UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicaciones de Visión Artificial en Dispositivos de Edge Computing

Presentado por Miriam Torres Calvo
en Universidad de Burgos — 30 de junio de 2022
Tutor: Bruno Baruque Zanón

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanón, profesor del departamento de Ingeniería Informática., área de Ciencia de la Computación e Inteligencia Artificial.

Expone:

Que la alumna Dª. Miriam Torres Calvo, con DNI 45575901K, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

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

En Burgos, 30 de junio de 2022

Vº. Bº. del Tutor:

D. Bruno Baruque Zanón

Resumen

La tecnologia avanza continuamente y a velocidades que hace unos años eran inexplicables, contamos con su ayuda en diferentes entornos de la sociedad y cada vez en mayor medida. Pero también, contamos con situaciones en las cuáles está no se encuentra tan fácil de acceder, ya que no se cuenta con la facilidad para transportarla y así poder usarla en muchos mas lugares y mas comodamente.

Así surge este proyecto, con la idea de poder usar un modelo de Machine Learning en dispositivos de Edge Computing, como puede ser la Jetson Nano de NVIDIA.

Para su desarrollo, se contara con el lenguaje Python y el modelo escogido para entrenar ha sido YOLO en su cuarta versión.

Descriptores

Deep Learning, Edge Computing, Jetson Nano, YOLO, Python Object Detection, Trasnsfer Learning

Abstract

Technology is advancing continuously and at speeds that were inexplicable a few years ago, we count on your help in different environments of society and to a greater extent. But also, we have situations in which it is not so easy to access, since it is not easy to transport it and thus be able to use it in many more places. and more comfortably.

This is how this project arose, with the idea of being able to use a Machine Learning model in Edge Computing devices, such as the NVIDIA Jetson Nano.

For its development, the Python language will be used and the model chosen for training has been YOLO in its fourth version.

Keywords

Deep Learning, Edge Computing, Jetson Nano, YOLO, Python Object Detection, Trasnsfer Learning

Índice general

Índice de figuras

Índice de tablas

Introducción

Hoy en día la tecnologia ha avanzado tanto, que es muy fácil contar con ella a la hora de realizar ciertas tareas, pero cada vez la demandamos más para poder trabajar codo con codo con ella, es decir, recibir su ayuda de tal forma que les podamos asignar tareas asegurando que tendrán un porcentaje de acierto igual o superior al que tendría si lo realizasemos cualquiera de nosotros.

Pero generalmente, para poder llevar a cabo estas tareas, se necesitan dispositivos con una gran cantidad de computo, ya que necesitaremos entrenarlo con el objeto u objetos a predecir, siendo está la tarea más importante y la que más capacidad de computo va a necesitar y la que más recursos va consumir. Tras su entrenamiento, volveremos a consumir recursos para su detección, de tal forma que necesitaremos un equipo lo suficientemente potente para poder realizar ambas tareas con efectividad y poder obtener buenos resultados.

Debido a esto, el poder entrenar el modelo en un ordenador lo suficientemente potente y seguidamente poder adaptarlo para poder ser utilizado en dispositivos pequeños como puede ser la Jetson Nano de NVIDIA, y que este dispositivo lo ejecute, sacrificando el porcentaje de acierto pero respetando los tiempos de ejecución, puede facilitar a muchos trabajadores y/o investigadores en sus trabajos ya que pueden tener una herramienta funcional en poco espacio y además fácil de transportar para poder usarla en diferentes lugares.

1.1. Estructura de la memoria

La memoria consta de las siguiente estructura:

- **Introducción:** establece el contexto inicial entorno a la idea que se va a desarrollar, además de la estructura del documento y de los materiales que se van a entregar.
- **Objetivos del proyecto:** objtivos que se desean alcanzar durante el desarrollo del proyecto.
- **Conceptos teóricos:** exponer los conceptos que son necesarios disponer para llevar a cabo el proyecto.
- **Técnicas y herramientas:** muestras las técnicas y las herramientas que se han utilizado durante el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** recopilación de los aspectos más representativos que han tenido lugar durante el desarrollo del proyecto.
- **Trabajos relacionados:** presentación de trabajos que se encuentran relacionados de manera destacable con el desarrollo o el concepto del proyecto.
- **Conclusiones y líneas de trabajo futuras:** descripción de las conclusiones obtenidas durante la realización del proyecto y tras la misma, así como las posibles líneas de mejora.

Además, junto a la presente memoria se incluyen los siguientes anexos relacionados con el desarrollo del modelo de detección y su correspondiente prueba en el dispositivo de Edge Computing:

- **Plan de Proyecto Software:** presentar la planificación temporal llevada a cabo durante el desarrollo del proyecto, así como un estudio de la viabilidad del desarrollo.
- **Especificación de Requisitos:** describir de forma detallada lso objetivos generales y los objetivos del proyecto llevado a cabo.
- **Especificación de diseño:** presentar el diseño final del modelo, describiendo el diseño de datos, procedimental y arquitectónico del desarrollo.
- **Documentación técnica de programación:** en este apartado se describen los conocimientos técnicos más relevantes del proyecto, los cuáles son necesarios para poder continuar con el desarrollo.

- **Documentación de usuario:** apartado dirigido al usuario final, dónde se describen los requisitos necesarios en un dispositivo para poder utilizar la herramienta, la instalación de cada uno de ellos, y un manual de usuario, en el que se mostrarán todas las posibles opciones que dispone la herramienta.

Objetivos del proyecto

En este apartado se van a presentar los objetivos que han marcado el proyecto, tanto a nivel software, como técnico.

2.1. Objetivos Software

- Creación del script que permita entrenar el modelo de detección con las clases seleccionadas, obteniendo como resltado el modelo entrenado (al realizarse mediante YOLO, devolverá un fichero .weights)
- Convertir el modelo YOLO (.weights) a un modelo Tensorflow (.pb) para poder trabajar con él.
- Creación de los scripts que permitan la detección de los objetos, ya sea partiendo de una imagen o de vídeo (cargando un vídeo o conectando la webcam), los cuáles contarán con diferentes flags de accion durante la ejecución.
- Creación de un script de preprocesado de cara a la evaluación de varias imagenes etiquetadas, de tal forma que podamos obtenerla información original en un único fichero.
- Medir la calidad de la evaluación del modelo, es decir, calcular el IoU entre las posiciones originales y las predecidas por el modelo, pudiendo obtener su mAP, y resultados sobre la predicción(verdaderos positivos, falsos postivos...), además de que por cada imagen se devolverá la imagen con la posición original, la posición predicha y el IoU.
- Mostrar las predicciones en un csv, que muestre el tiempo en el que se ha detectado la predicción, el número de objetos predecidos en dicho

instante y el tipo de objeto que es, así como la posición o posiciones en las que se ha encontrado.

