UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Máster Universitario en Ingeniería y Ciencia de Datos

Agricultura de precisión

Optimización del uso de herbicidas mediante visión artificial

Luis Enrique Pina Herce

Dirigido por: Rafael Pastor Vargas

Febrero 2023

Resumen

Durante miles de años, los agricultores han buscado formas de aumentar la producción de alimentos en parcelas. A medida que los equipos y la tecnología han evolucionado, las granjas se han vuelo más grandes y los rendimientos han aumentado. Sin embargo, este desafío continua hoy en día y la versión moderna ha recibido un nombre: Agricultura de precisión.

La presente memoria busca estudiar una de las ramas a explotar en la agricultura de precisión, la aplicación de herbicidas, mediante el uso de la visión artificial. El proyecto se centra en investigar un modelo rápido y eficiente para la detección de mala hierba en imágenes. Este modelo es la pieza principal de un sistema que recibe imágenes del suelo en tiempo real, y en función de lo que ve, aplica o no el herbicida sobre el suelo.

Abstract

For thousands of years, farmers have sought ways to increase food production on plots of land. As equipment and technology have evolved, farms have become bigger and yields have increased. However, this challenge continues today and the modern version has been given a name: Precision Agriculture.

The present report has the aim to study one of the branches to be exploited in precision agriculture, the application of herbicides, using artificial vision. The project focuses on investigating a fast and efficient model for weed detection in images. This model is the main piece of a system that receives images of the soil in real time, and depending on what it sees, it applies or not the herbicide on the soil.

Tabla de contenido

Introducción a la agricultura de precisión

La agricultura de precisión se entiende como la aplicación de nuevas tecnologías de la información a tareas agrícolas con el objetivo de mejorar la productividad de los cultivos y disminuir el impacto medioambiental. Esta conlleva una estrategia de manejo dirigida a incrementar la productividad y los retornos económicos con un impacto reducido en el medio ambiente.

Desde una perspectiva económica, la agricultura de precisión podría definirse simplemente como producción agrícola eficiente, o como el uso correcto de insumos en el tiempo y en el espacio

Estas definiciones nos sugieren múltiples ventajas tanto para el productor como para la economía global. (Valdés, H., 2017)

- Aumento de los rendimientos. Un estudio realizado en 2012 nos revela que sería posible tener un aumento de hasta un 30% en la producción global.

- Mayores rendimientos significa mayor cantidad de alimentos. Mayor seguridad alimentaria y, mayores ganancias para los agricultores.

- Beneficios ambientales. Entre ellos:

    - Aumento de producción sin necesidad de expandir el área agrícola, lo que implica menos deforestación y agotamiento de recursos naturales.

    - La reducción de productos fertilizantes, herbicidas y otros agroquímicos puede generar enormes beneficios térmicos de reducciones de gases de efecto invernadero, así como la reducción en la contaminación de suelos y masas de agua por la escorrentía proveniente de los cultivos.

    - El uso preciso de agua implica un ahorro en este recurso tan escaso.

- Sanidad agrícola. Si se logra implementar de manera frecuente un monitoreo de los cultivos, por ejemplo, con drones y otras técnicas de teledetección, la incidencia de las plagas y otras enfermedades se puede controlar de manera más rápida, facilitando así la contención de la epidemia.

Por lo general, la agricultura de precisión involucra el uso de la más avanzada maquinaria, algunas fincas son enormes por lo que usan tractores sin conductor dirigidos por GPS en tiempo real con una precisión de hasta 2 cm. Asimismo, exige recolectar grandes cantidades de datos, tanto en el espacio como en el tiempo para poder analizarla y poder determinar las acciones necesarias para maximizar el rendimiento de la parcela.

Es por todo esto por lo que la agricultura de precisión puede llegar a ser una tarea costosa y difícil.

- Los equipos y maquinaria asociados son costosos, si bien su precio continúa bajando.
- Su operación y mantenimiento requieren competencias especializadas.
- La interpretación de la información proveniente del terreno y de otras fuentes es una tarea compleja que consume tiempo y que puede requerir la contratación de los servicios de una firma especializada.
- Los mercados de insumos, maquinaria, partes y mano de obra calificada deben desempeñarse eficientemente para que la agricultura de precisión pueda funcionar y ser rentable.

