente TFG aporta al ámbito de la investigación las siguientes aportaciones:

- Un análisis del estado del arte en detección de objetos.
- Un análisis de las capacidades técnicas de los modelos de aprendizaje profundo en la detección de residuos en ámbitos de playa y costa.
- Una evaluación de los modelos de aprendizaje profundo del estado del arte en detección de objetos.
- Vista previa de la segmentación de escenas para dirigir el uso de los modelos a zonas especificas
- Experimentación con los modelos del estado del arte, comprobando sus capacidades técnicas y dificultades especificas

Capítulo 2

Estado del arte

El aprendizaje profundo (*Deep Learning, DL*) lidera la visión por computador actual, desplazando las técnicas tradicionales para convertirse en el principal foco de atención. Provocando que se concentren tanto recursos monetarios como intelectuales en su desarrollo, convirtiéndola en una tecnología aún más llamativa, dando como resultado la fiebre actual con respecto a esta tecnología. El cambio de paradigma se debe en gran medida a las mejoras en *hardware*, especialmente a las unidades de procesamiento gráfico *GPUs* que permiten la paralelización y potencia de cálculo suficiente para la viabilidad del desarrollo de estas soluciones a diferencia de su implementación en unidades de procesado general *CPUs*.

Además, para el desarrollo de las soluciones se cuenta con plataformas de desarrollo que implementan los recursos necesarios en la creación de los modelos, siendo los mayormente utilizados PyTorch Facebook AI Research lab (FAIR) [16] y Tensorflow Google [1]. Conjuntamente se dispone de *FrameWorks* con soluciones ya entrenadas para una gran variedad de modelos y configuraciones, destacando *Detectron2, Facebook* [24] y *mmDetection, MIT* [4].

Fuera de estos agregadores nos encontramos con soluciones genéricas que se ofrecen para su adaptación al medio concreto en el que van a desempeñar su tarea como *DPT, Intel* René Ranftl *et al.* [19] y YOLO Joseph Redmon *et al.* [20] que requieren de un reajuste final para su puesta a punto. Sin embargo, la carencia de los datos suficientes para realizar este reajuste impide la puesta en producción de soluciones que realmente sean un valor añadido. Por esta limitación nos enfocaremos en determinar que modelos ofrecen los mejores resultados.

El problema de la limitación de datos y la calidad de estos se tiene listado como uno de los principales problemas que se encuentra a la hora de dirigir los modelos a resolver la tarea designada. Pues generar un conjunto de datos en el que se representen la mayoría de las situaciones posibles a las que se enfrentara el modelo, en gran medida no depende de las acciones de los desarrolladores, pues situaciones de baja visibilidad, lluvia, calima, sombras, reflejos, son condiciones

no controlables actualmente. Aun así, con el afán de mejorar el rendimiento se aplican métodos sintéticos que simulan estas condiciones, llamadas comúnmente técnicas de aumento de datos *Data Augmentation* que palían simultáneamente la limitación de datos generado más y las condiciones representadas en el conjunto, a costa de la calidad de los datos introducidos.

La adición de los datos sintéticos aumentara el entendimiento que tendrán los modelos de aprendizaje automático, permitiéndole tener una mayor capacidad para discernir en aquellos casos límites de su entrenamiento. Por lo que el análisis de que técnicas deben ser utilizadas es un paso de vital importancia. Por ejemplo en la figura 2.1 se ilustra la rotación aplicada a un conjunto de números, en especial el número 6, con una rotación de 90^{o} pierde su significado como número y una de 180^{o} lo conlleva a solaparse con otra clase distinta el 9.

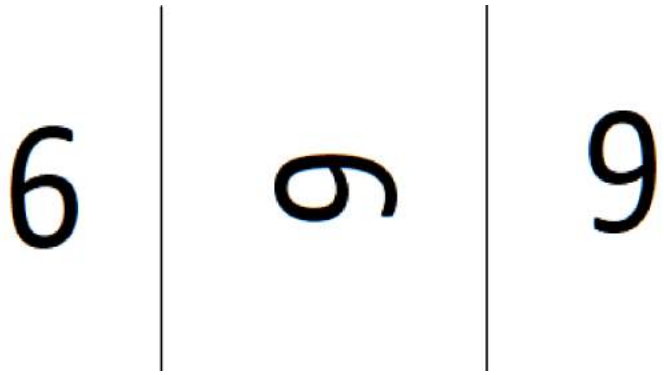


Figura 2.1: Transformación del significado de una instancia, a la izquierda la imagen original, en el centro la imagen rotada 90^{o} donde empieza a ser discutible su significado y a la derecha la imagen rotada 180^{o} donde el significado de la imagen ha sido totalmente transformado. Imagen de creación propia

2.1. Visión por Computador mediante redes neuronales

Las imágenes pueden ser procesadas de distintas maneras, para resaltar el contenido relevante a la tarea designada. En concreto la detección de residuos en ambientes costeros tiene una gran variabilidad en cuanto a los acercamientos posibles, teniendo distintas vertientes:

2.1.1. Detección de instancias

Principalmente en esta aproximación se intenta construir un sistema capaz de delimitar el marco de información que representa la clase a detectar, asignándole la clase que identifique correctamente de que objeto se trata. De acuerdo con lo cual, se crean marcos y etiquetas de clase diferentes para cada objeto detectado.

El sistema inteligente que lleva a cabo esta tarea inicialmente carece de la especificidad necesaria para llevar a cabo su tarea en condiciones óptimas, ya que se recurre a modelos previamente entrenados en grandes conjuntos de datos de uso general, entre los cuales se destaca COCO Tsung-Yi Lin *et al.* [14], utilizado comúnmente como conjunto de datos de referencia.

Normalmente la representación de la información en el esquema de la detección de instancias se corresponde con la adición de un recuadro delimitador del objeto y su clase, tal como se aprecia en la figura 2.2.

Figura 2.2: Esquema básico de la detección de una instancia, a la izquierda la imagen original y a la derecha la detección. Imagen de creación propia.

2.1.2. Segmentación de imagen

En este paradigma se opta por un tratamiento a nivel de píxel dentro de la imagen, otorgándole a cada uno su clase correspondiente. Su principal ventaja es también su mayor debilidad, su definición de clases con la unidad mínima de representación, permite generar mascaras que solamente delimiten aquellos conjuntos de píxeles que representan el objeto. Sin embargo, su flexibilidad conlleva la creación de conjuntos de datos con mayor especificidad y precisión, dificultando enormemente la creación de conjuntos y sumándole la cantidad mayor de tiempo

empleada para la asignación de clases, crear un amplio repertorio de escenas y situaciones queda en manos de la comunidad.

Su representación más común es la superposición de la máscara al objeto que delimita, y por separado el significado correspondiente a cada color superpuesto, como se puede apreciar en la figura 2.3.

Figura 2.3: Esquema básico de la segmentación, a la izquierda la imagen original y a la derecha la segmentación superpuesta. Imagen extraída del conjunto TACO [17].

2.2. Paradigmas específicos de la detección de residuos

Dentro del marco de acción de la detección de residuos se diferencian las siguientes ramas específicas de interés:

2.2.1. Detección de materiales

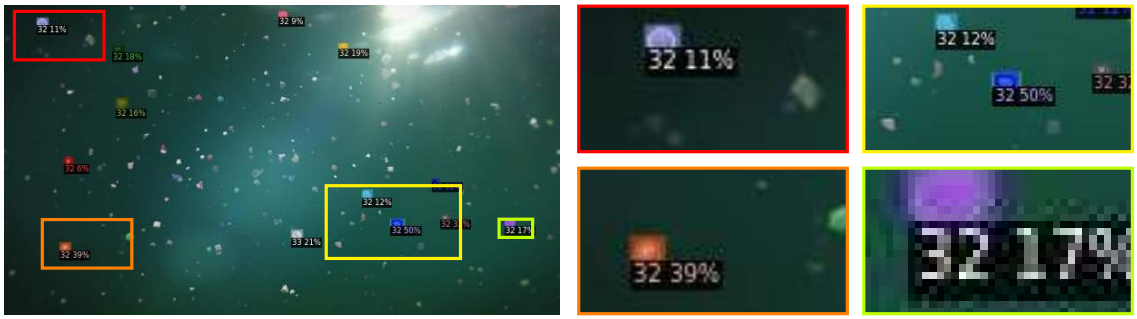
Se enfoca en disociar las clases de los objetos, para centrarse únicamente en los materiales que los componen, otorgándole el valor a la diferenciación y estimación de la presencia de estos en entornos costeros.

Se han intentado distintos acercamientos como TrashNet Yuheng Wang *et al.* [23] o AquaVision Panwar *et al.* [15], en la proposición de una solución satisfactoria, sin llegar a terminar de resolver el problema.

2.2.2. Detección de micro plásticos

Este tipo de estrategias requieren de un análisis previo del entorno costero, pues se necesita de una infraestructura capaz de obtener las imágenes necesarias para la optimización del modelo, dado que ningún sistema es capaz de resolver un problema que no ve, daremos por solucionado la obtención de información y la precisión con la que se adquiere.

Además se expone un problema concreto al propio enfoque, en el que dos objetos aparentemente iguales de color, tamaño y forma tienen una composición química diferente, dificultando enormemente la capacidad del sistema para resolver el problema, porque se necesitaría de comprobar que partículas se han detectado. Vemos un ejemplo en la figura 2.4.



(a) (b)

Figura 2.4: Detección de micro plásticos, a la izquierda la imagen original y a la derecha aumentos sobre las detecciones. Imagen extraída de istockphoto [10].

2.3. Arquitecturas relevantes en la detección de características

La variedad de modelos y el interés que estos suscitan han propiciado la aparición de estructuras dentro de los modelos que llevan a cabo una función específica, aumentando las capacidades y complejidades de las arquitecturas. A continuación, se describen las estructuras especializadas de los modelos, las arquitecturas de los modelos y los conjuntos de datos utilizados.

2.3.1. Estructuras Internas

Estructuras especializadas que permiten la diferenciación de las arquitecturas, pues son el mecanismo a partir del cual se construyen los modelos intentando potenciar y mejorar sus resultados. A continuación, se procede a describirlos.

- Capa convolucional

  Las capas convolucionales constituyen un mecanismo capaz de extraer características dentro de un conjunto de datos, normalmente estos datos son imágenes, en los que se infiere la información espacial representativa que considera el aprendizaje automático como elemento clave en la realización de su tarea.

  El proceso que realiza se denomina convolución, donde se recorre iterativamente la imagen con una ventana de acción, llamada *Kernel* que realiza una multiplicación de matrices con el segmento correspondiente de la imagen, tal y como se representa en la figura 2.5.

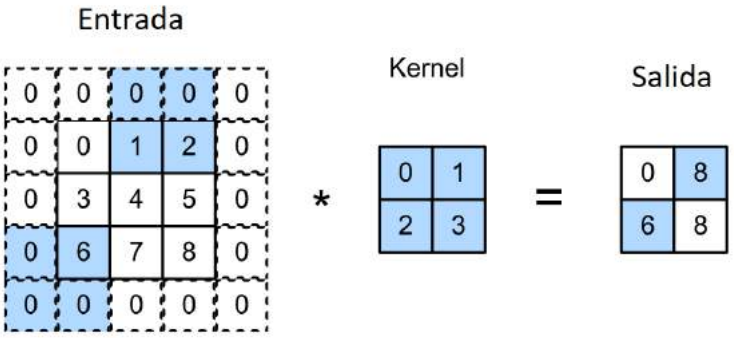


Figura 2.5: Visualización del proceso de convolución, donde se aplica la operación con el kernel obteniendo su correspondiente salida. Imagen de creación propia.

- Autocodificador (*Autoencoder*)

  El autocodificador se compone de tres partes diferenciales que le otorgan sus particulares propiedades teniendo el esquema representado en la figura 2.6.

  A continuación, se procede a describir cada una de las partes.

  - Codificador

El codificador se encarga de recibir la entrada de datos e intentar mediante la búsqueda de características la compresión de la información, prevaleciendo aquellos cúmulos de información con mayor explicabilidad de la imagen.

- Decodificador

  El decodificador es la parte final de autocodificador, que a partir de la información que le ofrece como input el codificador, intenta reconstruir con la mayor precisión posible la imagen comprimida por el codificador.

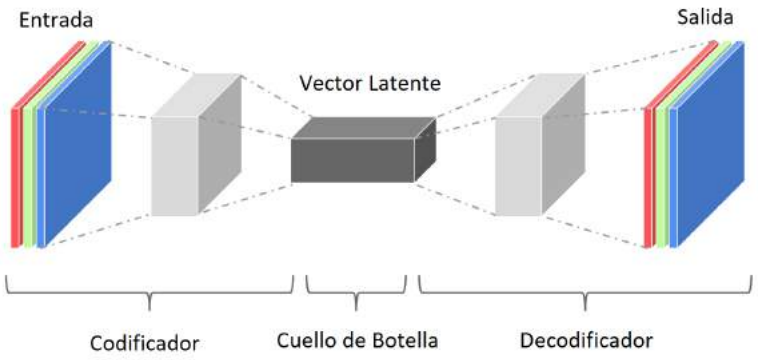


Figura 2.6: Estructura autocodificador, Imagen extraída de hackernoon [7].

- Vector Latente

  Se trata de un lugar de especial interés, pues es normalmente el vector que representa la imagen comprimida al máximo que puede la configuración del autocodificador, que por las propiedades matemáticas inherentes a la estructura podemos realizar operaciones con él. Algunos ejemplos de las capacidades del autocodificador son realizar operaciones aritméticas con la información inferida, tal cual se observa en la figura 2.7 y la posibilidad de explorar las distintas interpolaciones dentro del espacio inferido como se representa en la figura 2.8.

Figura 2.7: Posibilidades aritméticas en el vector latente, donde se realizan operaciones del elemento conceptual inferido por el autocodificador. Imagen extraída de hackernoon [7].

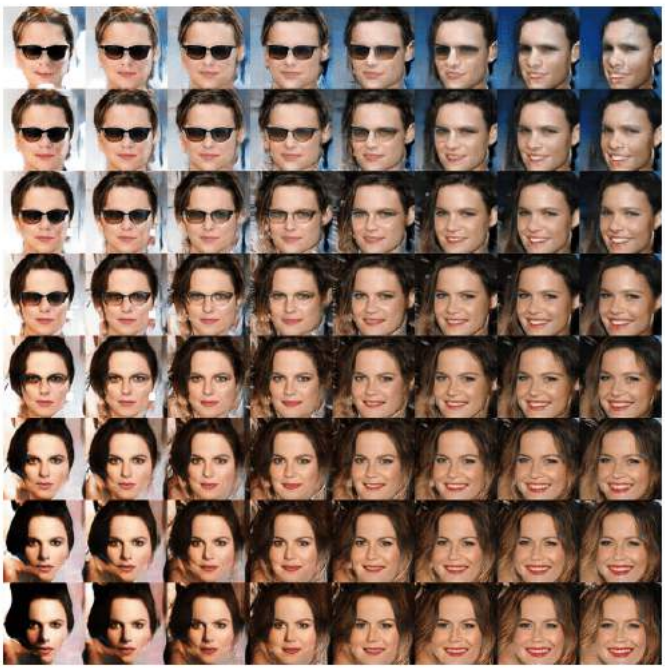


Figura 2.8: Interpolación a través del espacio de representación inferido por el autocodificador, donde se aprecia las imágenes intermedias creadas para llegar de una a otra. Imagen extraída de hackernoon [7].