- Crear un script, que permita identificr las clases con un tracker, es decir, que identifique las clases y nos las vaya etiquetando según vaya detectando.

2.2. Objetivos Técnicos

- Convertir el modelo a uno apto para la característicasde la Jetson Nano.
- Usar la plataforma *GitHub* para la organización y gestión del proyecto.
- Seguir los principios de la *metodología ágil Scrum*.
- Usar herramientas que permitan medir la calidad del código.

Conceptos teóricos

Para la compresion de este proyecto, se deben conocer los siguientes conceptos:

3.1. Deep Learning

El Deep Learning [1] es una rama del Machine Learning, donde los algoritmos inspirados en el funcionamiento del cerebro humano (redes neurnales) aprenden a partir de grandes cantidades de datos y tratan con un alto número de unidades computacionales.

Gracias a la neurociencia, el estudiio de casos clínicos de daño cerebral sobrevenidoy los avances en diagnóstico por imágenes sabemos que hay centros específicos del lenguaje, que existen redes especializadas en detectar diferentes aspectos de la visión, como los bordes, la simetría, áreas relacionadas con el reconocimiento de rostros y las expresiones emociales de los mismos. Los módelos de Deep Learning imitan estas características de arquitectura del sistema nervioso, permitiendo que dentro del sistema global haya redes de unidades de proceso que se especialicen en la detección de determinadas características que se encuentran ocultas en los datos. Dicho enfoque, ha permitido obtener mejores resultados si los comparamos con la redes monolíticas de neuronas artificiales

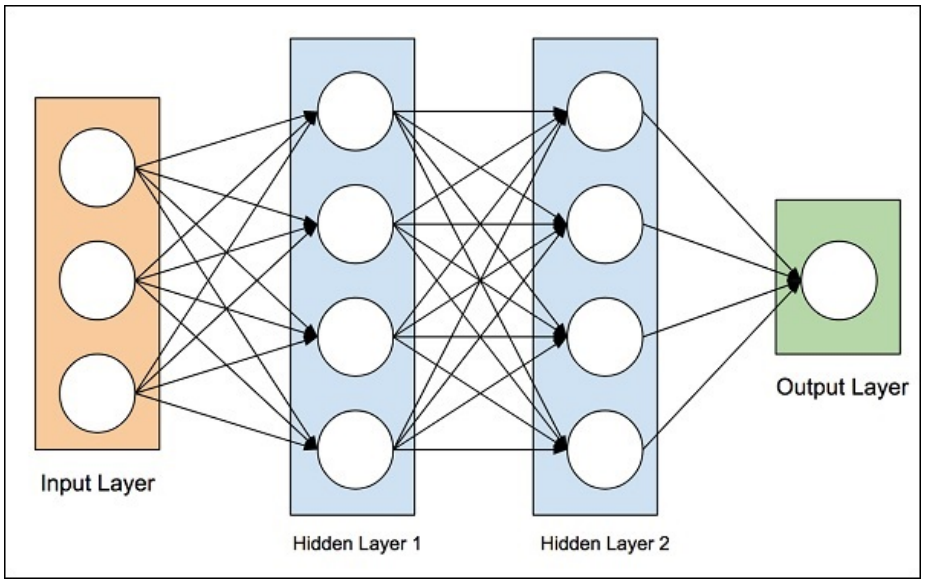

Figura 3.1: Red neuronal convolucional

3.2. Edge Computing

El Edge Computing [2] es un tipo de informática que ocurre, en la ubicación fisica del usuario, en la ubicación de la fuente de los datos o cerca de estas. Permitiendo que los usuarios obtengan servicios mas rápidos y fiables.

La ventaja fundamental de esto, es que permite a las empresas analizar los datos que sean importantes casi en tiempo real, un hecho que en áreas como la fabricación, la sanidad, las telecomunicaciones o la industria financiera, es una necesidad latente y continua.

Las necesidades industriales hacen que esta tecnología cada vez sea más demandada, debido a que en ciertos entornos la única forma de poder automatizar más los procesos, consiste en tratar de evitar lo máximo posible la comunicación con la nube, consiguiendo reducir las latencias, consumir menos ancho de banda y por su puesto acceder de manera inmediata a análisis y evaluación del estado los sensores y dispositivos que la constituyen.