Es un hecho que la unión europea está apostando muy fuerte por ir adoptando progresivamente el uso de la agricultura de precisión. Si nos fijamos en los 10 países principales consumidores de pesticidas de la unión europea, vemos que España encabeza el ranking seguido de Francia e Italia. (Schrijver, R et all., 2016) (Rodriguez, M et all., 2018)

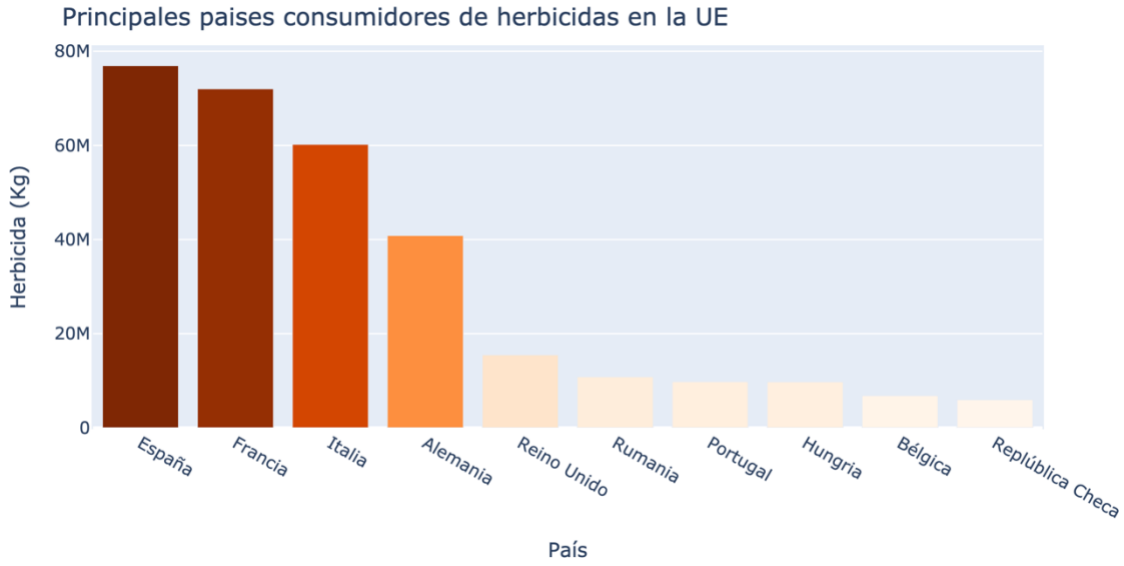


Ilustración 1. Principales países consumidores de herbicida (Rodriguez, M et all.,, 2018)

Teniendo en cuenta lo perjudicial que son estos pesticidas para el medio ambiente, se debe minimizar su uso en la medida de lo posible en los próximos años.

Actualmente la agricultura de precisión es un sector muy atrasado tecnológicamente con respecto al resto de sectores. Esto puede deberse a que la media de edad de los agricultores es alta, y, por tanto, no confían en la tecnología y prefieren los métodos tradicionales.

Las principales tareas para las que se usa la agricultura de precisión hoy en día, son las siguientes (Hortalizas, 2018):

- Drones para las Inspecciones en Campo. Los vehículos aéreos no tripulados (UAV) que vuelan a 40 pies de altura a un poco menos de 30 millas la hora, pueden cubrir 40 hectáreas en 20 minutos y revisan con facilidad cientos de hectáreas al día, capturando cientos de imágenes. Estos ojos aéreos ayudan a detectar cambios sutiles en los cultivos, como deficiencia de nitrógeno o falta de riego.

- Drones pulverizadores. Existen drones con depósitos de hasta 40 Litros que permiten pulverizar tanto herbicidas como semillas de pequeño tamaño en lugares donde no podría llegar un tractor.

- Tractores no tripulados. Los tractores controlados por computadora utilizan sistemas automatizados de direccionamiento de alta precisión basados en GPS. Con los avances en la tecnología de direccionamiento, los tractores auto-impulsados incluirán un sistema "inteligente para evitar obstáculos".

- Dirección automática. Los suelos compactados reducen el rendimiento, pero darles seguimiento a las huellas dejadas por el tractor reduce la compactación entre las hileras del cultivo. El tractor automático combinado con la tecnología de dirección está siendo extrapolada para colocar ruedas autodirigidas.