- *Embeddings*

  Los *embeddings* son una aproximación diferente para la representación de las clases que se van a definir en el modelo, en este cambio de representación se opta por definir cada clase con una dimensión propia, en vez de usar la representación categórica, de tal modo que quedaría con la forma representada en la figura 2.9.

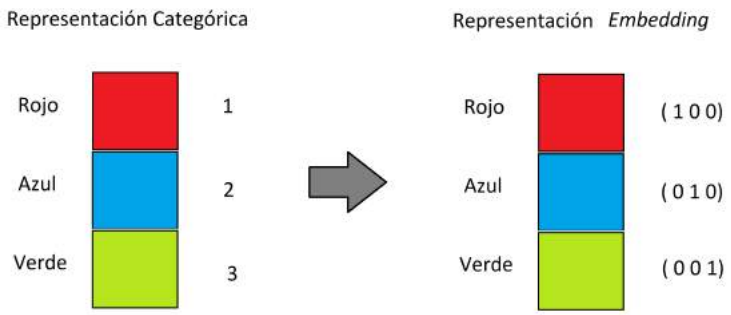


Figura 2.9: Diferencias entre la representación Categórica y la representación en los *Embedding* de las clases, Imagen de creación propia.

La codificación descrita para los *Embedding* se denomina *One-Hot-Encoding* y ofrece dos propiedades realmente interesantes para la estructuración de la información inferida por la red.

1. Cálculo del error cometido por el modelo

   A la hora de determinar el error de la clase predicha por el modelo y la real, como se vislumbra en la figura 2.10, en la representación categórica se comete un desbalanceo del error, ya que el error se calcula dependiendo de la distancia a la que se encuentren la clase predicha de la real, mientras que por las propiedades de la codificación *One-Hot-Encoding* en los *embeddings* el error entre la clase real y la predicha se diferencian en un único valor.

2. Representación espacial de las clases

   En la codificación *One-Hot-Encoding* de los *embeddings* cada clase puede interpretarse como una dimensión al estilo de la figura 2.11, dado lo cual implica que el modelo dispone de un mecanismo que le permite no solamente tener una dimensión independiente por cada clase, sino también, decidir que clases tienen una dimensión con mayor cercanía.

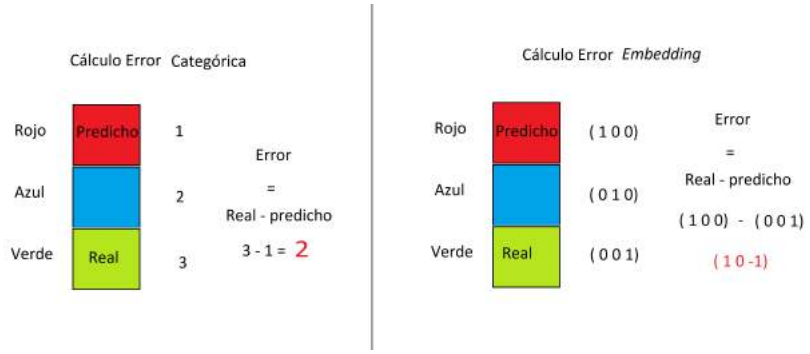


Figura 2.10: Diferencias del cálculo del error. Imagen de creación propia.

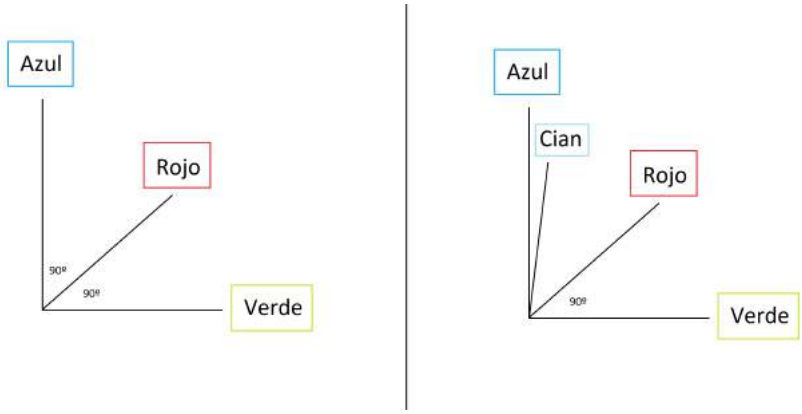


Figura 2.11: Representación del espacio de los *Embedding* en la disposición de las dimensiones. Imagen de creación propia.

2.4. Arquitectura de Modelos

La inversión en el desarrollo e investigación de los sistemas de aprendizaje automático ha permitido la definición y especialización de arquitecturas adaptadas a la resolución de los distintos paradigmas. A continuación, se detallarán los modelos de mayor interés del estado del arte, además de sus capacidades singulares.

2.4.1. Arquitectura de Detección

- YOLO (*You Only Look Once*)

  La característica diferenciadora de YOLO (trabajo presentado en Joseph Redmon *et al.* [20]) reside en su forma de generar las cajas delimitadoras de las detecciones, tal como se observa en la figura 2.12. Dicha metodología consiste en la división de la imagen en diversas secciones, donde posterior-

mente se generan las cajas delimitadoras que podrían contener el objeto, dejando aquellas que presenten una mayor certeza. Dichas cajas conforman un parámetro definible llamado *anchors*

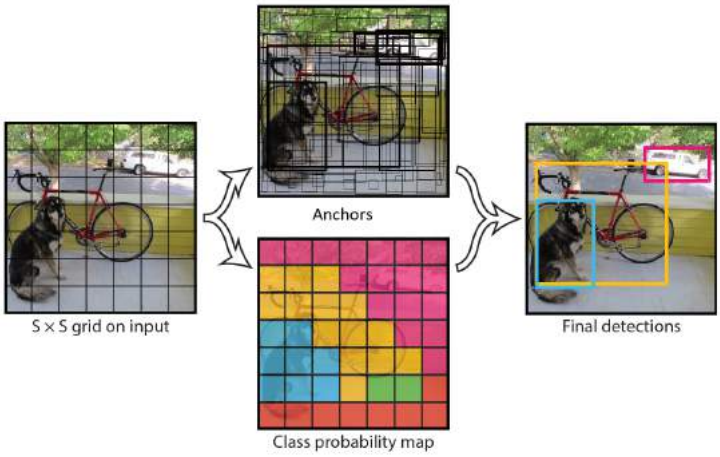


Figura 2.12: Flujo de trabajo de YOLO. Imagen extraída del artículo [20].

El éxito obtenido por esta arquitectura ha supuesto que sus autores hayan continuado mejorando el modelo en distintas versiones. La versión 4 de Yolo se representa en la figura 2.13 dónde vemos que dispone de un gran repertorio de elementos intercambiables lo que se traduce en la posibilidad de que la red sepa adaptarse mejor ante distintas circunstancias.

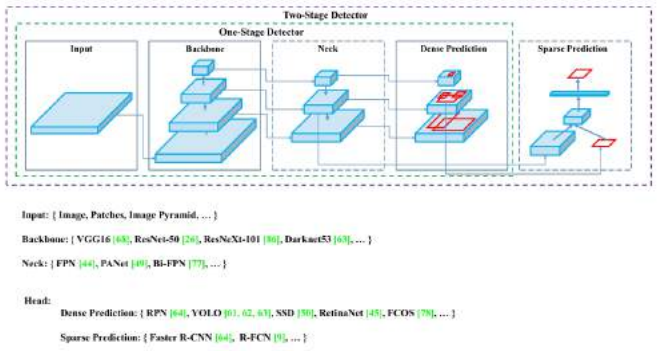


Figura 2.13: Esquema de la Arquitectura de YOLOv4 y la especificación de las piezas intercambiables, Imagen extraída del paper de YOLOv4 Alexey Bochkovskiy *et al.* [3].

- Faster R-CNN *(Region based - Convolutional Neural Network)*

  En el articulo Shaoqing Ren *et al.* [21] se define Faster R-CNN como una estructura en la que se diferencian dos pasos, tal como se muestra en la figura 2.14. Primero aplica la red en busca de las zonas con mayor probabilidad de convertirse en zonas de activación, tarea acometida por *Region Proposal Network* Shaoqing Ren *et al.* [21]. Y segundo se procede a la búsqueda de objetos en dichas zonas interesantes.

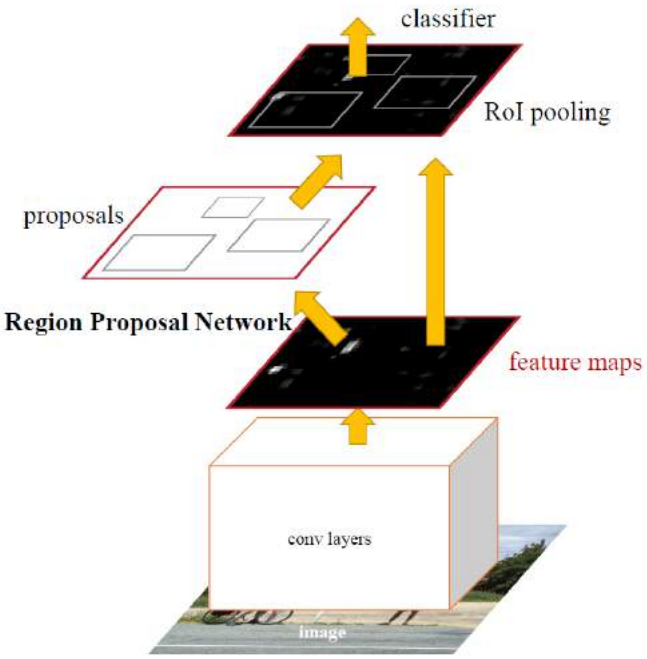


Figura 2.14: Representación arquitectura Faster R-CNN, Imagen extraída del paper de Faster R-CNN Shaoqing Ren *et al.* [21].

- FPN *(Feature Pyramid Network)*

  En el paper Tsung-Yi Lin *et al.* [13] se define FPN como un sistema para la detección de objetos a diferentes escalas como se observa en la figura 2.15, habitualmente se utiliza como un extractor de características. Su funcionalidad es una aproximación distinta a los *anchors* que se utilizan en YOLO [20].

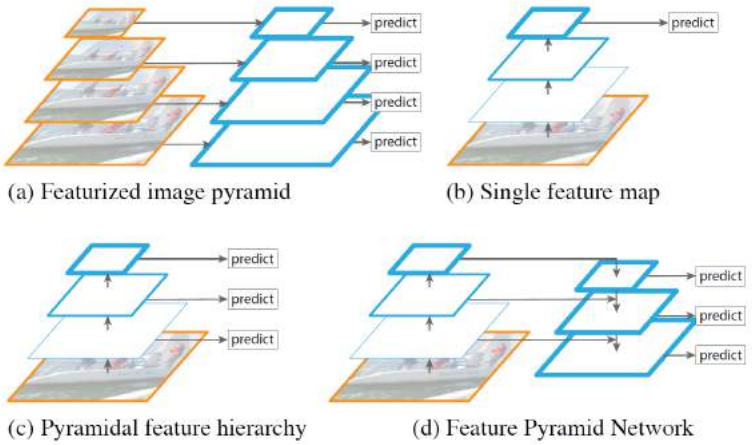


Figura 2.15: Representación de la arquitectura FPN, Imagen extraída del paper de FPN Tsung-Yi Lin *et al.* [13].

2.4.2. Arquitecturas de Segmentación

- Mask R-CNN *(Mask Region based - Convolutional Neural Network)*

  El trabajo presentado en Kaiming He *et al.* [8] es una extensión de la red Faster R-CNN [21], a la cual se le añade una rama para la predicción de la máscara dentro de la caja del objeto detectado, la cual se ejecuta en paralelo, teniendo el esquema representado en la figura 2.16.

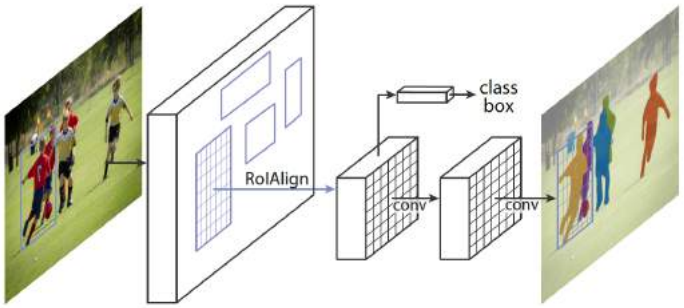


Figura 2.16: Representación de la rama de segmentación de Mask R-CNN. Imagen extraída de la publicación original (He *et al.* [8].)

- ResNet *(Residual neural Network)*

  En Kaiming He *et al.* [9] se propone añadir conexiones de salto (*Skip Connections*) al estilo de la figura 2.17, como respuesta al problema de la perdida de información posicional de los objetos, posibilitándole a la red la capacidad de decidir qué información conserva inalterada a través del proceso de inferencia.

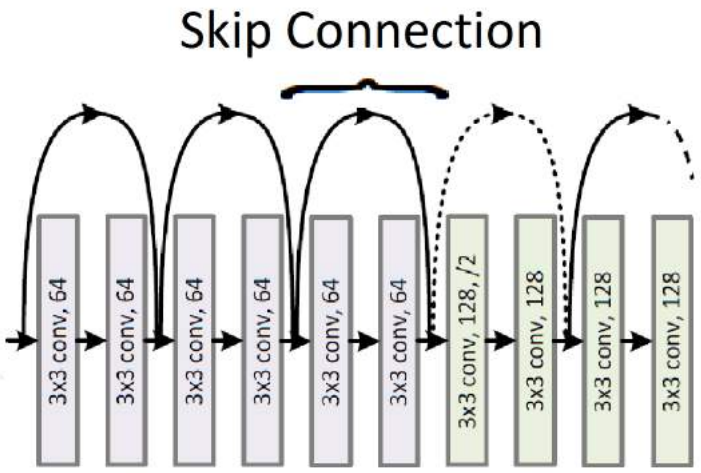


Figura 2.17: Representación de las (*Skip Connection*) en la arquitectura ResNet. Imagen extraída de paper de Residual neural Network Kaiming He *et al.* [9].