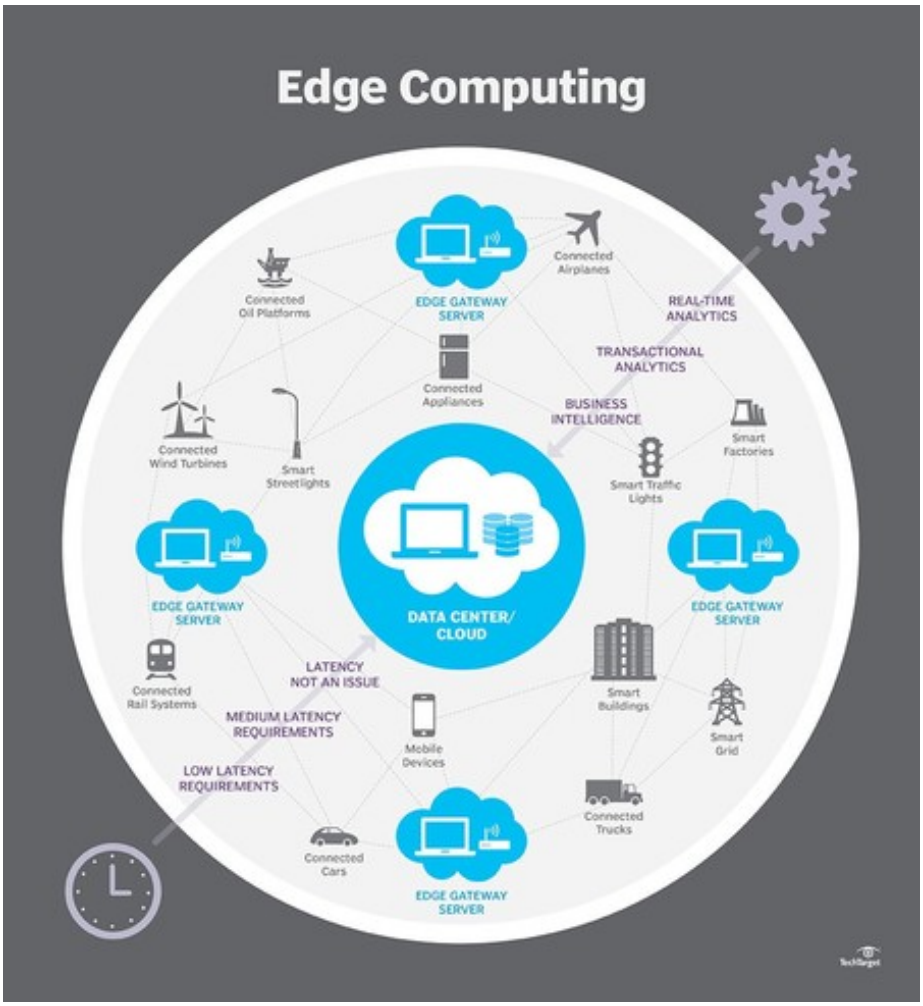

Figura 3.2: Estructura Edge Computing

3.3. Jetson Nano

Una Jetson Nano [3] es un mini PC de bajo coste, el cual cabe en una mano. Se encuentra compuesto por un SoC, procesador ARM de 64 bits de 4 núcleos y una GPU con arquitectura Maxvell con 128 núcleos de procesameinto gráfico, conectividad de red, contando con una potencia total de 472 Gflops, cuenta a su vez, con puertos USB-A, salidas de vídeo HDMI y DisplayPort y un puerto para su conexión a Internet.

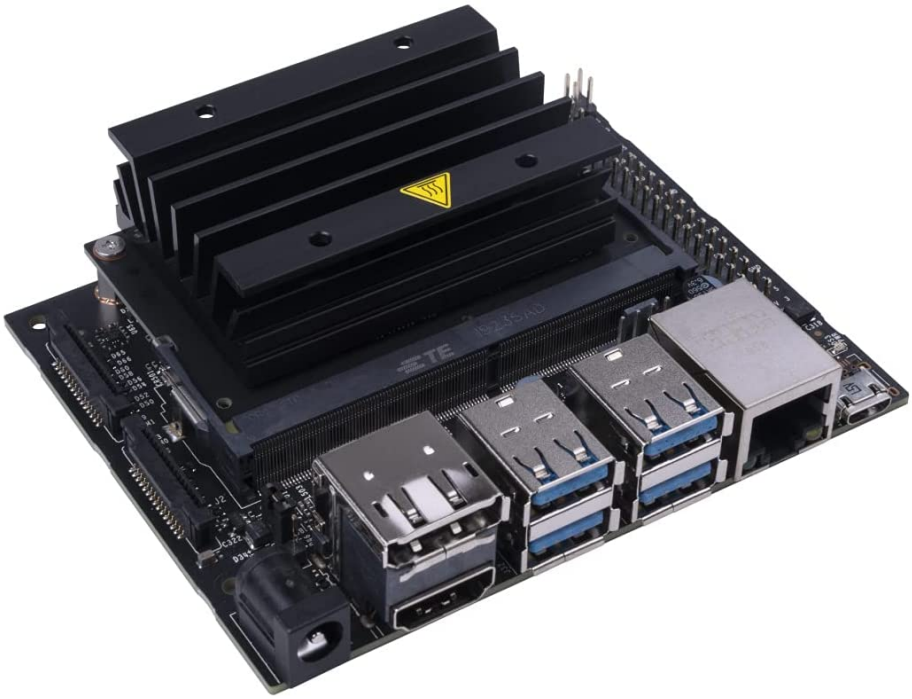

Figura 3.3: Jetson Nano

3.4. YOLO

You Only Look Once (YOLO) [4] es un algoritmo de detección que usa Deep Learning y CNN para ello, como su nombre indica sólo necesita mirar la imagen una única vez, de tal forma que la detección es mucho más rápida que en otros algoritmos, pero a cambio de sacrificar rendiemiento a la hora de predecir. Para llevar a cabo la detección, divide la imagen en una cuadríucla de SxS (imagen de la izquierda). Por cada cuadríucla, predice N posibles "bounding boxesz calcula la probabilidad de cada una de ellas, es decir, en total se predicen SxSxN cajas diferentes (la gran mayoria con una probabilidad muy baja) (imagen del centro). Por último, se eliminan las cajas que estan por debajo de un límite, conocido este como non-max-suppression, de tal forma, que se eliminan los objetos detectados por duplciado, dejando los que poseen un mayor valor de predicción (imagen de la derecha).

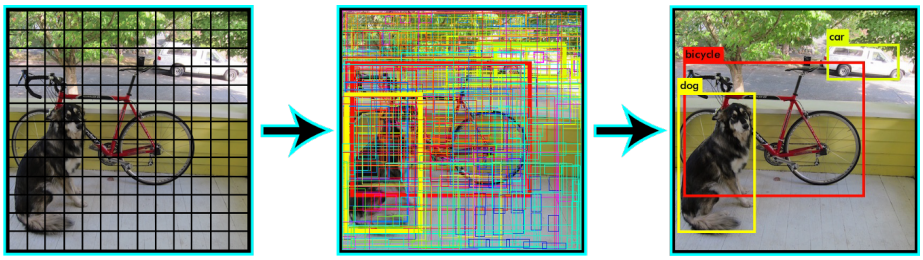

Figura 3.4: Explicación YOLO

Para entrenar un modelo, basado en el algoritmo YOLO, tendremos que tener las imégenes con las que vamos a entrenar etiquedas con un contenido como el siguiente:

```
0 0.16937475 0.7628560000000001 0.14848899999999998 0.06350499999999999
```

Figura 3.5: Representación del contenido etiquetado YOLO

El primer parámetro representa el id de la clase, es decir, cuando se etiquetan las imágenes se creara un fichero llamado classes.txt, con los nombres de todas las clases que se han etiquetado para el modelo. El segundo representa la distancia desde la coordenada 'x' al centro, mientras que el tercero hace lo propio desde la coordenada 'y'. El cuarto parámetro representa el ancho de la anotación, es decir, el ancho del recuadro que conforma la anotación, el último párametro representa el alto de la anotación.

YOLO utiliza una red CNN llamada Darknet, aunque puede ser utilizada cualquier otra red convolucional a la hora de entrenar. Además, YOLO utiliza las redes convolucionales al final de la cadena, sin necesidad de tener que convertir a una red *tradicional*. La principal cr´tica que tiene es que a pesar de ser rápida, obtiene peores resultados que las redes *R-CNN*, pero con el paso del tiempo las nuevas versiones que se han lanzado se centran en mejorar dicha precisión en los *bounding boxes*, respetando su eficacia.

La arquitectura de la red se basa en una red convolucional **GoogLeNet** [5], la cuál consta de 24 capas convolucionales. Enbebe en su salida tanto la parte que clasifica la imágenes como la de posicionamiento y tamaño de los objetos.