- Manejo de las flotillas. Los productores más grandes han copiado la tecnología que usan los transportistas comerciales para rastrear sus flotillas por medio de las pantallas de las computadoras. La telemática muestra la ubicación de los vehículos de manera ininterrumpida.

- Manejo del riego. Se utilizan plataformas de manejo remoto del riego para simplificar de manera drástica los sistemas y automatizar el uso de metodologías de programación del riego, ayudando a que los productores decidan cuándo, dónde y cuánto regar.

- Sensores. La tecnología avanza a pasos agigantados en la detección en tiempo real por medio de dispositivos montados en vehículos que controlan la aplicación y exactitud de insumos tales como agroquímicos y fertilizantes, además de realizar las pruebas necesarias para obtener la calidad requerida y modificar cuando sea necesario la cantidad, porcentaje, velocidad y profundidad de aplicaciones.

- Nivelación del suelo por láser. Los terrenos poco uniformes afectan la absorción del agua. Las superficies planas garantizan que el agua llegue a

todas las partes del campo de cultivo con un mínimo de escurrimientos y anegación. Sin los escurrimientos, ni el agua anegada que deja el riego por inundación, todo el campo de cultivo recibe los mismos volúmenes de agua, al mismo tiempo".

Problema a resolver

La aplicación de herbicidas líquidos es uno de los temas a explotar en la agricultura de precisión. La correcta aplicación de herbicidas es clave para mantener el cultivo libre de malas hierbas. Las pérdidas por bajo rendimiento en las campañas están asociadas en buena medida a la presencia de malezas que compiten con las plantas.

Método de aplicación actual

Los equipos de aplicación de herbicidas en cultivos extensivos se corresponden con máquinas que denominamos habitualmente, pulverizadores hidráulicos, equipos de barras, o máquinas para tratamientos en cultivos bajos. Estos equipos pueden ser accionados por un tractor, uniéndose a él en forma suspendida o semiarrastrada (según la capacidad del depósito de líquido), o pueden ser automotrices, para capacidades de trabajo elevadas. La finalidad de estos equipos es repartir el líquido herbicida sobre una "superficie objetivo", para lo que se exigen dos funciones:

- Transformar el líquido en gotas.
- Distribuir éstas adecuadamente.

La superficie objetivo puede ser plana, por ejemplo, un suelo desnudo, o irregular y profunda, representada por las hojas y otras estructuras de las plantas sobre las que se desea depositar el líquido.

La constitución básica de estos equipos incluye, un chasis que soporta el resto de componentes, un sistema hidráulico, formado por un depósito de almacenamiento, una bomba de desplazamiento positivo, un distribuidor del líquido impulsado, varios filtros, unas boquillas de pulverización hidráulica y las conducciones correspondientes para formar los circuitos del sistema, y una estructura metálica de soporte (conocida como barra) a la que se sujetan las conducciones que montan las boquillas y que, en posición de trabajo, colocan a las boquillas en posición horizontal; esta estructura dispone de un sistema de plegado para posibilitar su transporte en las vías públicas. Al analizar estas máquinas siempre se suelen valorar dos aspectos diferenciados, la seguridad y la eficiencia (Boto, J., 2010).

- La seguridad de un equipo está relacionada, tanto a los riesgos para las personas que realizan la aplicación, debido a que se manejan materias tóxicas y porque las máquinas en si son potencialmente peligrosas, como a la posible contaminación del medio que se puede provocar, ya sea al agua, al aire o a los productos expuestos a la aplicación. La seguridad de estas máquinas está reglamentada y es obligatoria, tanto en lo que se

refiere a la seguridad intrínseca de la máquina, de acuerdo con lo establecido en la "Directiva máquinas", como a su utilización, según la "Directiva de uso sostenible de plaguicidas".