- DPT *(Dense Prediction Transformer)*

  El trabajo presentado en René Ranftl *et al.* [18] presentó una arquitectura basada en *Transformers*, que actualmente ocupa los primeros puestos en los conjuntos de datos de segmentación semántica, teniendo el esquema representado en la figura 2.18. Su característica diferenciadora reside en el uso de una red convolucional como decodificador.

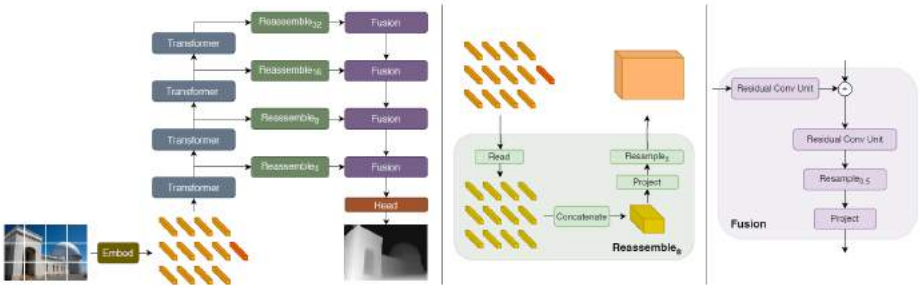


Figura 2.18: Representación Arquitectura DPT. Imagen extraída del paper de Dense Prediction Transformer René Ranftl *et al.* [18].

2.5. Conjuntos de datos

2.5.1. COCO (*Common Object in COntext*)

COCO es uno de los conjuntos de datos más populares para la evaluación de los modelos de aprendizaje profundo. Actualmente cuenta con más de 300.000 imágenes divididas en 80 clases de objetos.

2.5.2. TACO (*Trash Annotations in COntext*)

TACO es un conjunto de datos enfocado en las anotaciones de residuos en contexto, específicamente provee de anotaciones semánticas y de la posición de los objetos con sus respectivas etiquetas de clase. Actualmente el conjunto está en crecimiento y cuenta con 1500 imágenes etiquetadas de gran resolución.

Capítulo 3

Recursos utilizados y Presupuesto

Durante el desarrollo del presente TFG se ha hecho uso de numerosos recursos, por lo que se procede a enumerarlos y estimar los costes vinculados a la realización del mismo.

3.1. Recursos utilizados

Primero se enumeraran los recursos utilizados distinguiendo entre recursos hardware y software.

3.1.1. Recursos Hardware

- Cámaras PTZ (*Pan-Tilt-Zoom*)

  Utilizadas en la obtención de imágenes que representan escenas reales de entornos de costa y playa.

- Servidor de trabajo

  Utilizado como espacio unificador de trabajo en la oficina de QAISC y para el trabajo en remoto de su personal, se ha utilizado un servidor DELL T440 que cuenta con un dual Intel Xeon de 24 núcleos conformando un total de 48 núcleos, 128 Gb de Ram DDR4 y cuenta con el sistema de virtualización de KVM el cual simplifica el traslado de maquinas entre sistemas físicos.

- Ordenador portátil

  Utilizado como punto de acceso al espacio de trabajo desde remoto.

3.1.2. Recursos Software

- ONVIF (*Open Network Video Interface Forum*)

  Protocolo utilizado en la comunicación de las cámaras PTZ, en el cual se unifican las diferencias de implementación en la comunicación de cámaras de los distintos fabricantes. Permitiendo el desarrollo de aplicaciones independientes del periférico.

- Tensorflow

  Framework de desarrollo de Google para python, que permite la abstracción de las partes de los modelos acelerando su desarrollo, además de ofrecer métricas de medición de rendimiento y optimizaciones en la ejecución de funciones.

- PyTorch

  Framework de desarrollo de código abierto mantenido por Facebook, que ofrece una interfaz de alto nivel en la gestión del desarrollo de modelos de deep learning en python.

- Darknet

  Framework utilizado para el entrenamiento de YOLOv4.

- Framework utilizados en la evaluacion del estado del arte: Detectron2, mmDetection

- Lenguajes de programación: C++, C, Python 3.8

3.2. Presupuesto

A continuación, se realizara una estimación de los costes de este TFG. En el desarrollo del proyecto no se ha utilizado ningún software de subscripción, por lo que se presupuestaran los recursos humanos y costes hardware.

Costes de personal

A continuación, se desglosa la estimación para las tareas realizadas durante el TFG, donde se ha estimado el coste por hora a 18.75€.

Tarea	Horas	Coste
Familiarización con el problema a resolver	30	562,5€
Estudio de las herramientas a emplear	20	375€
Análisis del estado del arte	80	1500€
Evaluación de los modelos del estado del arte	30	562,5€
Segmentación de escenas	35	656,5€
Experimentación con el ajuste de modelos	40	750€
Reuniones a la semana	15	281,25 €
Memoria de las tareas realizas	50	937,5€
TOTAL		5625,25€

Costes Hardware

A continuación, se desglosan los componentes utilizados.

Componente	Precio
Camara PTZ	2900€
Portatil	275€
Servidor DELL	10100€
TOTAL	13275€

Coste Total

Recurso	Cantidad
Costes de personal	5625,25€
Costes de Hardware	13275€
TOTAL	18900,25€

Capítulo 4

Plan de trabajo y temporización

Para la gestión de en la planificación del desarrollo de los objetivos propuestos se ha utilizado la metodología de desarrollo ágil SCRUM, donde se han tenido reuniones semanales tanto con sus directores como con otros miembros de la empresa QAISC para el seguimiento del desarrollo de los objetivos planteados, además de proponerse soluciones a los problemas encontrados durante el desarrollo.

La gestión de tareas se ha realizado mediante la aplicación Trello, donde se definen listas que definen el estado de las tareas concretas para cumplir los objetivos, las cuales tendrán un recorrido a través de las listas hasta su finalización.

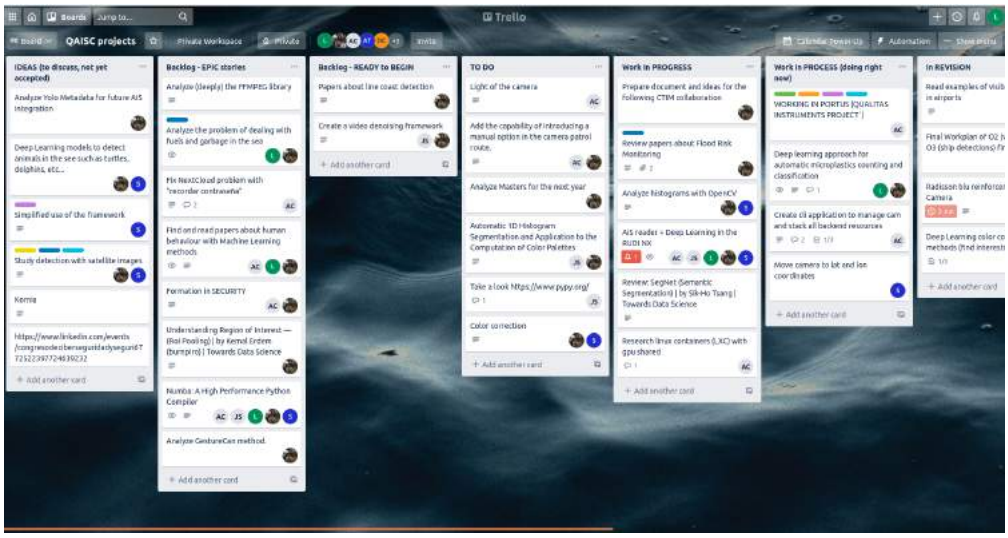


Figura 4.1: Pantalla de ejemplo del tablero empleado en QAISC para la gestión de los trabajos de su equipo.

A continuación, se muestra el esquema utilizado para la planificación de las tareas del proyecto que ha sido una continuación del periodo de prácticas del

estudiante en QAISC.

Fases	Duración Estimada (300 horas)	Tareas
Creación de una primera pila de producto	20	Primer acercamiento al entorno descrito por la cámara y selección de escenas (Estimado 10 horas). Maquetación de las primeras historias de usuario y priorización de necesidades. (Estimado 10 horas).
Sprint Zero	60	Estudio de las herramientas a emplear y familiarización con el problema (estimado 20 horas). Análisis de la literatura del estado del arte en detección de residuos en entornos de playa y costa (estimado 40 horas).
Desarrollo de la pila de producto	10	Maquetar la pila de producto esperada para el desarrollo de este TFG.
Release1 (Sprint 1)	20	Reutilizando un sistema de pruebas (pipeline) desarrollado durante el periodo de prácticas del estudiante, se realizarán las modificaciones pertinentes para el estudio de las secuencias que se juzguen interesantes para el estudio que nos ocupa.
Release2 (Sprint2, 3 y 4)	60	Experimentación de las redes neuronales de interés de incorporación dentro del pipeline en el sistema NVIDIA.
Release3 (Sprint 5, 6, 7 y 8)	80	Desarrollo de un prototipo que integre el conocimiento adquirido dentro de la rutina de trabajo de la cámara PTZ.
Documentación / Presentación	50	Redacción de la memoria y la presentación acorde a los reglamentos de TFT de la EII

Cuadro 4.1: Planificación temporal de trabajo

Sin embargo, a lo largo del desarrollo de proyecto se realizaron las siguientes modificaciones:

En primer lugar, debido a los resultados obtenidos en la evaluación de los modelos se decidió realizar un entrenamiento, para obtener un punto de comparación frente a un sistema entrenado para el problema, aumentan do el tiempo requerido para su finalización.

En segundo lugar, en busca de mejorar el proceso de detecciones, se añadió la segmentación de escenas en busca de determinar las áreas con mayor interés en la monitorización de residuos.

Finalmente, el desarrollo de las nuevas tareas elevaron la carga de trabajo dificultando la incorporación de los modelos dentro del sistema NVIDIA y dentro de la rutina de la cámara PTZ las cuales se encuentran en un avanzado estado de desarrollo.

Capítulo 5

Resultados

El objetivo de este TFT es determinar la capacidad de varias redes neuronales en la detección de objetos (residuos) en el ámbito de playa y costa. Para ello, existen herramientas que aglomeran distintos modelos, además de aportar la posibilidad de utilizarlos ya entrenados, entrenar los propios y su evaluación, por lo que una vez terminada la exploración se realizara la evaluación mediante las métricas definidas.

Para cumplir el objetivo se utilizaron las siguientes herramientas: mmDetection Chen Kai *et al.* [4], Detectron2 Yuxin Wu *et al.* [24] y INTEL ISL DPT (René Ranftl *et al.* [18]. En cuanto a los datos de pruebas utilizados se ha extraído un subconjunto de imagen interesantes del conjunto de datos TACO (Pedro F. Proença *et al.* [17].

5.1. mmDetection, modelos de detección

El proyecto mmDetection (presentado en Kai *et al.* [4]) se lleva a cabo por personal del *Massachusetts Institute of Technology* (MIT) y recoge la mayoría de los modelos de redes neuronales para la detección de objetos. Además, este framework es parte de un proyecto mayor llamado OpenMMlab, que aparte de la detección, ofrece repositorios de modelos de segmentación, clasificación, e incluso su propia librería de aceleración de cálculos.

Para su uso, el framework ofrece una API, mediante la cual podemos elegir el modelo y su correspondiente configuración, además de permitir su modificación. Los modelos preentrenados que ofrece han sido entrenados en COCO con PyTorch *Facebook AI Research lab* (FAIR) [16] y CAFFE *Berkeley AI Research* (BAIR) [11].

Dada la abrumadora cantidad de modelos y pesos preentrenados que ofrece el framework, se decidió acotar el estudio a los modelos que ofrecían mejores resultados al ser evaluados en COCO Tsung-Yi Lin *et al.* [14].

El listado de modelos escogidos de mmDetection para la experimentación es el siguiente:

Modelos	backbone	method	Framework
Faster rcnn	R50	DC5	CAFFE
RPN	X101	64x4d FPN	Pytorch

Cuadro 5.1: Lista de modelos evaluados de mmDetection

Resultados de los modelos de mmDetection

A continuación, se muestran los resultados de los modelos de mmDetection, en ámbitos de costa y playa. Para probar los modelos se realizaron pruebas con imágenes en condiciones que dificultaran la identificación, por ejemplo, oclusiones con sombras, objetos parcialmente enterrados y objetos flotando en el agua.

Faster rcnn R50 DC5

En la comparación 5.1 se puede apreciar que no se detecta ningún pájaro, y solo se detectan algunas personas sin una gran consistencia entre ellas, pues aún estando a distancias similares, la sombra de los árboles modifica la imagen lo suficiente como para que dejen de ser detectadas.

Ground Truth

Inference

Figura 5.1: Ejemplo de la evaluación de Faster R50 DC5. A la izquierda la imagen original y a la derecha la inferencia

Como se puede observar no detecta ningún pato, aunque realmente son objetos diferenciables de la superficie del agua. Además de la cantidad de personas que deja sin detectar, tal como se ve en los aumentos en la figura 5.2.

(a) (b)

Figura 5.2: Zoom zonas de interés evaluación Faster R50 DC5

En el caso de objetos parcialmente enterrados el modelo tiene grandes dificultades para la detección, dejando incluso objetos en totalmente visibles sin detectar, como se aprecia en la figura 5.3.

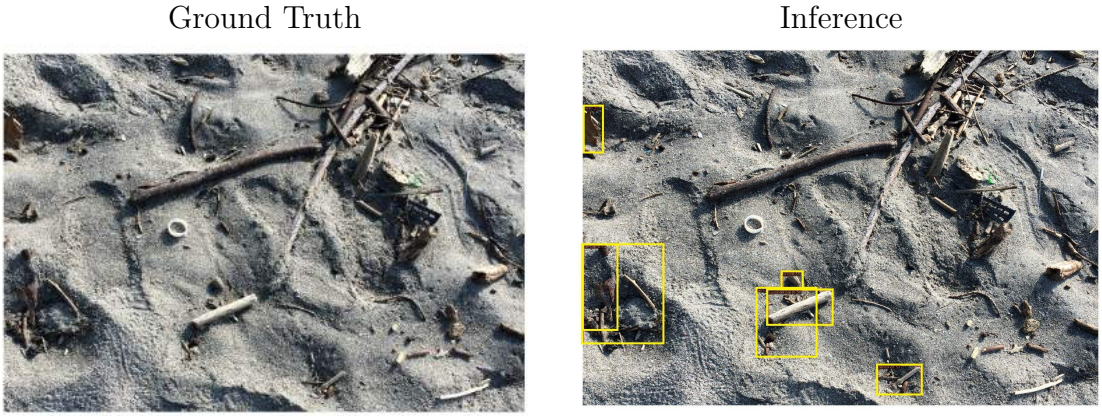


Figura 5.3: Ejemplo de la evaluación de Faster R50 DC5 en objetos ocultados por arena. A la izquierda la imagen original y a la derecha la inferencia

En específico se observan casos interesantes como la detección de objetos al borde de la imagen, y la no detección de objetos con discontinuidad en su representación, como en el caso de la tapa de botella parcialmente ocluida por la arena, como se ve en la figura 5.4.