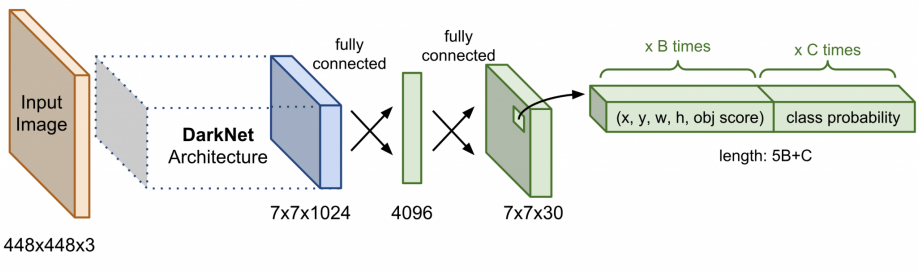

Figura 3.6: Arquitectura de la red convoluvional Darknet

Por ejemplo, para el **CocoDataset**, la cuál debe detectar 80 objetos diferentes, nos dará una salida:

Tamaño Grilla	Cantidad Anclas	Cantidad Clases	Ccore, X, Y, Alto, Ancho
13*13	*3*	(80 +	* 5)

Tabla 3.1: Salida COCO Dataset

3.5. Object Detection

El Object Detection [6] es un técnica de vision por ordenador que permite localizar imágenes y/o vídeos. Estos algoritmos se aprovechan del aprendizaje automático o del profundo con el objetivo de obtener resultados significativos, es decir, intentan replicar la inteligencia humana a la hora de reconcoer un objeto.

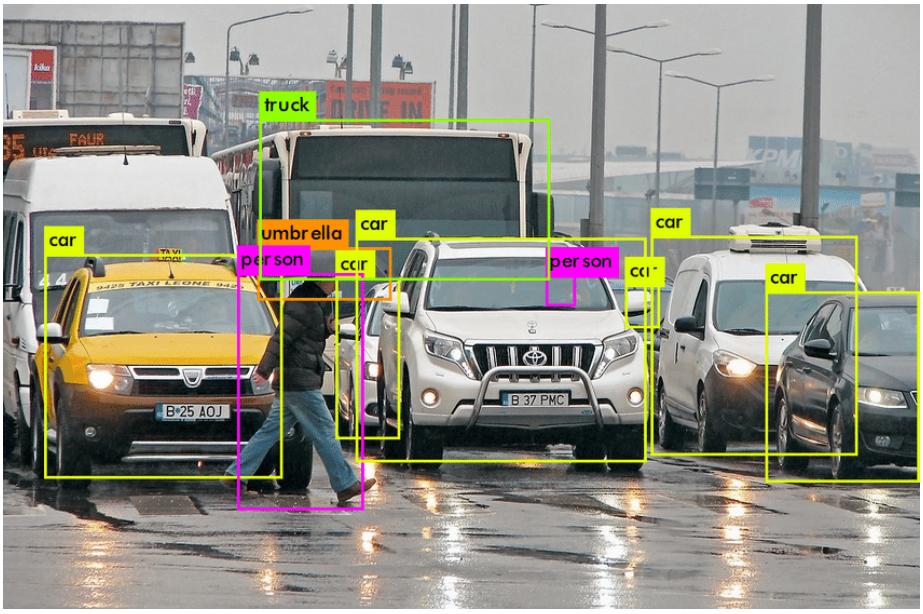

Figura 3.7: Detección de objetos

Técnicas y herramientas

4.1. Metodología

A lo largo del proyecto se intentado seguir la *metodología ágil Scrum*, pero adaptada ya que para pdoer aplciar esta metodología es necesario contar con un equipo, en el cuál los diferentes miembros se reparten las roles entre los diferentes miembros que lo conforman. En este caso, lso roles recaeen todo sobre una única persona.

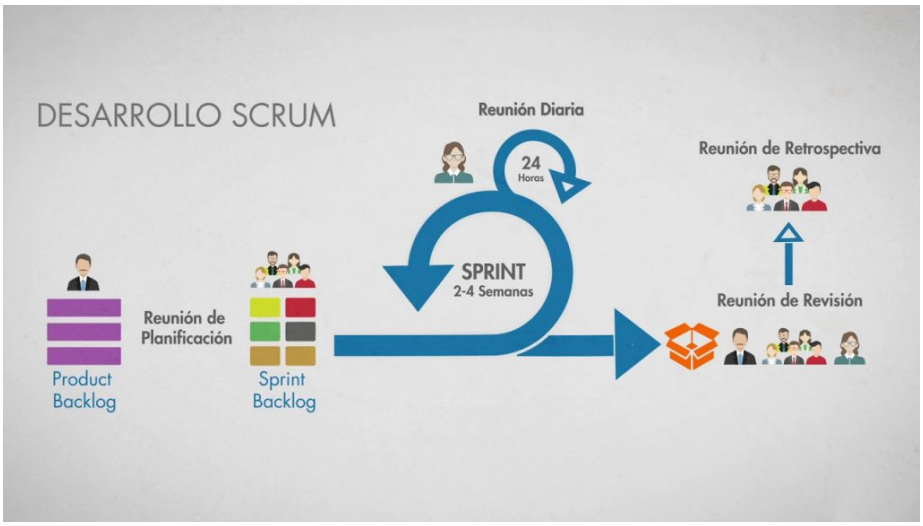

Figura 4.1: Pasos de la metodología Scrum

En primer lugar se encuentra el *Product Backlog* [7] que se trata del alcance del proyecto, el cuál va variando dependiendo de los *feedbacks* que se van obteniendo en cada *sprint*.

Seguidamente, se encuentra el *Sprint Backlog*, dónde se marcan los requerimientos que deben de alcanzar durante el *sprint* que se va a iniciar, es decir, se trata de acortar las tareas de cada uno de los *sprints*.

La siguiente etapa es el *Sprint*, en la cuál tiene lugar la planificación, la implementación, revisión y retrospectiva de la nueva característica software. Esta etapa suele tener una duración de una a dos semanas.

Como último paso del proceso, se enecuentra el *incremento del producto*, esta fase consiste en tener una reunión con el cliente con la nueva característica en funcionamiento con el objetivo de obtener una *retroalimentación* por parte del cliente y así volver a empezar el proceso de nuevo.

4.2. Lenguaje de programación

A la hora de empezar un nuevo proyecto es importante relacionado con el *Machine Learning* y el *Edge Computing*, es muy importante seleccionar el lenguaje, con el cuál queremos trabajar destacando dos: **Python** [8] y **Matlab** [9]. En este caso se decantó por el uso de *Python*, debido al mayor conocimiento de este lenaguaje y haber trabajado más con este lenguaje que con *Matlab*. No hay grandes ventajas entre escoger uno u otro.

Python

Python es un lenguaje de programación multiplataforma y multiparadigma [1], destacando entre sus características la legibilidad y la limpieza del código. Python es un *software open source*, siendo por ello gratuito sus uso.