- La eficiencia de un equipo está relacionada con la homogeneidad de reparto del líquido pulverizado sobre la superficie objetivo, la que, junto con el producto empleado, la dosis utilizada y el momento de la aplicación, condicionarán la efectividad del herbicida. A diferencia de la seguridad, no existe obligatoriedad en unas prestaciones mínimas que puedan afectar a la eficiencia de la máquina; si bien se puede resaltar que, actualmente, solo se garantizan unas prestaciones mínimas, en las máquinas que "voluntariamente" se han sometido y superado unas pruebas normalizadas, realizadas en algún Centro de la Red de Laboratorios ENTAM3 (European Network for Testing of Agricultural Machines), y aquellas que el Ministerio de Medio Ambiente, y Medio Rural y Marino tiene autorizadas para acogerse al Plan Renove.

Estos herbicidas se suministran de manera uniforme en toda la parcela sin tener en cuenta si se está aplicando sobre la mala hierba o no, es decir, no se aplica de manera selectiva. Tal y como se muestra en la siguiente imagen.

Ilustración 2. Tractor fumigando

Si reducimos la constitución básica del equipo en un diagrama básico:

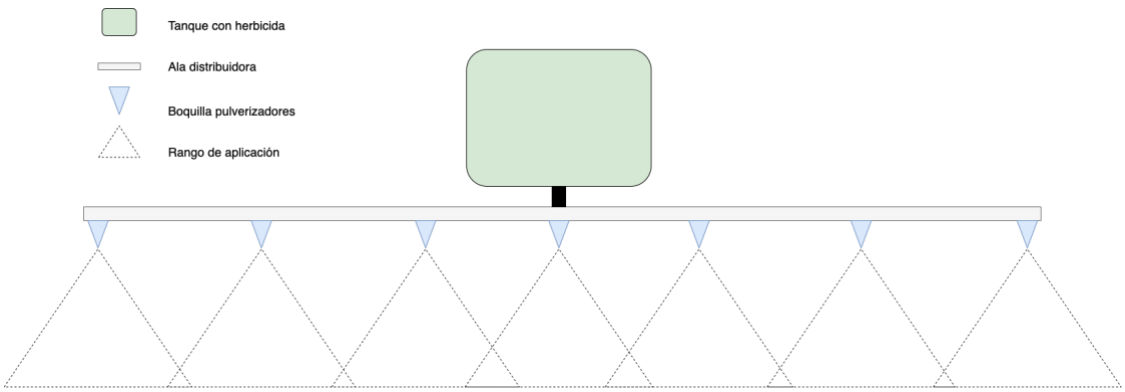


Ilustración 3. Diagrama constitución básica de un equipo de aplicación de herbicidas

Como se aprecia en el diagrama, el pesticida se almacena en el tanque que se distribuye por el ala y se libera a través de las boquillas pulverizadoras. Como se puede apreciar también, el rango de aplicación se solapa en algunos puntos, esto provoca que algunas partes de la parcela tengan más agroquímicos que otras.

Método de aplicación inteligente

Existen varias alternativas al método de aplicación tradicional.

- En tiempo real. Este método consiste en ir recorriendo la parcela y con forme se detecta mala hierba se aplica al instante el pesticida sobre ella. Donde no se detecta, no se aplica.

- Por procesos. En este caso constaría de 3 fases, la primera de ella orientada en la recogida de datos de la parcela, la segunda se enfoca en procesar estos datos con el fin de saber dónde hay que aplicar y la tercera fase consiste en la aplicación del pesticida en función de los resultados obtenidos en la anterior fase.

Ambos métodos pueden realizarse utilizando un dron pulverizador (ilustración 4) o un tractor convencional (ilustración 2) (T30 DJI, 2021).

Ilustración 4. Dron con deposito para pulverización. (T30 DJI, 2021)

No ha sido hasta hace muy pocos años en los que se han empezado a introducir técnicas de inteligencia artificial y Machine Learning para minimizar costes. Una de las empresas punteras y líderes en el sector es John Deere, la cual anunció en 2021 un modelo de maquinaria con la tecnología “See and Spray” (Mira y pulveriza), donde, a partir de sensores colocados en las alas de la pulverizadora, reconoce si hay mala hierba y pulveriza en caso afirmativo.

Ilustración 5. Pulverizador inteligente de John Deere. (John Deree, 2021)

Aseguran que se reduce hasta un 77% de herbicida utilizado gracias a este método de pulverización inteligente (John Deree, 2021). El problema es que casi ningún agricultor está dispuesto a cambiar su maquinaria actual por esta nueva con los elevadísimos costes que tiene (+100.000 €). Por lo tanto, la inmensa mayoría de los agricultores siguen utilizando sus pulverizadoras tradicionales, aplicando la misma cantidad de herbicida a toda su parcela.