(a) (b)

Figura 5.4: Zoom zonas de interés evaluación objetos ocultados por arena Faster R50 DC5

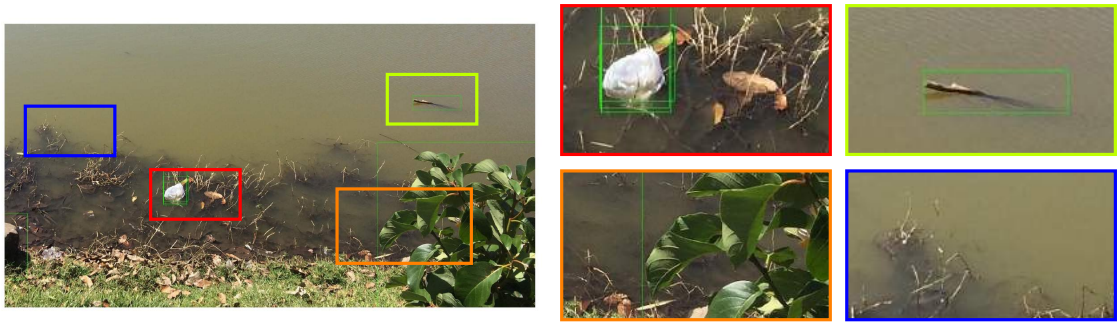
RPN x101 64x4d FPN

El modelo RPN x101 64x4d FPN ofrece resultados más prometedores, aunque ahora se aprecia una mayor sensibilidad para la detección de objetos, aparecen problemas asociados a ella, como se observa en la comparativa 5.5.

Ground Truth Inference

Figura 5.5: Ejemplo de la evaluación de RPN x101 64 RPN. A la izquierda la imagen original y a la derecha la inferencia

En el caso de detección de objetos flotantes se aprecia una mayor cantidad de detecciones. Sin embargo, aparecen múltiples detecciones para el mismo objeto y se continua con la dificultad de detección de objetos sumergidos, quedando estos hechos visibles en la figura 5.6.



(a) (b)

Figura 5.6: Zoom zonas de interés evaluación objetos flotantes RPN x101 64 RPN.

Para los objetos parcialmente enterrados de forma similar se incrementa el número de objetos detectados a costa de detectarlos múltiples veces, como se aprecia en la comparativa 5.7.

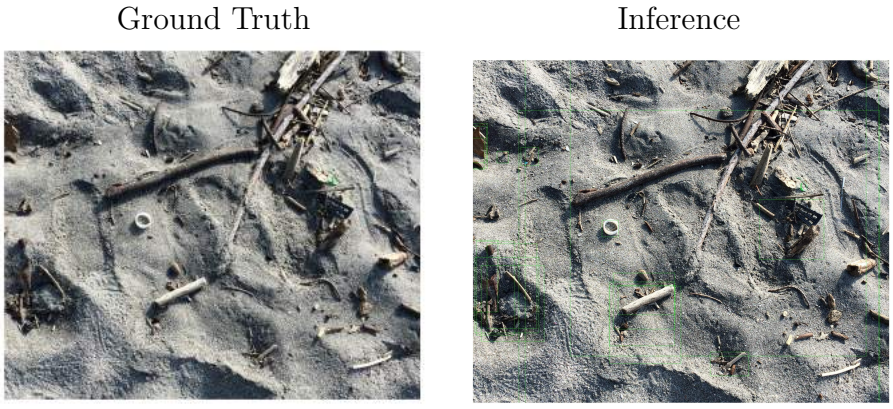


Figura 5.7: Ejemplo de la evaluación de RPN x101 64 RPN en objetos oculfiguretados por arena. A la izquierda la imagen original y a la derecha la inferencia

En este caso particular el modelo si es capaz de detectar algunos de estos objetos parcialmente ocluidos, como se puede observar en el caso de la tapa de botella. Sin embargo, se aumentan los casos de falsos positivos como revela el aumento en azul en la figura 5.8.

(a) (b)

Figura 5.8: Zoom zonas de interés evaluación objetos ocluidos por arena RPN x101 64 RPN.

5.2. Detectron2, modelos de detección

Tras estos resultados se decidió explorar otros frameworks siendo finalmente escogido Detectron2 [24] un proyecto desarrollado por Facebook, que ofrece el soporte para la inferencia, entrenamiento, uso de modelos preentrenados, e incluso la modificación de la configuración de los modelos. Los modelos preentrenados usan como base PyTorch *Facebook AI Research lab* (FAIR) [16] y han sido entrenados en COCO Tsung-Yi Lin *et al.* [14].

Dentro de todas las opciones que ofrece Detectron2 [24] se exploraron los siguientes modelos:

Modelos	backbone	method
Faster rcnn	R50	C4
Faster rcnn	R50	DC5
Faster rcnn	X101	32x8d FPN
Keypoint rcnn	R101	FPN
Mask rcnn	R50	C4
Mask rcnn	R50	DC5
Mask rcnn	X101	32x8d FPN
Retinanet	R101	FPN

Cuadro 5.2: Lista de modelos evaluados de Detectron2

Faster rcnn R50 C4

Como se observa existe una mejoría respecto a los modelos probados en mmDetection Chen, Kai *et al.* [4], incluso se detectan objetos de tamaños variados, aunque se vislumbra un nuevo elemento para tener en cuenta, la detección de las sombras de las huellas como objetos, como se muestra en la comparativa 5.9.

Ground Truth Inference

Figura 5.9: Ejemplo de la evaluación de Faster R50 C4 en objetos ocultados por arena. A la izquierda la imagen original y a la derecha la inferencia

En más detalle se destaca la detección de sombras del zoom en verde y la inespecificidad de las detecciones en el zoom azul de la figura 5.10.

(a) (b)

Figura 5.10: Zoom zonas de interés evaluación objetos ocluidos por arena Faster R50 C4.

Faster rcnn R50 DC5

Ante este cambio repentino de resultados y existiendo tanto la posibilidad del error humano en la exploración de mmDetection Chen, Kai *et al.* [4], como la disponibilidad del mismo modelo en Detectron2 [24]. Se decide comparar el mismo modelo extraído de distintos framework en las mismas condiciones. Teniendo el resultado mostrado en la comparativa 5.11.

Detectron2 mmDetection

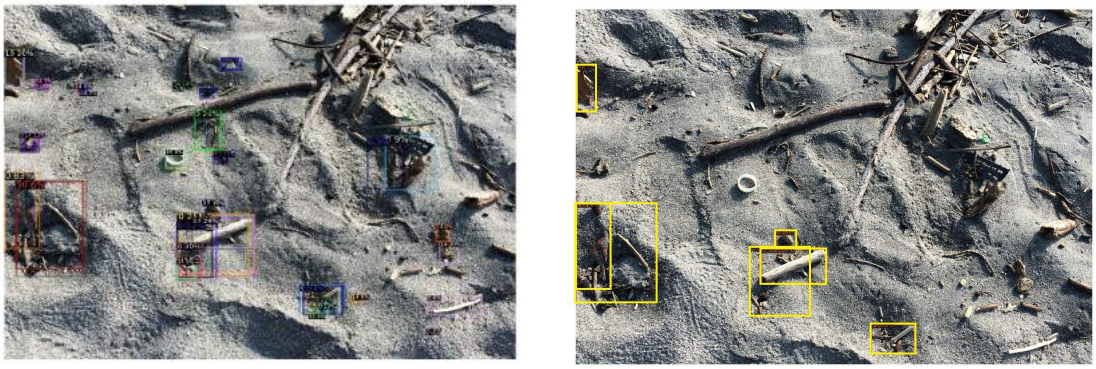


Figura 5.11: Comparación entre Detectron2 y mmDetection en objetos ocultados por arena. A la izquierda Detectron2 y a la derecha mmDetection

Como se observa en la comparativa 5.11, a pesar de ser el mismo modelo, misma imagen. La diferencia del entrenamiento realizado le permite al modelo de Detectron2 recoger las detecciones de mmDetection, además de añadir los objetos pequeños presentes en la imagen.

Faster rcnn X101 32x8d FPN

En base a esta diferencia entre modelos dependiente del entrenamiento. Se propone comprobar las capacidades del modelo que presenta mejores métricas, obteniendo el resultado de la comparativa 5.12.

Detectron2 mmDetection

Figura 5.12: Comparación entre Detectron2 y mmDetection en objetos flotantes. A la izquierda Detectron2 y a la derecha mmDetection

La mejora es clara, ahora se detecta la mayoría de los patos sobre el agua, e incluso se observa una disminución en la redeteccion de los objetos, viéndose de manera más clara en la figura 5.13.

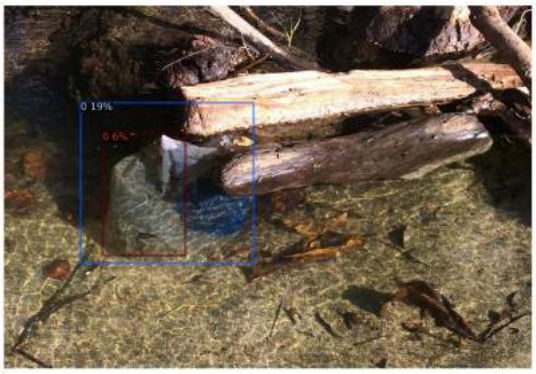
(a) (b)

Figura 5.13: Zoom zonas de interés evaluación objetos flotantes Faster X101 FPN.

Keypoint rcnn R101 FPN

Durante la exploración se encontraron situaciones de difícil elección, donde decidir qué modelo es más interesante es cada vez más complicado. Por ejemplo, tenemos el caso de la comparativa 5.14. Como se puede observar Keypoint ofrece una mayor generalización del objeto detectado con una confianza del 19 % mientras que Faster x101 32x8d FPN detecta la marca de la bolsa con una confianza del 25 %, ambos necesitan ajustar sus pesos, a simple vista determinar cuál de los dos realiza mejor la detección no es una tarea sencilla.

Keypoint



Faster R-CNN

Figura 5.14: Comparación entre (Keypoint rcnn R101 FPN) y (Faster x101 32x8d FPN) en objetos parcialmente sumergidos.

5.3. Detectron2, modelos híbridos

En búsqueda de mejorar los problemas derivados de las detecciones, dado que las detecciones siempre tienen una forma rectangular y objetos que tienen una forma alargada y quedan en diagonal, aunque queden dentro del perímetro de la detección, una gran superficie se etiquetara con la clase del objeto, a pesar de no representar la superficie que detecta la red. Se propuso explorar modelos híbridos que aúnan detección con segmentación. Donde se realiza la detección de los posibles objetos en las imágenes y posteriormente, dentro del marco de detección, se intenta segmentar el objeto detectado.

Una arquitectura que contiene todas estas características es Mask rcnn, del cual se procede a presentar los resultados de la evaluación usando los modelos de Detectron2.

Mask rcnn R50 C4

Como se aprecia en la comparativa 5.15 se amplía la información de las detecciones quedando, además del recuadro de detección, la segmentación del objeto detectado.

Mask R-CNN Faster R-CNN

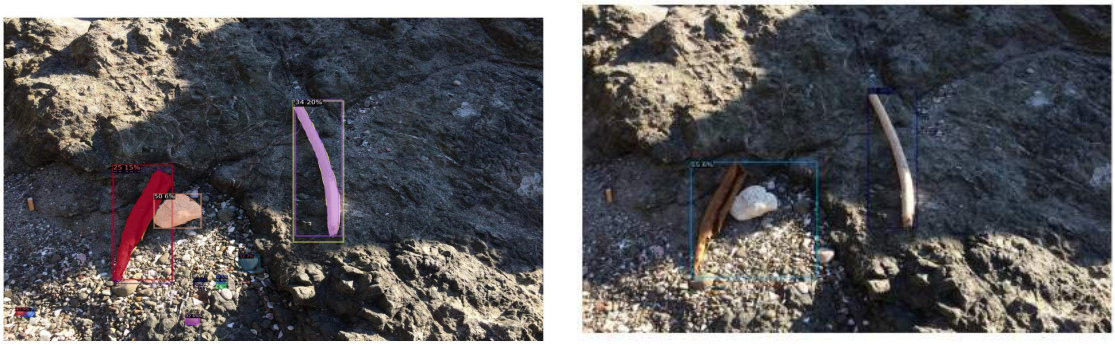
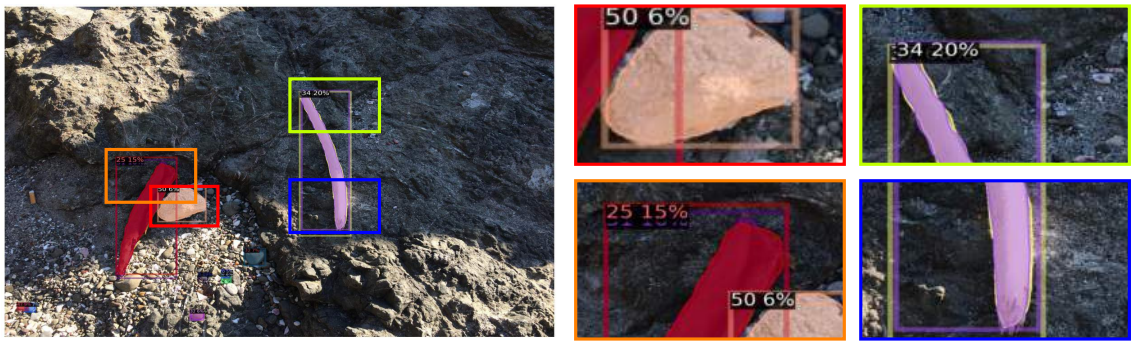


Figura 5.15: Comparación entre Mask rcnn y Faster rcnn detectando objetos alargados.

En más detalle como muestra la figura 5.16, vemos como se segmenta la forma de los objetos mejorando de forma sustancial la información de los objetos detectados.



(a) (b)

Figura 5.16: Zoom zonas de interés evaluación objetos flotantes Mask rcnn R50 C4.

En el caso de detección de objetos parcialmente enterrados se identifican la mayoría de los objetos, aunque se siguen escapando objetos se aprecia una superioridad frente a Faster rcnn, detectando una mayor cantidad de objetos más el

añadido de segmentar que parte especifica de la imagen constituye la detección, tal como se presenta en la comparativa 5.17.