Fue desarrollado al inicio de la década de los 90 por *Guido van Rosseun*. Fue implementado como el sucesor del lenguaje *ABC* y se encuentra fuertemente influenciado por otros como: ABC, Ada, ALGOL 68, APL, C, C++, CLU, Dylan, Haskell, Icon, Java, Lips, Modula-3, Perl, Standard ML.

Su principal objetivo es automatizar los procesos, con el fin de minimizar tanto el tiempo de desarrollo, como complicaciones, debido a esto, hoy en día Python es uno de los lenguajes más usados para el desarrolo de todo tipo de aplicaciones.

[1] Debido a que soporta parcialmente la orientación a objetos, la programación funcional y la imperativa.

4.3. Algoritmo de detección

A parte de tener claro el lenguaje que se quiere utilizar, otra característica a tener en cuenta es elegir el *algoritmo de detección* en el que se va a basar el modelo. Existen diferentes algoritmos:

- **CNN:** *(Convolutional Neuronal Network)* es la opción maás básica que se puede escoger,ya que se parte de una red neuronal convolucional [10] la cuál itera la imagen hasta devolver las posiciones de los objetos que detecta.

  Está opción trae consigo diferentes inconvenientes:

  - Si la imagen detecta varios objetos, situados en zonas opuestas, ¿cuántos píxeles tendremos que desplazarnos en cada dirección?.
  - El tiempo de cómputo es variable, pudiendo llegar a ser muy largo, ya que por cada movimiento implica una clasificación individual con la red.
  - Deetectar un objeto dentro de la red, no indica que se poseen los valores 'x' e 'y' de su posición.
  - Si por un casual el desplazamiento de píxeles que se realiza es muy pequeño, podríamos estar detectadndo el mismo objeto múltiples veces.
  - Si dos objetos se encuentran muy juntos, se podrían llegara a detectar como un único objeto.

- **R-CNN:** *(Region Based Convolutional Neural Networks)* surgen en el año 2014, con la siguiente propuesta: determianr primero las regiones de interés de la imagen y después realizar la clasificación de imagenes sobre dichas áreas usando una red preentrenada.[11] Esto implica, que haya un primer algoritmo que detecte las áreas de interés de la imagen, las cuáles pueden ser muchas y de diversos tamaños. Seguidamente, se pasán las diferentes regiones por la CNN, validandose las clases correctas mediante un clasificador bianrio, de tal forma, que se eliminarán las que tenga un bajo nivel de confianza. Por último, se ajustaría la posición mediante un regresor.

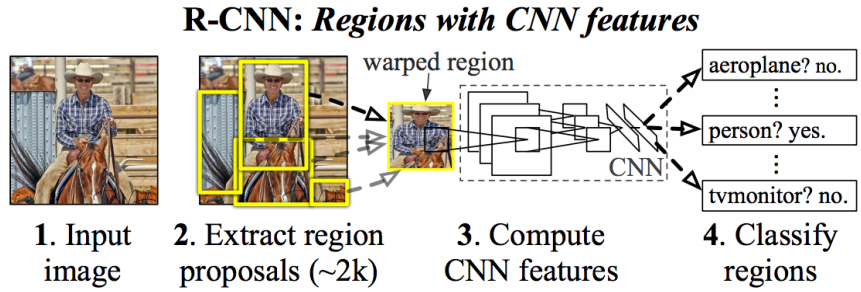

Figura 4.2: Pasos de la detección en R-CNN

- **Fast R-CNN / Faster R-CNN:** Son dos algoritmos que surgen como mejora a R-CNN [12]:
  - **Fast R-CNN:** mejora el algoritmo inicial reutilizando algunso recursos, como las *features* extraídaa por la CNN, de tal forma que se agiliza el entreno y la detección de las imágenes. Esta red, posee también mejoras en el cálculo del IoU *(Intersection Over Union)* y en la función de *Loss*. Pero a pesar de esto, no tiene mejoras drásticas en la velocidad de entrenamiento y en la detección.

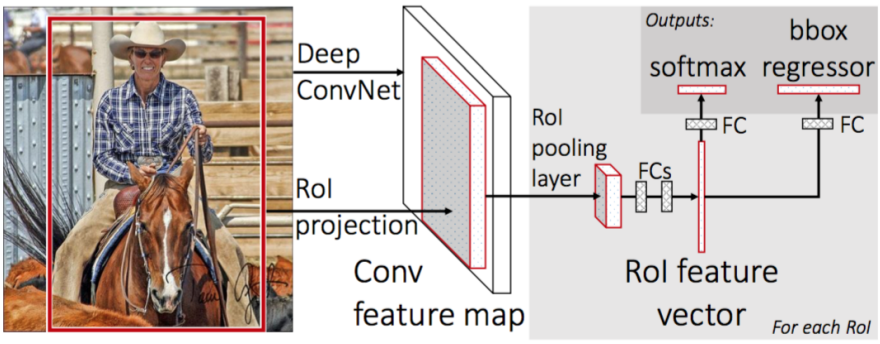

Figura 4.3: Arquitectura red Fast-RCNN

- **Faster R-CNN:** logra una mejora de velocidad al integrar el algoritmo de *region proposal* [13] sobre la propia CNN. Además aperece el concepto de usar *anchors* fijos, lo cuál consiste en usar tamaños pre calculados para la detección de obejtos de la red.

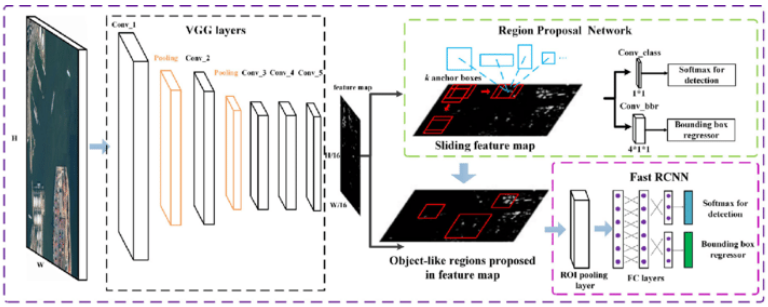

Figura 4.4: Arquitectura red Faster-RCNN

- **YOLO:** surge en 2016, su nombre viene formado por las siglas de *You Only Look Once* [4]. Esta red, como su propio nombre indica hace una única pasada a la red y detecta todos los obejtos para los que ha sido entrenada para clasificar, al realizar un único vistazo obtiene velocidades muy buenas en equipos que no son necesariamente potentes. Lo cuál permite, detecciones en tiempo real de cientos obejtos de forma simúltanea y su ejecución en dispositivos móviles.

Debido a esto, el modelo escogido ha sido YOLO y en su versión 4, la cuál fue lanzada en Abril del año 2020.