Otro producto es Eco Sniper, de la empresa Milar Agro Tech, este es un sistema de aplicación selectiva, que funciona en barbecho o rastrojo, y que permite aplicar herbicidas específicamente sobre las malezas, evitando hacerlo sobre la totalidad del lote. “Cuando el dispositivo observa un objeto de coloración verde, da la orden de apertura de la válvula correspondiente realizando así la aplicación del producto herbicida puntualmente sobre la maleza hallada. Con su utilización pueden obtenerse ahorros de hasta un 80% en el uso de productos químicos”, cuenta Elgart. “A diferencia de otras tecnologías similares importadas, que trabajan con sensores, Eco Sniper funciona con cámaras ópticas en tiempo real lo que posibilitará en un futuro distinguir entre objetos de coloración verde. La evolución de este sistema permitirá realizar aplicaciones selectivas con cultivo implantado, diferenciando un objeto verde de interés (cultivo) de un objeto verde no deseado (maleza)”, agrega. “La eficiencia se basa en que la mayoría de los lotes en barbecho que se pulverizan con herbicidas tienen menos del 5% de cobertura verde, esto nos ha llevado a medir en promedio ahorros del 70% del producto químico en casos reales en lotes propios, de productor, ensayos con Regionales Aapresid, INTA y grupos CREA”, completa. (Berreta, J., 2020)

Ilustración 6. Eco Sniper. (Berreta. J, 2020)

Este último producto, actualmente solo diferencia colores, pulverizando así sobre cualquier tipo de hierba encontrada. Ya es un método más preciso en comparación a la aplicación de herbicida sobre toda la parcela (método tradicional).

Objetivo del TFM

Visto lo visto hasta ahora, parece que no sería muy complejo construir un sistema capaz de aplicar esta misma idea de "See and Spray" independientemente de la maquinaria utilizada por el agricultor y con unos costes muchísimo más asequibles.

El sistema montado sobre una maquinaria estándar, tendría un aspecto como el del siguiente diagrama (ilustración 7):

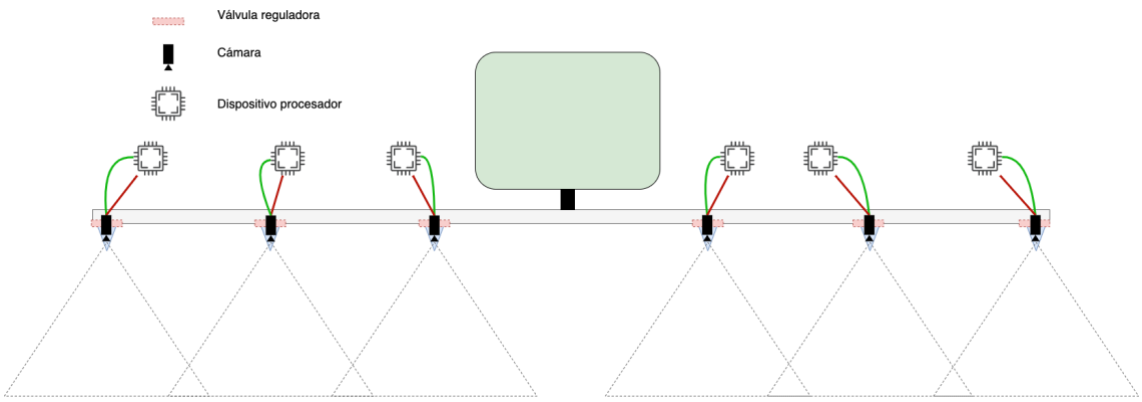


Ilustración 7. Prototipo de sistema inteligente

Cómo se puede observar, es el diagrama (diagrama 1) pero con algún añadido:

- Válvula reguladora: Tiene como única función bloquear o no el paso de líquido a la boquilla pulverizadora.
- Cámara: Recoge imágenes del suelo sobre el que pasa el tractor.
- Dispositivo procesador: Procesa las imágenes recogidas por la cámara y en función de lo que ve, abre o cierra la válvula.