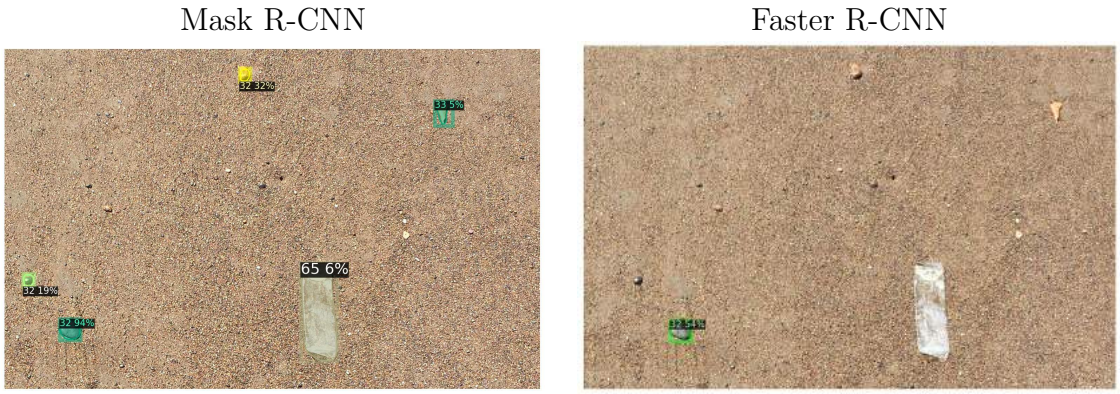
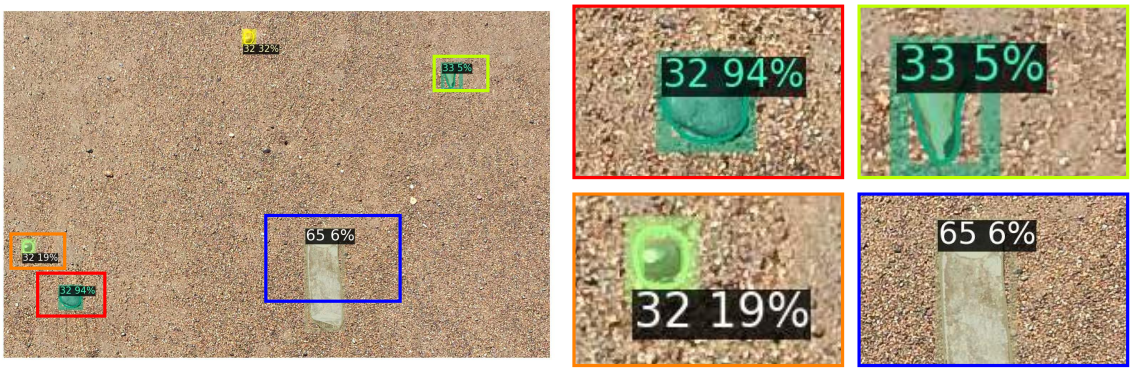


Figura 5.17: Comparación entre Mask rcnn y Faster rcnn en objetos sobre la arena.

Entrando en más detalles en la ampliación de zonas interesantes de la figura 5.18, se destaca la detección y correcta segmentación de objetos semitransparentes, además de la detección de objetos pequeños.



(a) (b)

Figura 5.18: Zoom zonas de interés evaluación objetos sobre arena Mask rcnn R50 C4.

Mask rcnn R50 DC5

En la experimentación con Mask rcnn R50 DC5 se observó una gran disminución de los objetos detectados, a costa de detecciones con mayor confianza, además se limitan los casos de redetección de objetos, tal como se ilustra en la comparación 5.19.

Mask R-CNN R50 DC5 Mask R-CNN R50 C4

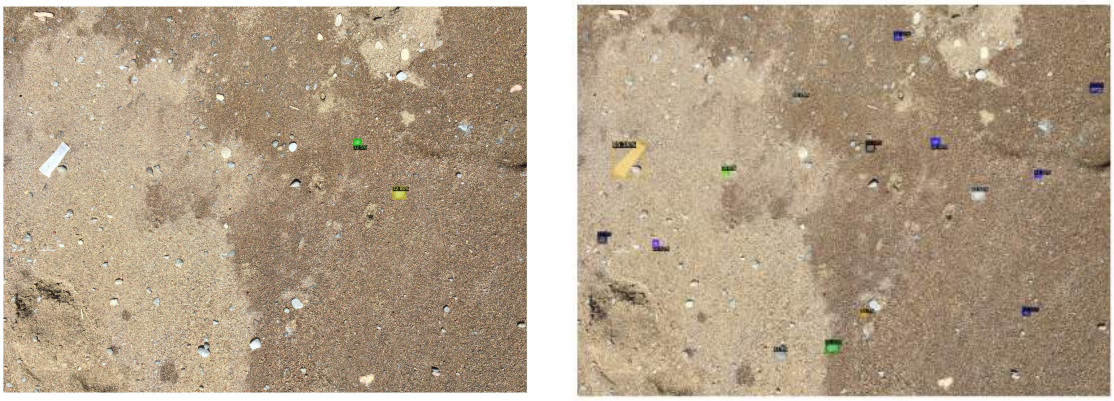
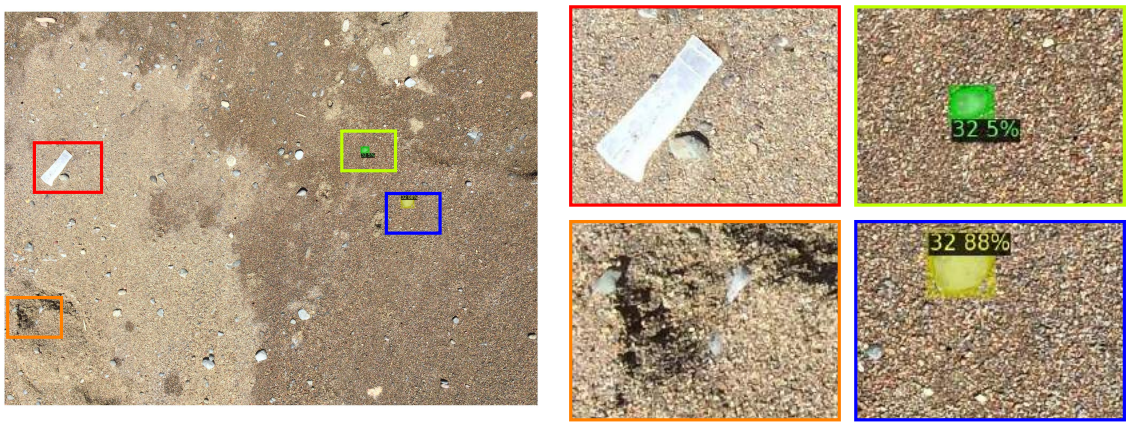


Figura 5.19: Comparación entre Mask C4 y Mask DC5 en objetos sobre la arena.

En más detalles se puede observar se pierde una gran cantidad de detecciones de objetos claros como se vislumbra en el zoom rojo y naranja, y como contraparte la gran confianza de detección en el zoom azul, tal como se muestra en la figura 5.20.



(a) (b)

Figura 5.20: Zoom zonas de interés evaluación objetos sobre arena Mask rcnn R50 DC5.

Mask rcnn X101 32x8d FPN

Ante esta pérdida de generalidad en las capacidades de detección del modelo, se procedió a explorar un modelo con mejores métricas, en busca de determinar si se trata de un caso específico, obteniendo los resultados de la comparativa 5.21.

Mask R-CNN x101 FPN Mask R-CNN R50 C4

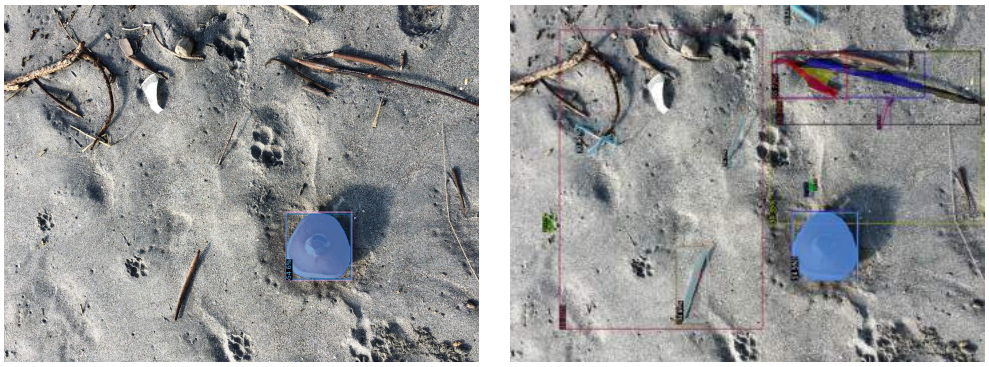


Figura 5.21: Comparación entre Mask x101 FPN y Mask R50 C4 en objetos sobre la arena.

En este caso se repite el patrón de la perdida de generalidad de las detecciones en pos de un mejor resultado en la evaluación. En más detalles se muestra la figura 5.22.

(a) (b)

Figura 5.22: Zoom zonas de interés evaluación objetos sobre arena Mask rcnn x101 FPN.

Retinanet R101 FPN

En el caso del modelo Retinanet nos encontramos con un par de situaciones interesantes, la primera consiste en los falsos positivos debidos a las características de la arena como se vislumbra en la comparativa 5.23.

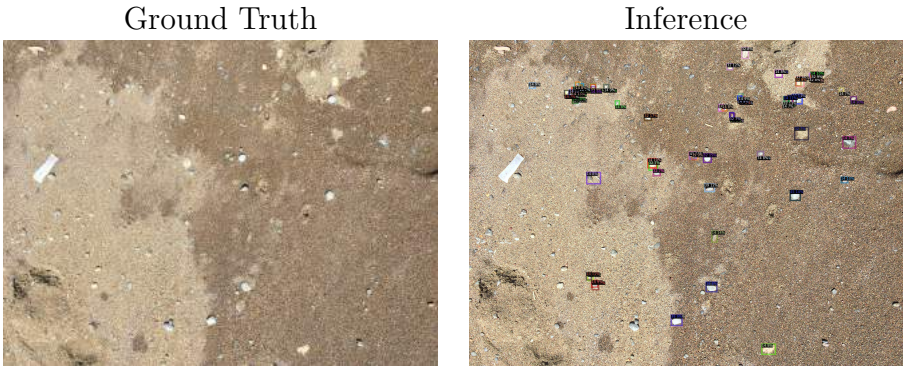


Figura 5.23: Ejemplo de evaluación de Retinanet en objetos sobre la arena.

En específico se destacan las zonas aumentadas en la figura 5.24, especialmente el aumento naranja, donde se aprecia la inestabilidad que sufren los modelos ante la variedad de tonos de la arena.

(a) (b)

Figura 5.24: Zoom zonas de interés evaluación objetos sobre arena Retinanet R101 FPN.

El segundo caso viene derivado de las propiedades de la arena, donde su superficie cambia a lo largo del tiempo, creando situaciones de luces y sombras que el modelo identifica como objetos, como se refleja en la comparativa 5.25.

Figura 5.25: Ejemplo de evaluación de Retinanet R101 FPN en objetos sobre la arena con sombras.

Concretamente se destacan las zonas aumentadas de la figura 5.26.

(a) (b)

Figura 5.26: Zoom zonas de interés evaluación objetos sobre arena con sombras Retinanet R101 FPN.

5.4. INTEL ISL DPT

Tras la puesta en contexto de las capacidades de los modelos encargados de la detección y segmentación de residuos en entornos de costa y playa. Se procedió a investigar sobre modelos capaces de segmentar una escena, puesto que los modelos de detección de objetos requieren de imágenes con la calidad de información suficiente para realizar las detecciones.

En el actual estado del arte entre las arquitecturas más prometedoras destaca DPT que está basada en los Transformers, los cuales desplazan a las redes convoluciones dado su mayor entendimiento del conjunto global de la información. Para la obtención de las imagen de los escenarios de playa se utilizaron las cámaras PTZ.

En el escenario de playa con líneas de costa bien definidas, el modelo infiere correctamente la zona de arena, además de ser capaz de diferenciar objetos sobre la arena, tal como muestra la comparativa 5.27.

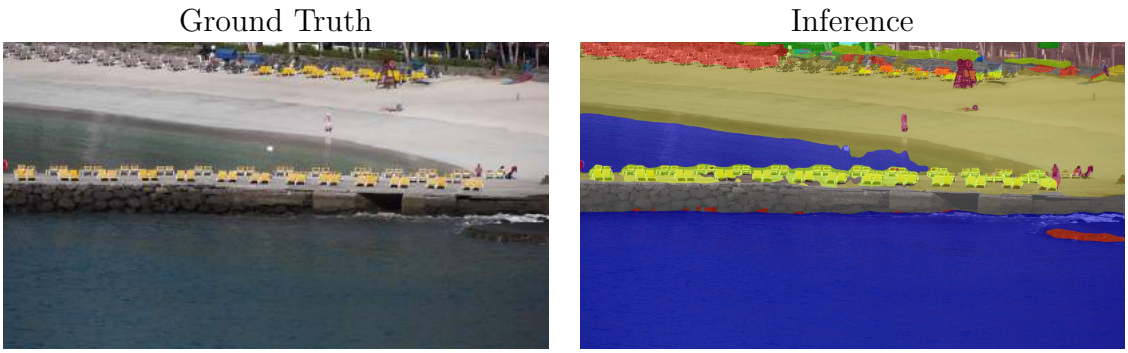
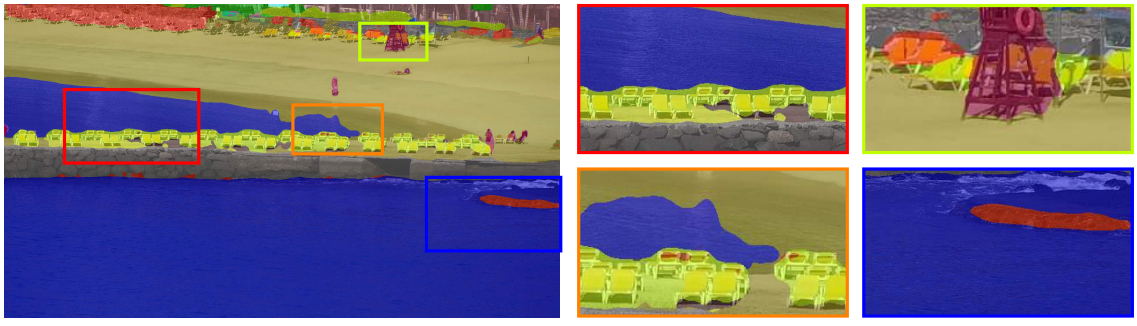


Figura 5.27: Ejemplo de evaluación de DPT en escena de playa de Anfi del Mar.

Con una vista con más detalle como muestra la figura 5.28, el modelo es capaz de separar el muelle de la costa, aunque tiene errores la segmentación del mar y presenta problemas en la definición de la línea de costa.



(a) (b)

Figura 5.28: A la izquierda la imagen original mientras que a la derecha distintos *zooms* en zonas de interés para evaluar la segmentación de escenas de la playa de Anfi del Mar.