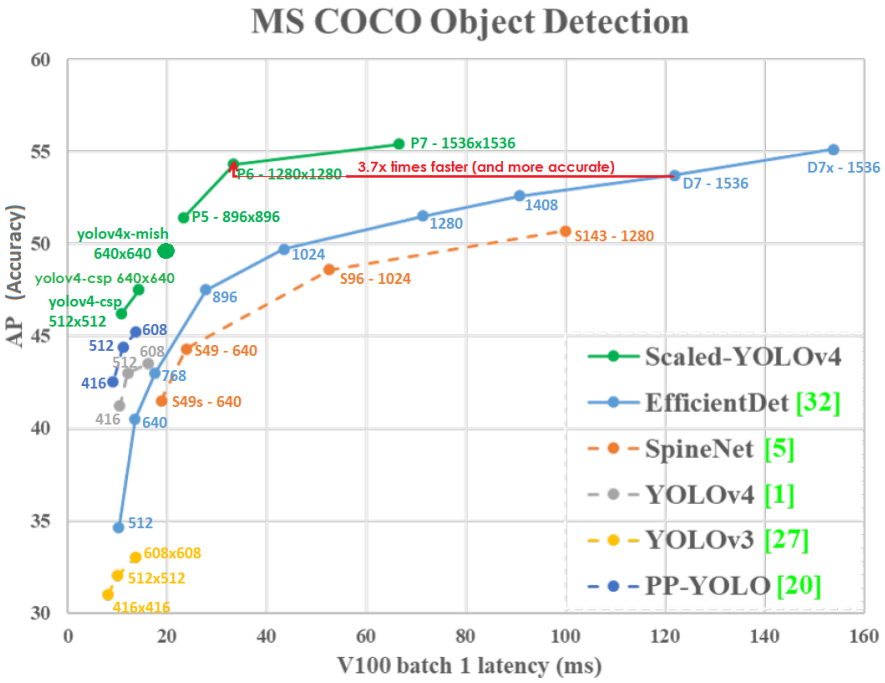

Figura 4.5: Comparativa resultados Dataset COCO

4.4. Tensorflow

Tensorflow [14] es una biblioteca de código abierto, la cuál fue lanzada en el año 2015 por Google, la cuál es muy utilizada por muchas empresas resultado de gran utilidad por su versatilidad y nivel de desarrollo.

Esta herramienta se basa en el **Deep Learning,** a pesar de que el mercado habia herramientas similares como *DistBelief* [15], la cuál también fue construida por Google en el año 2011, como un sistema propietario de aprendizaje automático. Su uso creció rápidamente debido al uso de compaañias como *Alphabet*. Pero los años pasaron y el avance de la tecnología hizo que las necesidades aumentasen, haciendo que Google invirtiese tiempo para mejorarla, dando lugar a la actual **Tensorflow,** una herramienta con un conjunto de datos mayor y con mayor capacidad de almacenamiento y modificación.

Se basa en un sistema de redes neuronales, lo cuál permite relacionar varios datos en la red de manera simultánea, es decir, imita lo que hace el cerebro humano.

4.5. TensorRT

TensorRT [16] es un *framework* de aprendizaje automático, el cuál fue publicado por **Nvidia** para ejecutar inferencias que son interferencias de aprendizaje automático en su hardware. Este *framework*, se encuentra altamente cualificado para ejecutarse en *GPUs Nvidia*, siendo una de las formas más rápidas de ejecutar un modelo en este momento.

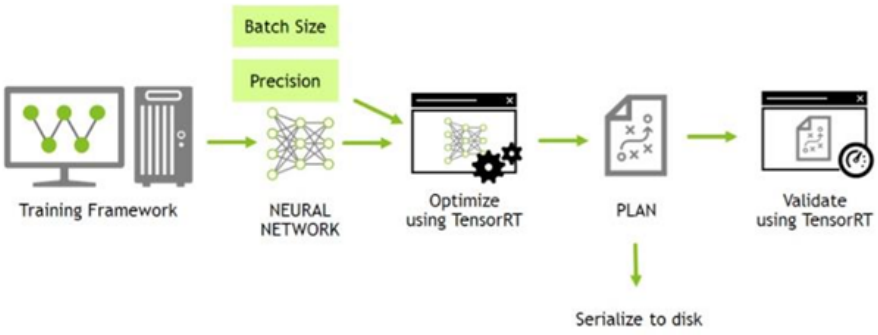

Figura 4.6: Workflow del framework *Tensorrt*

4.6. GitHub

GitHub [17] es una herramienta para el alojamiento del código desarrollado, por cualquier individuo en la web. Esta herramienta permite llevar un control de versiones, siendo esto muy interesante de cara al proyecto que se va a llevar a cabo. Otra característica destacable es que se puede integrar este sistema en *Visual Studio Code* [18] y que podemos enlazar de forma fácil a nuestro repositorio los códigos realizados con *Google Colab*[19].

4.7. Herramientas de control de calidad del código

Las herramientas de control de calidad de código evaluan y procesan de forma automática las líneas de código que conforman el proyecto, comparándolas con unos estándares de calidad. El proyecto será evaluado y dependiendo de si se cumplen o no los estándares y en que medida, la calificación será mejor o peor.

La integración de herramientas de evaluación de código, ayudan a los desarrolladores a evaluar cómo de bueno es un producto software, consiguiendo una mejor mantenibilidad, reducir los errores, hacer revisiones del código centrándose en puntos especificos del código, así como determianr como de buena es la corbertura de pruebas con el objetivo de aumentar el valor del producto software.

Para llevar acabo esta evaluación, se ha contando con la herramienta *SonarCloud*.

SonarCloud

SonarCloud [20] es una herramienta de evaluación de código que es capaz, de calcular de forma automática, la duplicación de código, la corbertura del código, su fiabilidad, mantenibilidad, así como su seguridad y sus posibles puntos vulnerables. Para llevar a cabo sus evaluaciones, SoanrCloud se fija en los estándares y reglas de cada lenguaje.

- **Duplciación de código:** identifica las líneas, fragementos o ficheros de código duplicado.
    1. Las duplicaciones en un único archivo, es igual al número de copias encontradase en dicho archivo.
    2. Un archivo se considera duplicado de otro, si el número de líneas iguales supera un determinado umbral.
- **Seguridad:** para analizar la seguridad de código, se fija en dos puntos: *Security Hostpots* y en las *vulneravilidades*
    1. Los Security Hostpots, son partes de código sensible a la seguridad, estos pueden estar bien, pero requieren de una revisión humana.

2. Las vulnerabilidades de seguridad requieren una acción inmediata, para ello, se fija en las reglas y en los estándares del lenguaje concreto

- **Quality Gate:** son un conjunto de condiciones *booleanas*, los cuáles siguen los estándares de vulnerabilidad y revisión de puntos de acceso. Esta métrica, nos indica si el código está listo pra pasar a producción.