Se trata de colocar un dispositivo en cada boquilla, que sea capaz de leer el suelo y abrir la boquilla cuando detecte mala hierba sobre el suelo leído. Para este sistema, las cámaras deben abarcar la misma cantidad de suelo que la que abarca el líquido pulverizado por las boquillas pulverizadoras, de tal forma que el procesador ira haciendo lecturas constantes del suelo y abrirá la boquilla siempre que detecte mala hierba sobre la imagen procesada.

Vista de perfil:

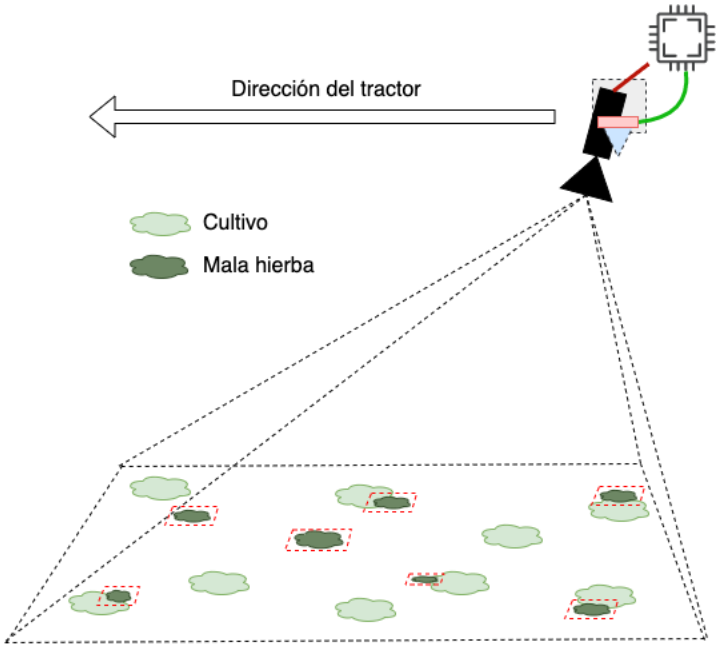


Ilustración 8. Vista de perfil del prototipo

Vista de frente:

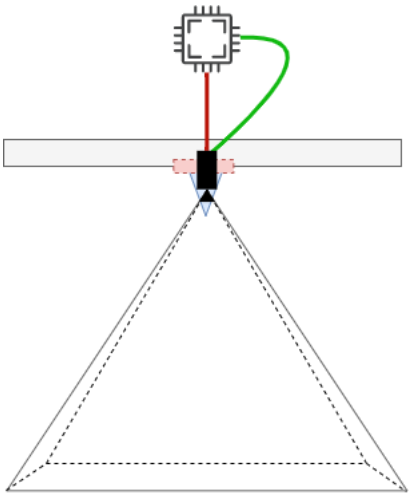


Ilustración 9. Vista de frente del prototipo

Este trabajo de fin de master tiene por objetivo estudiar y desarrollar el algoritmo de detección de mala hierba sobre cultivo de girasol en tiempo real que se procesará en los dispositivos del diagrama, este algoritmo proporcionará un valor de probabilidad que indique la apertura de la válvula o no. Es decir, tener como input una imagen y dará de output si es necesario abrir la boquilla o no.

Aplicación DL/ML

Se debe desarrollar una aplicación DL/ML capaz de ser ejecutada en dispositivos de bajo coste, pues buscamos que el dispositivo final tenga un coste muy asequible para el usuario final. La aplicación como tal, recibirá imágenes como input (obtenidas por la cámara en tiempo real) y enviará una señal a la boquilla que debe abrirse para pulverizar el herbicida sobre el objetivo (mala hierba).

El sistema deberá contar con un modelo de DL/ML ya entrenado capaz de reconocer mala hierba en una imagen y ser ejecutado en dispositivos de bajo coste.