En el escenario de una playa con una línea de costa irregular como lo es la playa de pata la vaca se puede observar que se segmenta con bastante precisión la línea de costa, dejando una clara distinción entre el mar y la arena de la playa expuesta, tal como muestra la comparativa 5.29.

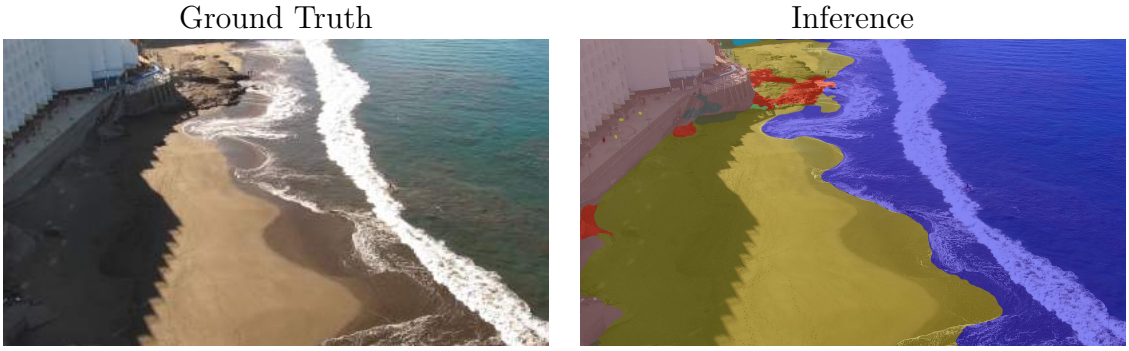
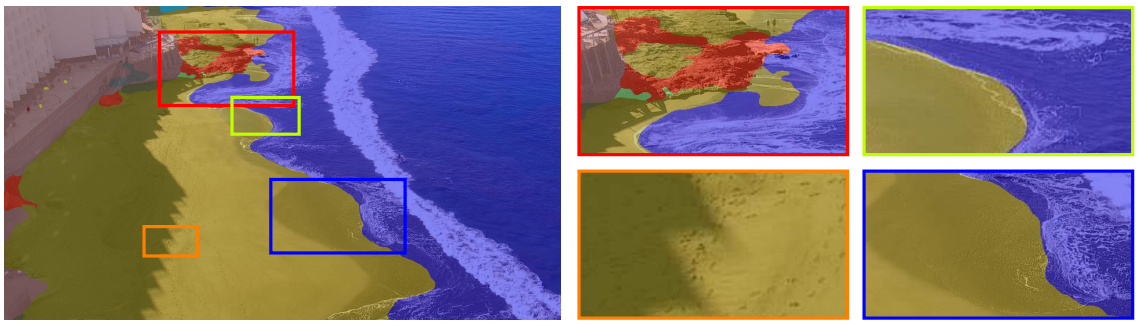


Figura 5.29: Ejemplo de evaluación de DPT en escena de playa de Patalavaca.

Incluso el modelo es capaz de generalizar la zona de arena, a pesar del presente cambio en la iluminación y el color de la arena al mojarse, tal como se aprecia en los aumentos en 5.30.



(a) (b)

Figura 5.30: A la izquierda la imagen original mientras que a la derecha distintos *zooms* en zonas de interés evaluación segmentación de escenas con obstáculos.

5.5. Métricas de Evaluación

Para la calificación de los diferentes modelos de una manera fidedigna y ajustada ha la realidad, se utilizaran procedimientos que determinaran las diferencias de certeza entre los modelos.

5.5.1. Conceptos de evaluación

- TP (*True Positive*)

  Contabiliza aquellos casos donde la clase predicha se corresponde con la real.

- FP (*False Positive)*

  Contabiliza los casos donde se predice una instancia que no corresponde a ninguna real.

- FN (*False Negative*)

  Contabiliza la cantidad de instancias reales que carecen de una predicción

- Exhaustividad (*Recall*)

  Mide la capacidad de la correcta identificación en las instancias reales.

$$Recall = \frac{TP}{TP + FN}$$

- Precisión (*Precision*)

  Calcula el porcentaje de aciertos de las predicciones realizadas.

$$Precision = \frac{TP}{TP + FP}$$

- IoU (Intersection Over Union)

  Calcula la semejanza entre dos áreas, es decir, la unión de ambos conjuntos, obteniendo 1 cuando ambas áreas tienen exactamente el mismo dominio y 0 cuando no comparten ningún valor.

$$IoU = \frac{Area\,of\,Overlap}{Area\,of\,Union}$$

5.5.2. Métricas Evaluadoras

- mAP *(mean Average Precision)* Se define como:

$$mAP = \sum_{n=1}^{N} \frac{AP(n)}{N}$$

N: Total de clases
AP(n): Precisión de clase

- F1

Corresponde con la media armónica de la precisión con la exhaustividad

$$F1 = 2 \times \frac{Accuracy \times recall}{Accuracy + recall}$$

- Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.6. Evaluación de los modelos de Detectron2

Para la evaluación de los modelos de forma objetiva se ha escogido un subconjunto de imágenes interesantes del conjunto de datos TACO, en específico la división 7, además debido a las diferencias entre las clases de COCO con las que en sido entrenados los modelos, se ha interrelacionado las clases de ambos conjuntos de datos, tal como se describe en el cuadro 5.3.

TACO	COCO
Other	apple, banana, baseball, bat, broccoli, cake, carrot, donut, orange, pizza, sandwich
String rope	backpack, baseball, glove, couch, handbag, kite, umbrella
book	Carton
Plastic	bench, bottle, bowl, cell phone, chair, clock, dining table, fork, frisbee, hair drier, keyboard, laptop, mouse, oven, skateboard, snowboard, spoon, sports ball, surfboard
Can	cup
Cup	cup, bowl
Metal	fire hydrant, fork, knife, refrigerator, remote, scissors, toaster
Glass	vase, wine glass

Cuadro 5.3: Interrelación entre las clases de TACO con COCO

Evaluación de modelos de detección

En la evaluación de los modelos de detección de detectron2 se obtuvieron tablas de los resultados de las métricas. A continuación, se procede a explicar los datos más relevantes.

En la cuadro 5.4 se presentan los resultados del modelo de Detectron2 Faster rcnn R50 FPN, donde se observa a primera vista un dato interesante en la clase *Can* la IoU es de 0.508 mientras que el accuracy es de 0.0, lo cual se debe a hecho de que el modelo detecta con relativa precisión los objetos dentro de la imagen, sin embargo la clase que le asigna a la detección no es la correcta, lo cual tiene sentido al ser un modelo que no ha sido entrenado en el ámbito especifico de la detección de residuos.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.508	0	7	10	0	0.0	0.0	0.0	0	0.0	1.0
Other	0.279	12	54	8	22	0.354	0.182	0.6	0.279	0.6	0.711
Metal	0.332	2	50	7	3	0.081	0.038	0.222	0.066	0.222	0.943
Plastic	0.259	64	49	39	83	0.626	0.566	0.621	0.593	0.621	0.371
Carton	0.374	0	2	39	0	0.0	0.0	0.0	0	0.0	1.0
Glass	0.719	6	11	13	8	0.368	0.353	0.316	0.333	0.316	0.579
String rope	0.6	2	90	1	1	0.032	0.022	0.667	0.042	0.667	0.989
Cup	0.528	2	12	9	3	0.192	0.143	0.182	0.16	0.182	0.8

Cuadro 5.4: Evaluación de métricas de Detectron2 Faster rcnn R50 FPN

Para el modelo Detectron2 Faster rcnn R101 FPN, tal como se muestra en el cuadro 5.5 se produce una mejora significativa respecto a Detectron2 Faster rcnn R50 FPN 5.4, teniendo solo el caso de la clase *Glass* en la que se pierde generalidad y se dejan de detectar objetos de la clase *Glass*.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.614	0	6	10	0	0.0	0.0	0.0	0	0.0	1.0
Other	0.309	10	38	10	26	0.429	0.208	0.5	0.294	0.5	0.594
Metal	0.36	2	38	7	2	0.082	0.05	0.222	0.082	0.222	0.95
Plastic	0.351	74	31	29	73	0.71	0.705	0.718	0.712	0.718	0.298
Carton	0.57	0	1	39	0	0.0	0.0	0.0	0	0.0	1.0
Glass	0.766	3	18	16	3	0.15	0.143	0.158	0.15	0.158	0.857
String rope	0.709	2	78	1	2	0.048	0.025	0.667	0.048	0.667	0.975
Cup	0.531	2	9	9	3	0.217	0.182	0.182	0.182	0.182	0.75

Cuadro 5.5: Evaluación de métricas de Detectron2 Faster rcnn R101 FPN

En el cuadro de los resultados de Detectron2 Faster rcnn X101 32x8d FPN 5.6, se observa un empeoramiento de los resultados, teniendo como caso interesante la clase *Metal* que recibe una mejora, no por el aumento de detecciones, sino por la disminución en las detecciones erróneas.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.47	0	5	10	0	0.0	0.0	0.0	0	0.0	1.0
Other	0.382	7	32	13	8	0.25	0.179	0.35	0.237	0.35	0.8
Metal	0.389	2	23	7	2	0.118	0.08	0.222	0.118	0.222	0.92
Plastic	0.36	46	14	57	53	0.582	0.767	0.447	0.564	0.447	0.209
Carton	0.469	0	2	39	0	0.0	0.0	0.0	0	0.0	1.0
Glass	0.863	4	7	15	5	0.29	0.364	0.211	0.267	0.211	0.583
String rope	0.685	0	62	3	0	0.0	0.0	0.0	0	0.0	1.0
Cup	0.539	1	6	10	2	0.158	0.143	0.091	0.111	0.091	0.75

Cuadro 5.6: Evaluación de métricas de Detectron2 Faster rcnn X101 32x8d FPN

En el caso de la evaluación de Detectron2 Retinanet R101 FPN 5.7 se observa un cambio drástico de los resultados, donde se aumenta significativamente el número de detección, pero empeorando el IoU de las clases, dejándolo como el modelo más sensible en ambos sentidos, para detectar el máximo número de objetos y el modelo que más falla en las detecciones.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.276	10	97	3	13	0.187	0.093	0.769	0.167	0.769	0.882
Other	0.093	30	658	14	193	0.249	0.044	0.682	0.082	0.682	0.773
Metal	0.294	5	262	0	14	0.068	0.019	1.0	0.037	1.0	0.949
Plastic	0.103	128	783	4	1154	0.62	0.141	0.97	0.245	0.97	0.404
Carton	0.405	3	18	33	2	0.089	0.143	0.083	0.105	0.083	0.9
Glass	0.361	20	93	19	32	0.317	0.177	0.513	0.263	0.513	0.744
String rope	0.262	2	552	0	4	0.011	0.004	1.0	0.007	1.0	0.993
Cup	0.407	2	101	0	3	0.047	0.019	1.0	0.038	1.0	0.971

Cuadro 5.7: Evaluación de métricas de Detectron2 Retinanet R101 FPN

Evaluación de modelos de híbridos

En la evaluación de los modelos de híbridos de detectron2 se obtuvieron los siguientes resultados.

Para el modelo Detectron2 Mask rcnn R50 FPN 5.8 se observan una gran catidad de detecciones como se ve en los valores de TP y TN de la clase *Plastic*, lo cual nos indica una gran sensibilidad en las detecciones realizadas, por otro lado la IoU refleja un bajo ajuste en las detección como se observa en la clase *Plastic*.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.451	27	239	131	22	0.117	0.102	0.171	0.127	0.171	0.916
Other	0.228	145	997	219	184	0.213	0.127	0.398	0.193	0.398	0.844
Metal	0.377	25	594	90	44	0.092	0.04	0.217	0.068	0.217	0.931
Plastic	0.228	1260	1026	514	1791	0.665	0.551	0.71	0.621	0.71	0.364
Carton	0.331	10	26	358	8	0.045	0.278	0.027	0.05	0.027	0.765
Glass	0.672	40	101	266	42	0.183	0.284	0.131	0.179	0.131	0.706
String rope	0.386	12	1166	24	18	0.025	0.01	0.333	0.02	0.333	0.985
Cup	0.615	22	140	79	15	0.145	0.136	0.218	0.167	0.218	0.903

Cuadro 5.8: Evaluación de métricas de Detectron2 Mask rcnn R50 FPN

En la evaluación de Detectron2 Mask rcnn R101 FPN 5.9 se observa una disminución la cantidad de detecciones, lo cual mejora la IoU al realizarse menores detecciones erróneas.

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.518	5	139	50	4	0.045	0.035	0.091	0.05	0.091	0.972
Other	0.242	75	517	79	100	0.227	0.127	0.487	0.201	0.487	0.838
Metal	0.402	9	285	47	9	0.051	0.031	0.161	0.051	0.161	0.969
Plastic	0.284	526	409	261	653	0.638	0.563	0.668	0.611	0.668	0.385
Carton	0.346	8	15	196	5	0.058	0.348	0.039	0.07	0.039	0.75
Glass	0.668	10	46	89	13	0.146	0.179	0.101	0.129	0.101	0.78
String rope	0.49	5	418	12	9	0.032	0.012	0.294	0.023	0.294	0.979
Cup	0.624	11	54	45	8	0.161	0.169	0.196	0.182	0.196	0.871

Cuadro 5.9: Evaluación de métricas de Detectron2 Mask rcnn R101 FPN

En el caso de Detectron2 Mask Rcnn X101 32x8d FPN 5.10 se obtiene un caso atípico en la clase *Glass* superando con creces al resto de modelos, sin embargo a cambio se ha perdido generalidad en las detecciones disminuyendo drásticamente el número de detecciones y dejando las clases *Can , Carton y String Rope* sin TP,

Class	IoU	TP	FP	FN	TN	accuracy	precision	recall	F1	TPR	FPR
Can	0.315	0	6	17	0	0.0	0.0	0.0	0	0.0	1.0
Other	0.353	8	31	12	9	0.283	0.205	0.4	0.271	0.4	0.775
Metal	0.348	2	22	7	2	0.121	0.083	0.222	0.121	0.222	0.917
Plastic	0.342	43	27	63	49	0.505	0.614	0.406	0.489	0.406	0.355
Carton	0.3	0	3	45	0	0.0	0.0	0.0	0	0.0	1.0
Glass	0.828	7	10	13	9	0.41	0.412	0.35	0.378	0.35	0.526
String rope	0.711	0	55	3	0	0.0	0.0	0.0	0	0.0	1.0
Cup	0.454	2	5	11	3	0.238	0.286	0.154	0.2	0.154	0.625

Cuadro 5.10: Evaluación de métricas de Detectron2 Mask Rcnn X101 32x8d FPN

Capítulo 6

Experimentos

A modo de trabajo complementario y dado que uno de los modelos con mejores métricas YOLOv4, no se encuentra disponible en los frameworks explorados, se ha realizado un entrenamiento del modelo con el conjunto de datos TACO, a modo de presentación de las capacidades que podría tener un modelo con conocimiento específico del problema.