4.8. Fork

Fork citefork, es un software diseñado para poder realizar de forma gráfica, todas las tareas que se realziarían mediante Git en la consola de comandos. Es una herramienta multiplataforma, que cuenta con soporte para Windows y MacOS; permite de forma sencilla mantenerse al tanto de repositorios, *branches*, etiquetas, históricos, realizar *commits*... Entre sus características más relevantes se encuentran:

- Integración nativa con GitHub Enterprise, GitLab, BitBucket.
- Edición y visualización de ramas, merging, histórico de commits.
- Simplicidad a la hora de hacer un merge, rebase y push.
- Creación, clonación y añadir dispositivos de forma remota.
- Muestra la diferencia entre los ficheros con cambios respecto a lo ya cometido.
- Soporte de GitFlow, Git Hooks y LFS

4.9. LabelImg

LabelImg [21] es una herramienta gratuita *open source* que permite etiquetar gráficamente imágenes. Se encuentra escrita en Python y de cara a la interfaz gráfica usa QT. Esta herramienta permite el etiquetado en los froamto de texto **VOC, XML** o **YOLO.** Para llevar acabo el etiquetado de imágenes, lo primero ha realizar es la apertura de la carpeta contenedora de las imágenes (haciendo *click* en ***Open Dir***), seguidamente presioanremos la tecla 'w', para comenzar con la seleción de la región a etiquetar, marcaremos el área de la etiqueta y seguidamente nos preguntará el nombre de la clase a la que pertenece dicha etiqueta, creando a su vez un fichero (*classes.txt*) con todas las clases usadas.

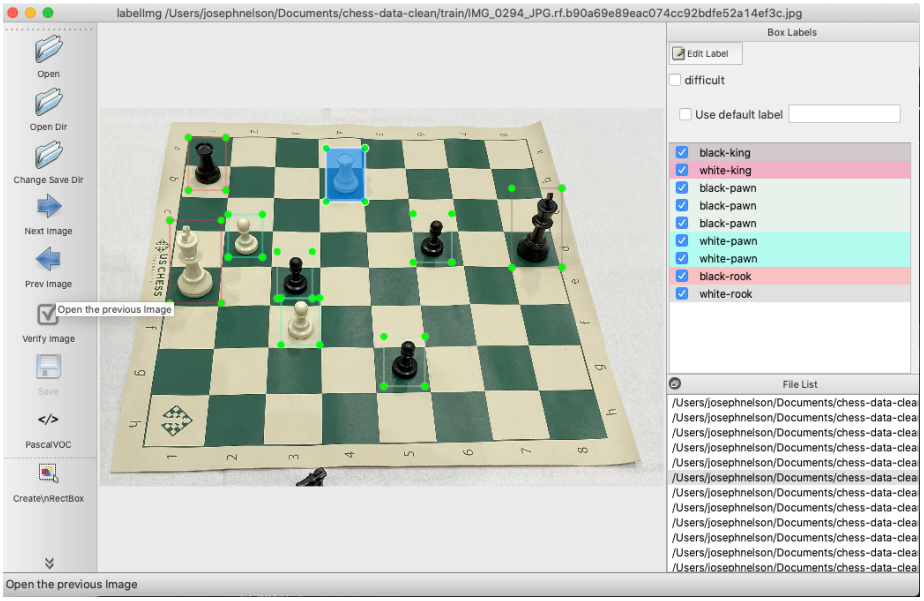

Figura 4.7: Funcionamiento de LabelImg

4.10. OIDv4 ToolKit

OIDv4 ToolKit [22] es un conjunto de herramientas, que permite obtener datos de *train* y de *validation* del *Open Images Dataset V4* Para descargar imágenes se usa el siguiente comando:

```
python main.py downloader --classes {nombre-clase} --type_csv {tipo-dato} --limit {número-imágenes}
```

Dónde *nombre-clase* representa la clase de la cuál se quieren descargar las imágenes(deber ser una clase reconocida por el dataset).

El campo *tipo-dato* indica si las imágenes van a ser utilizadas para *train* o para *validation*.

Por último, *número-imágenes* indica la cantidad de imágenes que se va a descargar para dicha clase.

Las imágenes descargadas con esta herramienta, ya vienen etiquetadas sólo quedaría convertilas al formato correcto del algoritmo que se vaya a usar. La información de la imagen que nos devuelve es la siguiente:

```
Nombre de la clase  Xmin  Ymin  Xmáx  Ymáx
```

Para *normalizarlas* al formato válido de *YOLO*, hay que realizar la siguiente conversión:

```
# Restar Xmin a Xmáx
coords[2] -= coords[0]
# Restar Ymin a Ymáx
coords[3] -= coords[1]
# Calcular las diferencias
x_diff = int(coords[2]/2)
y_diff = int(coords[3]/2)
# Sumar a Xmin la diferencia entre X
coords[0] = coords[0]+x_diff
# Sumar a Ymin la diferencia entre Y
coords[1] = coords[1]+y_diff
#Dividir Xmin entre el width de la imagen
coords[0] /= int(image.shape[1])
#Dividir Ymin entre el height de la imagen
coords[1] /= int(image.shape[0])
#Dividir Xmáx entre el width de la imagen
coords[2] /= int(image.shape[1])
```

```
#Dividir Ymáx entre el height de la imagen
coords[3] /= int(image.shape[0])
```

4.11. Google Colab

Google Colab [19] es una herramienta gratuita desarrollada por Google, esta se enecuentra alojada en la nube y basada en *Jupyter Notebook*. Los *notebook*, se encuentran formados por celdas, dónde cada una de de ellas puede contener código, texto, imágenes... Colab conecta el *notebook* a un entorno *runtime*, de tal forma que carga todas las aplicaciones de un programa y la ejecuta desde la nube. Pudiendo ejecutar el código Python sin necesidad de tenerlo instalado previamente, ya que este se encuentra instalado en el entorno de ejcución alojado en la nube. Además, cuenta con acceso gratuito a una GPU, lo suficientemente potente para la realización de ciertos trabajos, entre ellos, el entremiento de modelo de detección.

Aspectos relevantes del desarrollo del proyecto

En esta sección van a ser detallados los aspectos más relevantes que han tenido lugar a lo largo del desarrollo del proyecto. Al tratarse de un desarrolo *software*, y a lo largo de este se han encontrado diferentes retos e inconvenientes. A su vez, se han aplicado buenas prácticas con el objetivo de obtener un resultado de calidad.

5.1. Investigación

El eje principal que posee el proyecto es el *Object Detection*, el cuál a pesar de ser conocido, nunca se había trabajado con él, se puede decir que su conociemnto era puramente teórico. En la misma línea, se desconcoia completamente el funcionamiento de la *Jetson Nano*, la cuál a trabajar de forma intenra con *Ubuntu* ha supuesto una gran ayuda y beneficio al ser un sistema operativo previamente conocido. Por todo esto, ha supuesto un doble esfuerzo, especialmente por el primero de ellos, ya que se ha requerido una formación previa par poder conocer el correcto funcionamiento a la hora de entrenar un modelo de detección, así como averiguar cuál era la mejor opción de cara al proyecto. De cara al objetivo de obtener un mator conocimento acerca de *Object Detection* se leyeron diferentes *papers* [23, 24, 25], obteniendo de ellos los suficientes conocimientos sobre su funcionamiento, la calidad de la detección y el redimiento.

5.2. Metodología *Scrum*

Tal y coo se comento en el punto 4.1, el proyecto se ha reliazado siguiendo una metodología ágil, lo cuál nos permite trabajar con *sprints*, de tal manera, que el trabajo a relaizar en cada uno de ellos, se encuentre documentado desde el inicio, para así poder con una mayor eficiencia, pudiendo priorizar las tareas en función de las existentes y contando con diferentes versiones según a la vez que se va siguiendo el desarrollo del proyecto.

Los conocimeientos que se poseían de *Scrum* eran más bien teóricos, pero si que se había tenido la oportunidad de trabajar con ella, durante la realización de las prácticas en empresa.

Una de las principales dificultades encontradas, ha sido la estimación del tiempo que va a ser necesario para pdoer llevarlo a cabo, todo ello apartir de un nombre y descripción, lo cuál ha concluido con resultados no muy exactos, ya que se en algunos casos la estiamción se ha realizado por debajo y en otros por arriba. Si a esto se le suma que ciertas tareas han llevado más tiempo del debido a problemas versiones y de compatibilidades entre sistemas, hay que suamrle que al contar con un determiando número de asiganturas más la realización de manera paralela de las prácticas, ha conllevado que tareas que de normal llevan pcoo tiempo, han llevado más de lo planeado.

5.3. Creación del dataset de entrenamiento y entremiento del modelo de detección

A lo largo del Grado, no se ha visto nada relacionado con la **Inteligencia Artificial**, siendo esto una desventaja a la hora de crear el dataset del modelo y etiquetarlo con las características correspondientes al algoritmo de detección escogido. Lo que ha conllevado una inversión de horas, de cara a la búsqueda de infromación, así como de las diferentes formas para entrenar el modelo, y con ello la más asequible de cara a requisitos técnicos, debido a que llevar a cabo un entrenamiento de estas características, requiere de una gráfica potente, así como un equipo los suficientemente potente para llevarlo a cabo, ya que el mdoelo necesita estar un mínimo de horas entrenando para que el resultado sea medianamente decente y ser funcional. Debido a esto, las labores de entremiento, se han realizado sobre **Google Colab**, visto en el punto 4.11.

Trabajos relacionados

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

Conclusiones y Líneas de trabajo futuras

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

Bibliografía

[1] IBM, "Qué es deep learning." [Online]. Available: https://www.ibm.com/es-es/cloud/watson-studio/deep-learning

[2] R. Hat, "Qué es edge computing," 2021. [Online]. Available: https://www.redhat.com/es/topics/edge-computing/what-is-edge-computing

[3] M. Computer, "Nvidia jetson nano, una raspberry pi para ia," 2019. [Online]. Available: https://www.muycomputer.com/2019/03/19/nvidia-jetson-nano/

[4] E. A., "Detección de objetos con yolo: implementaciones y como usarlas." [Online]. Available: https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246

[5] R. Alake, "Deep learning: Googlenet explained," 2020. [Online]. Available: https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765

[6] Mathworks, "What is object detection?" [Online]. Available: https://es.mathworks.com/discovery/object-detection.html

[7] Proaglist, 2016. [Online]. Available: https://proagilist.es/blog/agilidad-y-gestion-agil/agile-scrum/los-11-pasos-para-implementar-metodologia-scrum/

[8] Oracle, "¿qué es python?" [Online]. Available: https://developer.oracle.com/es/python/what-is-python/

[9] MathWorks, "Matlab." [Online]. Available: https://es.mathworks.com/products/matlab.html

[10] Nebulova, "Predicción con redes neuronales convolucionales (cnn)," 2021. [Online]. Available: https://www.nebulova.es/blog/redes-neuronales-convolucionales

[11] S. Neelam, "Introduction to object detection with rcnn family models," 2021. [Online]. Available: https://medium.com/analytics-vidhya/introduction-to-object-detection-with-rcnn-family-models-310558ce2033

[12] R. Gandhi, "R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms," 2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[13] T. Karmarkar, "Region proposal network (rpn) — backbone of faster r-cnn," 2018. [Online]. Available: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9

[14] T. School, "¿qué es tensorflow?" 2020. [Online]. Available: https://www.tokioschool.com/noticias/que-es-tensorflow/

[15] OpenWebinars, "¿qué es tensorflow?" 2017. [Online]. Available: https://openwebinars.net/blog/que-es-tensorflow/

[16] J. Nelson, "What is tensorrt?" 2021. [Online]. Available: https://blog.roboflow.com/what-is-tensorrt/

[17] Xataka, "Qué es github y qué es lo que le ofrece a los desarrolladores," 2019. [Online]. Available: https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

[18] OpenWebinars, "Qué es visual studio code y qué ventajas ofrece," 2022. [Online]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

[19] A. Land, "Google colab: trabajando machine learning en la nube utilizando python," 2022. [Online]. Available: https://avertigoland.com/2022/04/google-colab-trabajando-machine-learning-en-la-nube-utilizando-python/

[20] SonarCloud, "Save time and effort with higher code quality." [Online]. Available: https://sonarcloud.io/code-quality

[21] J. Nelson, "Labelimg for labeling object detection data," 2020. [Online]. Available: https://blog.roboflow.com/labelimg/

[22] Ganesh, "How to prepare your own customized dataset using open images dataset v4," 2019. [Online]. Available: https://medium.com/@c.n.veeraganesh/how-to-prepare-your-own-customized-dataset-using-open-images-dataset-v4-8dfce9b9e147

[23] A. R. Pathak, M. Pandey, S. Rautaray, A. R. Pathak, . M. Pandey, . S. Rautaray, M. Pandey, and S. Rautaray, "Deep learning approaches for detecting objects from images: A review," *Advances in Intelligent Systems and Computing*, vol. 710, pp. 491–499, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-7871-2_47

[24] G. Yang, Q. Luo, Y. Yang, and Y. Zhuang, "Deep learning and machine learning for object detection in remote sensing images," *Lecture Notes in Electrical Engineering*, vol. 473, pp. 249–256, 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-7521-6_30

[25] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 7 2020.