6.1. Darknet

Como base del modelo se ha optado por el framework Darknet programado en C con soporte para GPU, concretamente se ha trabajado con darknet-AlexeyAB [2] una rama de darknet que ofrece el soporte para el entrenamiento, además de un punto de partida para la realización del entrenamiento, dado que también se ofrecen modelos preentrenados sobre el conjunto COCO.

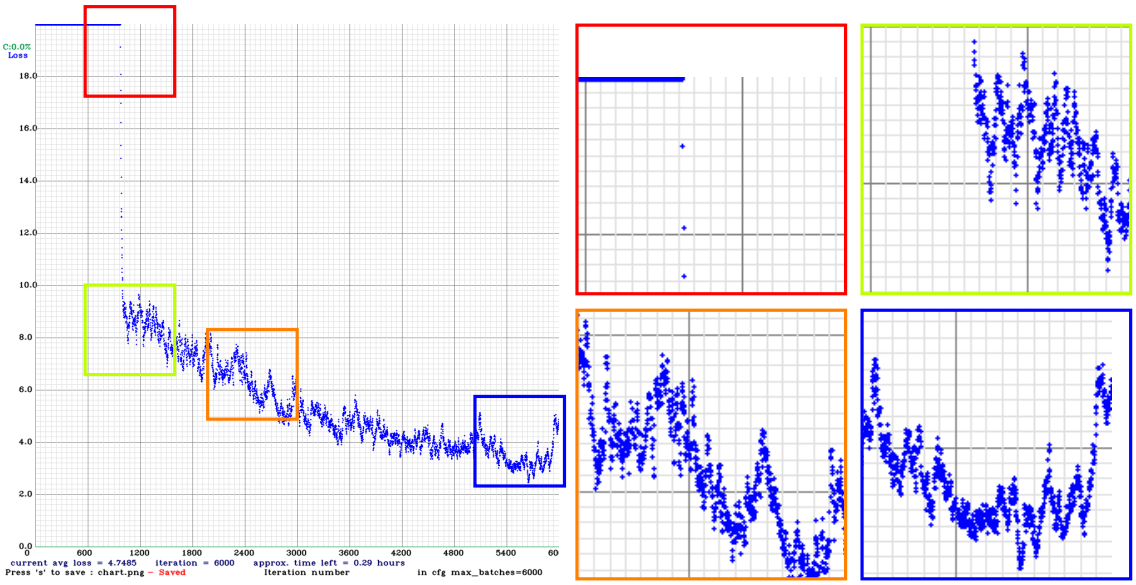
Las configuraciones necesarias para realizar el entrenamiento son:

- La elección de los pesos iniciales, para los que se eligieron yolov4.conv.137, un estado intermedio de entrenamiento en COCO
- La modificación de los archivos del modelo, para ajustar la cantidad de clases a detectar
- Cambio del formato de las etiquetas de TACO, al formato de YOLOv4
- Separación de un subconjunto de TACO para evaluar el modelo, en este caso se eligió la división 7 como conjunto de evaluación

6.1.1. Entrenamiento YOLOv4

Una vez realizadas las configuraciones iniciales, se procedió a entrenar YOLOv4 en una NVIDIA GTX 1070 con el conjunto de datos de TACO, durante 6000 iteraciones tomando 12 horas para la finalización, donde se obtuvieron los siguientes resultados:

Como se puede observar en la figura 6.1 que representa la función de perdida de YOLOv4, durante el inicio la función se encuentra fuera de rango y no comienza a descender hasta pasada la iteración 1300, la evolución es irregular y presenta grandes saltos en la optimización de la ecuación de coste, incluso al final del entrenamiento. Este comportamiento es habitual en conjuntos difíciles donde la información no es suficiente para generalizar los objetos a detectar, lo cual se corresponde con las características de TACO puesto que es un conjunto de tan solo 1500 instancias.



(a) (b)

Figura 6.1: Métricas de entrenamiento YOLOv4 en TACO.

6.1.2. Entrenamiento YOLOv4 con Data Augmentation

A la vista de los resultados y de la irregularidad en la optimización de la función de coste, se recurrió a la utilización de técnicas de data augmentation en búsqueda de aumentar la variabilidad de los datos aportados a la red.

A continuación, se presenta un ejemplo de las técnicas utilizadas:

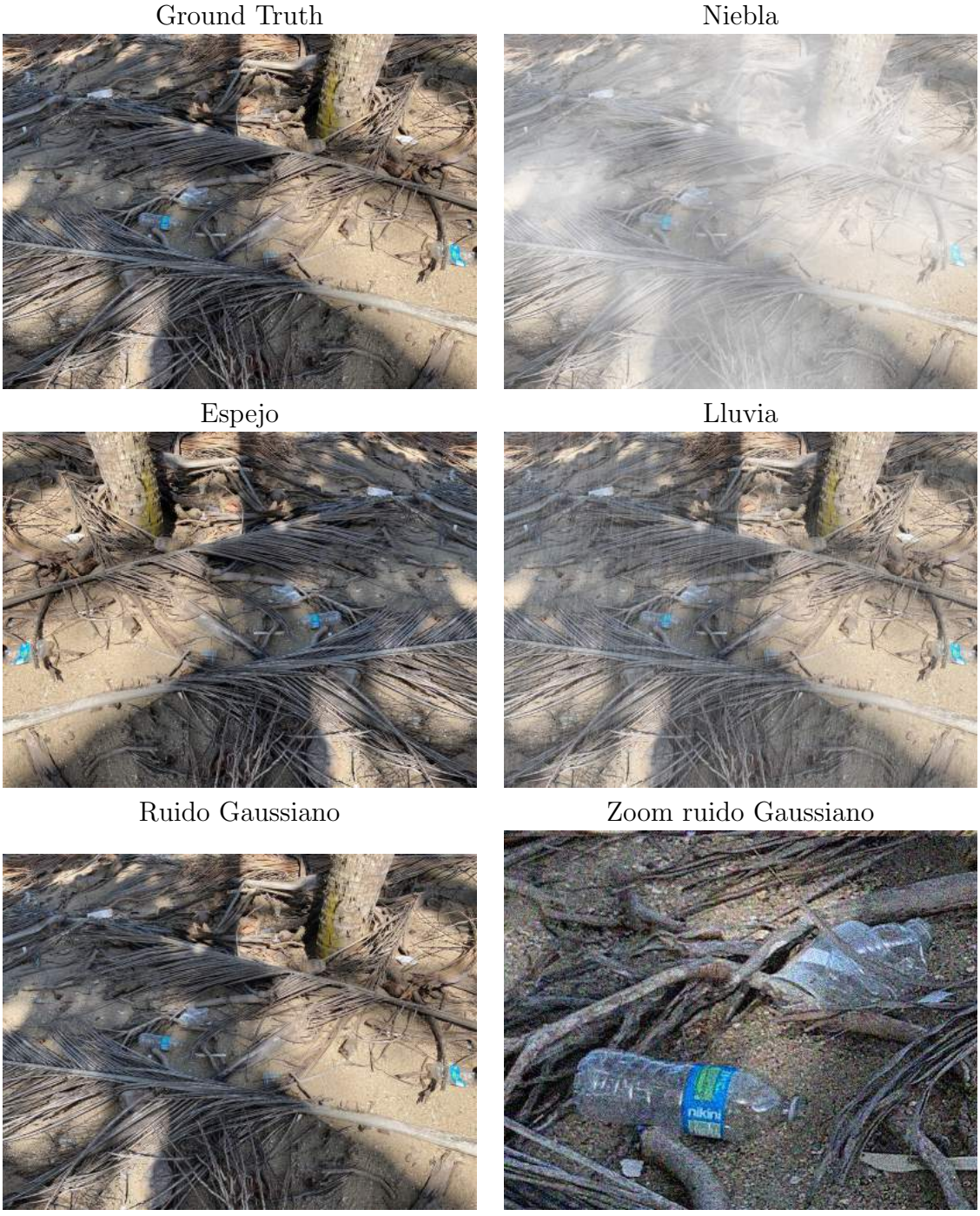
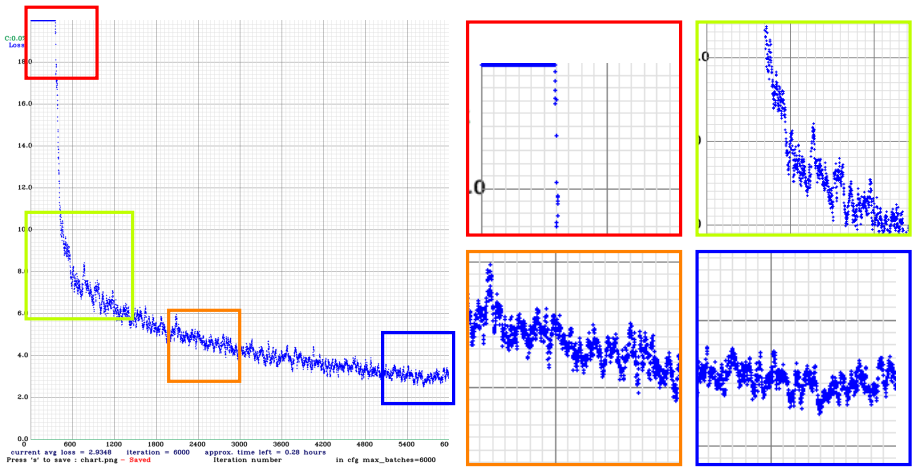


Figura 6.2: Ejemplo técnicas utilizadas en data augmentation.

Una vez aumentado el conjunto de datos se volvió a entrenar YOLOv4 desde el punto inicial obteniendo la función de coste representada en la figura 6.3. Donde

destaca la rapidez con la que la red optimiza la función de coste, sobre la iteración 450, además de la continua estabilización en la optimización sin la aparición de grandes picos.



(a) (b)

Figura 6.3: Métricas de entrenamiento YOLOv4 en TACO con data augmentation.

Una vez finalizados los entrenamientos se decidió comparar las capacidades de los distintos pesos con el subconjunto reservado de TACO para la evaluación. A continuación, se muestra una comparativa entre los dos entrenamientos.

YOLOv4 YOLOv4 Data Augmentation

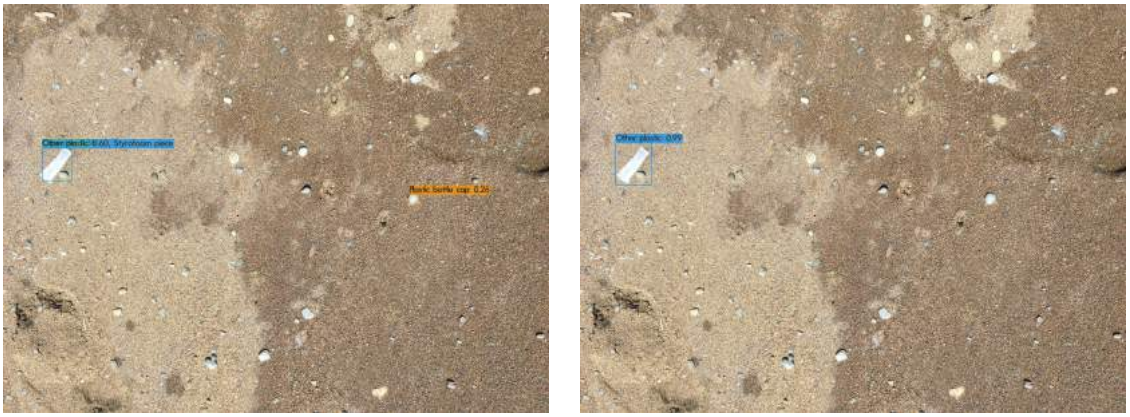


Figura 6.4: Comparativa entrenamientos YOLOv4 en imagen con huellas en la arena. En la imagen con data augmentation de aprecia la eliminación del falso positivo

YOLOv4 YOLOv4 Data Augmentation

Figura 6.5: Comparativa entrenamientos YOLOv4 en imagen con cambios en la iluminación. En la imagen con data augmentation se le asigna la clase correcta, además de no redetectar el plástico

YOLOv4 YOLOv4 Data Augmentation

Figura 6.6: Comparativa entrenamientos YOLOv4 en imagen con objetos enterrados. En la imagen con data augmentation se detecta el objeto plástico semienterrado en la arena

YOLOv4 YOLOv4 Data Augmentation

Figura 6.7: Comparativa entrenamientos YOLOv4 en imagen con objetos ocluidos. En la imagen con data augmentation de aprecia la eliminación de la redeteccion de la botella de plástico ocluida por la arena.

YOLOv4 YOLOv4 Data Augmentation

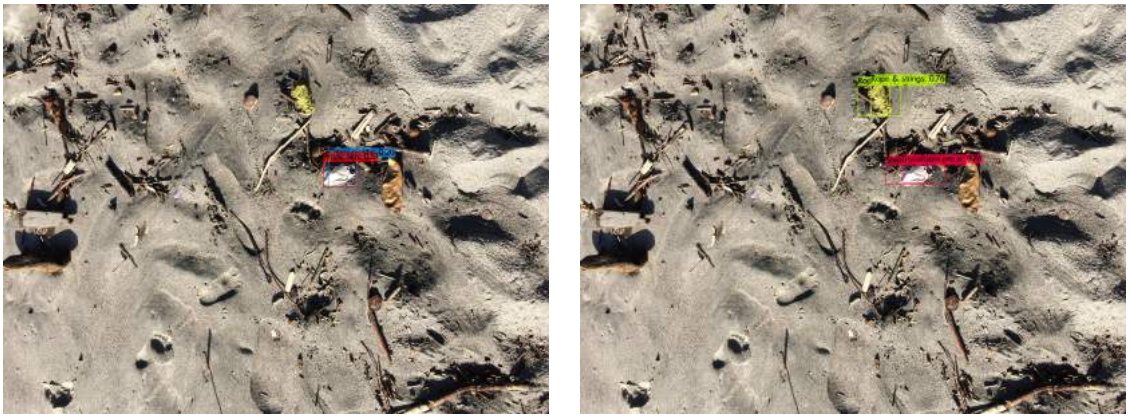


Figura 6.8: Comparativa entrenamientos YOLOv4 en imagen con sombras. En la imagen con data augmentacion se detecta el objeto textil dentro del hoyo de la arena

Capítulo 7

Conclusiones

Tras el trabajo realizado se han sacado las siguientes conclusiones. El grado de desarrollo actual de los métodos de Deep learning permite generar una amplia variedad de soluciones, como se observa en la cantidad de modelos donde no es de extrañar el intercambio de partes especializadas en busca de mejorar las capacidades de las redes. Además, se logrado una generalización lo suficientemente robusta para aplicar los modelos a ámbitos donde en principio no tienen conocimiento previo de sus características, donde se debe balancear entre el grado de especialización sobre el conjunto de datos entrenado y los de validación. A modo de aceleradores de trabajo se dispone de frameworks que aúnan diferentes versiones de los modelos con sus respectivos preentrenamientos, permitiendo la prueba inicial de los modelos en el ámbito concreto en los que se aplica y valorar las capacidades de las soluciones que ofrecen.

Por otro lado, en el ámbito concreto de costa y playa se identifican los siguientes problemas a resolver. La detección de objetos que contextualmente tengan el significado de residuo, así como sus diferentes vertientes en la detección de materiales, de objetos y microplásticos.

Dentro de los escenarios descritos se destacan las siguientes cuestiones:

- Detección de materiales La detección de los materiales conlleva asumir la creación de clases con mayor generalidad, por ejemplo, en la clase plástico se asociarían distintos tipos de plásticos en una gran variedad de formas y aquellos elementos que tengan el suficiente contenido de plástico como para incluirlos en la clase.

- Detección de objetos

  En la detección de objetos se encuentra la dificultad de elección en determinar que objeto constituye una instancia de residuo, puesto que la información que determina esta propiedad es contextual y constituye el siguiente paso en la

mejora de los detectores de objetos. Este escenario actualmente es el más explorado contando con los modelos a la vanguardia de la investigación.

- Detección de Microplásticos

  En el caso particular de los microplásticos nos encontramos una problemática totalmente distinta donde se procesan imágenes con una gran cantidad de posibles instancias en los que los ligeros matices de las partículas determinan su clase, dificultando especialmente la detección, ya que, además de partículas de plástico en suspensión, se encuentran restos de seres vivos y polvo en las imágenes que a simple vista son iguales, lo cual requiere de las muestras a las cuales se ha fotografiado para determinar su clase en el conjunto de entrenamiento.

Además de la detección de instancias es de especial interés la segmentación de las escenas para la identificación de las zonas, sobre las cuales se utilizarán los métodos en busca de los residuos, ya que resulta de interés focalizar que zonas son las realmente objetivos a monitorizar, a lo cual se le puede añadir la detección de los objetos con distintos aumentos, mejorando la cantidad de objetos detectados.

Respecto a la experimentación con los modelos en el reentrenamiento con un conjunto de datos específico para un problema concreto, se destaca la importancia de la variabilidad en los datos y la calidad de estos. Donde el uso de las técnicas de data augmentation pueden marcar la diferencia para la optimización de la función de coste, enriqueciendo las situaciones con las que se entrena a la red.

Capítulo 8

Trabajo futuro

En el trabajo realizado quedan áreas por explorar, en cuanto al entrenamiento realizado, se necesitaría de un conjunto con una mayor presencia de escenas del entorno marítimo, con el cual se podría comparar sobre las capacidades de TACO, permitiendo vislumbrar las posibles capacidades del modelo sin la restricción de la falta de datos. En cuanto al área de los microplásticos un conjunto de datos enfocado en su detección representa un punto clave para determinar cuáles son las capacidades de los modelos. En el escenario de la segmentación de imagen costeras se plantea el estudio del estado del arte comparando las capacidades de los distintos modelos, así como, la inclusión de un conjunto de datos especifico que permita su evaluación.

Además, queda pendiente la integración de los modelos dentro de la pila de trabajo, donde se seguiría el siguiente procedimiento:

- Movimiento del sistema de obtención de imágenes
- Segmentación de la escena capturada
- Delimitación de regiones con interés en la monitorización
- Detección de los objetos presentes en las escenas

Una vez terminada la obtención de las detecciones adquiere mayor relevancia el uso práctico de los datos obtenidos. Respecto a dicha cuestión se proponen las siguientes posibles utilidades:

- Sistema de notificación del estado de residuos de la escena monitorizada

  En esta situación se reportaría los datos a la entidad de la cual se le ofrecería el servicio de monitorización de residuos.

- Evaluación del estado de contaminación de microplásticos

  Una vez determinada la clasificación y contabilización de los objetos, se realizaría un análisis exploratorio del grado de contaminación de la arena y agua.

- Análisis del consumo de productos por zonas

  De manera indirecta se podría inferir los productos que se consumen en una zona monitorizada, lo cual permitirá realizar análisis y estudios de mercado para la implantación de productos dentro de una localización. Además, existiría la posibilidad de usar dicha información de forma complementaria a distintos análisis, por ejemplo, estimación de la salud de la población de manera sectorizada, hábitos de consumo por regiones, e incluso estimar que productos se consumen fuera del ámbito del hogar.

Capítulo 9

Glosario

9.1. Definiciones

- Framework

  Estructura inicial en el desarrollo de software, para la resolución de problemas de manera estandarizada.

- Inferencia

  Proceso mediante el cual se extraen conclusiones a partir de datos de entrada.

- Convoluciones

  Función transformativa que indica el grado de superposición de dos funciones.

- Transformer

  Modelo de deep learning especializado en el tratamiento de datos secuenciales, con la capacidad de procesarlos fuera de orden mediante mecanismos de atención.

- Red Neuronal (RN)

  Estructura básica, en la cual se transforma el input para la extracción de información no diferénciale a simple vista.

- Deep Learning (DL)

  Área de estudio de la inteligencia artificial, en el cual se intenta automatizar las tareas humanas mediante redes neuronales.

- Hardware

  Conjunto de elementos físicos que permiten la creación de un sistema Turing completo

- Unidad de Procesamiento Grafico (GPU)

  Elemento hardware especializado en el tratamiento de operaciones repetitivas masivas, normalmente trabaja con grandes conjuntos de datos.

- Unidad de Procesamiento Central (CPU)

  Elemento hardware centrado en la gestión de los recursos de los sistemas informáticos.

- Data Augmentation

  Técnica mediante la cual se transforman conjuntos de datos aumentando su variabilidad informativa, sin introducir contradicciones, o solapamientos.

- Dataset

  Conjunto de datos, normalmente con una temática que permite interconectarlos.

- Anotaciones

  Información adicional a los conjuntos de datos, donde es habitual que se la respuesta a obtener por el sistema inteligente.

- Interfaz de Programación de Aplicaciones(API)

  Puntos de conexión de las aplicaciones, que permiten su uso desde un programa distinto.

Apéndice A

Detalles técnicos sobre la implementación del proyecto

Durante el desarrollo de este TFG se ha hecho uso de diferentes scripts para la resolución de las tareas plantadas. A continuación, serán explicadas las partes más importantes.

Las dependencias utilizadas durante la ejecución del framework son las siguientes:

```
import json
import multiprocessing
import cv2
import torch
import compress_json
from detectron2.utils.visualizer import Visualizer
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
import os
```

A continuación, se implementan de las métricas de evaluación de los modelos respecto a sus predicciones.

```
def accuracy(TP, TN, FP, FN):
    if (TP + TN + FP + FN) == 0:
        return 0

    return (TP + TN) / (TP + TN + FP + FN)


def precision(TP, FP):
    if (TP + FP) == 0:
        return 0

    return TP / (TP + FP)


def recall(TP, FN):
    if (TP + FN) == 0:
        return 0

    return TP / (TP + FN)


def F1(TP, FP, FN):
    if (precision(TP, FP) + recall(TP, FN)) == 0:
        return 0

    return 2 * ((precision(TP, FP) * recall(TP, FN)) /
                (precision(TP, FP) + recall(TP, FN)))


def TPR(TP, FN):
    if (TP + FN) == 0:
        return 0

    return TP / (TP + FN)


def FPR(FP, TN):
    if (FP + TN) == 0:
        return 0

    return FP / (FP + TN)
```

```
def bbox_IoU(boxA, boxB):
    boxB = [boxB[0], boxB[1], boxB[0] + boxB[2], boxB[1] + boxB[3]]
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])

    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)

    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou
```

Para realizar el mapeo de clases entre el dataset de COCO y TACO se leen las clases de cada uno y se realiza el mapeo como se especifica es read_data(model), donde model es el nombre del modelo a evaluar

```
def read_data(model):

    map_COCO = []
    with open("./Clases_datasets/classes_COCO_map.txt", 'r') as reader:
        for index, line in enumerate(reader.readlines()):
            map_COCO.append([index, line.split(',')[1].split('\n')[0]])

    data = []
    with open("./RESULTS/Metricas/detectron2/raw" +
                                model + 'data.csv', 'r') as r:

        for line in r.readlines():
            classes = line.split('[')[1].split(']')[0].split(', ')

            for map in map_COCO:
                for index, cl in enumerate(classes):
                    if cl == str(map[0]):
                        classes[index] = map[1].strip()

            image = line.split(',')[0]
            IoU = line.split(']')[1].split(',')[1:-1]
            data.append([image, classes, IoU])
    return data
```

De igual manera que en el mapeo con COCO para TACO hay que realizar la misma operación para llegar a un conjunto de clases iguales como se implementa en get_GT_classes, además se retornan las anotaciones de las imágenes requeridas para la evaluación.

```
def get_GT_classes():

    with open("./annotations.json") as f:
        TACO = json.load(f)

    map_TACO = []
    images = TACO['images']
    annotations = TACO['annotations']

    GT_classes = []
    for annotation in annotations:
        GT_classes.append([annotation['image_id'],
                           annotation['category_id']])

    for image in images:
        for GT_classe in GT_classes:
            if GT_classe[0] == image['id']:
                GT_classe[0] = image['file_name']


    with open("./Clases_datasets/classes_TACO_map.txt", 'r') as reader:
        for index, line in enumerate(reader.readlines()):
            map_TACO.append([index, line.split(',')[1].split('\n')[0]])

    for GT_classe in GT_classes:
        for map in map_TACO:
            if map[0] == GT_classe[1]:
                GT_classe[1] = map[1].strip()

    data = {}

    for GT_classe in GT_classes:
        if not GT_classe[0] in data:
            data[GT_classe[0]] = [GT_classe[1]]
        else:
            data[GT_classe[0]] = data[GT_classe[0]] + [GT_classe[1]]

    return data
```

En la función make_metrics(model, GT_classes, head) se define el proceso de evaluación, donde model es el modelo a evaluar, GT_classes es el resultado de mapear las clases de TACO y COCO y head es la cabecera que tendrá el archivo CSV que contendrá el resultado de la evaluación.

```
def make_metrics(model, GT_classes, head):
    data = read_data(model)
    metrics_image = make_metrics_by_instance(GT_classes, data)
    metrics_class = make_metrics_by_class(GT_classes, data)
    to_csv(model, "/by-class", metrics_class, head)
```

Para realizar pruebas de inferencia sobre los modelos de manera rapida se utiliza la funcion model_inference_test_detectron2(model_name, url_weigth, config_dir, img_dir), donde model_name es el nombre del modelo, url_weigth es la url para descargar los pesos del modelo, config_dir es la configuracion del modelo y img_dir es la ruta con las imagenes.

```
def model_inference_test_detectron2(model_name, url_weigth,
                                    config_dir, img_dir):


    cfg = get_cfg()
    cfg.merge_from_file(config_dir + model_name + ".yaml")
    cfg.MODEL.WEIGHTS = "detectron2://" + url_weigth
    pred = DefaultPredictor(cfg)


    with torch.no_grad():
        inputs = cv2.imread(img_dir)
        outputs = pred(inputs)

        v = Visualizer(inputs[:, :, ::-1])
        out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
        cv2.imwrite("./RESULTS.jpg", out.get_image()[:, :, ::-1])
```

Para evitar que los modelos evaluados se queden con recursos de la maquina, en especifico la memoria de la GPU, se lanzan la evaluación de los modelos como hilos evitando el bloqueo a su acceso mediante la funcion start_thead(target) donde target es la función a lanzar como un hilo.

```
def start_thead(target):

    p = multiprocessing.Process(target=target)
    p.start()
    p.join()
```

La función model_inference_test_detectron2(model_name, url_weigth, config_dir) se utiliza para realizar las inferencias de los distintos modelos y se extraen las predicciones, además se guardan los datos comprimidos para su posterior análisis. Los parámetros se definen como model_name nombre del modelos a evaluar, url_weigth, dirección de la cual se descargan los pesos del modelo y config_dir como la configuración del modelo.

https://es.overleaf.com/project/60186aaf51b2208c0c9223e7

```
def model_inference_test_detectron2(model_name, url_weigth, config_dir):

    with open('./annotations.json') as f:
        data = json.load(f)

    cfg = get_cfg()
    cfg.merge_from_file(config_dir + model_name + ".yaml")
    cfg.MODEL.WEIGHTS = "detectron2://" + url_weigth
    pred = DefaultPredictor(cfg)

    os.makedirs("./RESULTS/Datos/detectron2" + model_name,
                                                exist_ok=True)

    os.makedirs("./Images_inference/detectron2" + model_name,
                                                exist_ok=True)


    os.makedirs("./RESULTS/Datos/detectron2" + model_name +
                                        "/batch_7", exist_ok=True)

    os.makedirs("./RESULTS/Imagenes/detectron2" + model_name +
                                        "/batch_7", exist_ok=True)


    with torch.no_grad():
        for image in data['images']:
```

```
if image['file_name'].split('/')[0] == "batch_7":
    inputs = cv2.imread('./Dataset_test/'
                        + image['file_name'])

    outputs = pred(inputs)

    os.makedirs("./RESULTS/Datos/detectron2"
    + model_name + "/" + image['file_name'].split("/")[0],
    exist_ok=True)

    instance = outputs["instances"]

    path = "./RESULTS/Datos/detectron2" + model_name
    + "/" + image['file_name'].split(".")[0] + ".json"

    compress_json.dump({'pred_boxes':
                        instance.pred_boxes.tensor.tolist(),
        'scores': instance.scores.tolist(),
        'pred_classes': instance.pred_classes.tolist(),
        #'pred_masks': instance.pred_masks.tolist()
        },
        path + ".gz")
    print("Imagen numero: ", image['file_name'], "\n")
```

Bibliografía

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] AlexeyAB. https://github.com/alexeyab/darknet, 2020.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[5] Sofía Cruz. Pinterest.com.mx/yverelintineo, 2017.

[6] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[7] hackernoon. https://hackernoon.com, 2020.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] istockphoto. istockphoto.com, 2018.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[12] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[13] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[15] Harsh Panwar, Pradeep Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, Prakhar Bhardwaj, Sudhansh Sharma, and Iqbal Sarker. Aquavision: Automating the detection of waste in water bodies using deep transfer learning. *Case Studies in Chemical and Environmental Engineering*, 2:100026, 07 2020.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[17] Pedro F. Proença and Pedro Simões. TACO: trash annotations in context for litter detection. *CoRR*, abs/2003.06975, 2020.

[18] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021.

[19] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[20] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[21] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[23] Yuheng Wang, Wen Jie Zhao, Jiahui Xu, and Raymond Hong. Recyclable waste identification using CNN image recognition and gaussian clustering. *CoRR*, abs/2011.01353, 2020.

[24] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.