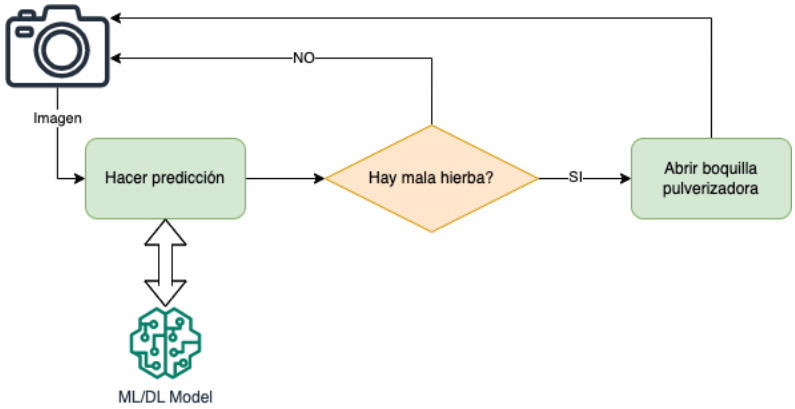


Ilustración 10. Aplicación DL/ML

La dificultad de la aplicación viene a ser la detección de objetos dentro de una imagen, para ello utilizaremos técnicas Deep Learning.

Estado del arte de la visión artificial

La visión artificial se podría definir como una disciplina dentro de la inteligencia artificial especializada en sistemas que adquieren, procesan y analizan imágenes reales para generar información que pueda asimilar una máquina. De la misma forma que las personas usan el sentido de la vista para comprender el mundo que las rodea, esta tecnología reproduce el mismo principio en un entorno industrial. Para ello, descompone las imágenes en pequeños fragmentos, conocidos como píxeles, con el fin de que una máquina las "entienda" y, por tanto, tome decisiones autónomamente. (Sick, CNN-ibm)

Un sistema de visión artificial requiere una cámara (el "ojo") y un Procesador (el "cerebro") para funcionar. La primera envía la información que capta al segundo, una especie de centro de toma de decisiones en el que se interpretan los datos para que la máquina actúe en consecuencia.

Ventajas

La visión artificial mejora la calidad y la producción, a la vez que disminuyen los costos de producción. Mientras que la visión humana es la mejor para la interpretación cualitativa de escenas complejas no estructuradas, la visión artificial sobresale en la medición cuantitativa de escenas estructuradas gracias a su velocidad, precisión y replicabilidad. Por ejemplo, en una línea de producción, un sistema de visión artificial puede inspeccionar cientos, incluso miles, de piezas por minuto. Un sistema de visión artificial desarrollado en base a la resolución de una cámara y óptica adecuadas puede inspeccionar fácilmente aquellos detalles de un objeto que son demasiado pequeños para ser detectados por el ojo humano.

Algunas de estas ventajas podrían ser:

- Precisión: Aumentan el nivel de precisión durante la fabricación y eliminan los errores humanos.
- Calidad: Garantizan la máxima calidad y que los productos cumplan con las especificaciones.
- Rentabilidad: Aumentan los beneficios y reducen los gastos generales, ofreciendo un alto ROI.
- Sostenibilidad: Permiten fabricar productos ahorrando recursos y reduciendo los desperdicios.
- Monitorización: Facilitan el diagnóstico y la resolución de problemas durante la producción.
- Seguridad: Mejoran la seguridad y la protección en entornos peligrosos para las personas.

Clasificación

Podemos hacer una clasificación dependiendo de la calidad de la imagen:

- Visión 1D: Opera con una señal digital que estudia la imagen línea a línea y no en su totalidad. Se emplea para detectar defectos en materiales fabricados en un proceso de flujo continuo, donde los materiales están en constante movimiento y son sometidos a un tratamiento mecánico (metales, papel, tejidos, plásticos…).
- Visión 2D: Las cámaras 2D pueden captar imágenes en color o en escala de grises y las convierten en matrices divididas en píxeles, por eso se utilizan en el reconocimiento de caracteres como en la lectura de códigos o, incluso, en metrología.

- Visión 3D: Estos sistemas recurren a varias cámaras colocadas en distintos lugares o sensores de láser por lo que reúnen información sobre la orientación del elemento. Suelen usarse en pick and place.

- Termografía: A través de una serie de sensores capta información sobre la temperatura de las piezas y se representa en una escala de colores. Es la alternativa más recomendable a la hora de detectar fugas o desequilibrios.

- Visión hiperespectral: Estas cámaras identifican la composición de cada elemento midiendo la longitud de onda. En alimentación se recurre a ellas para detectar intrusos como restos de tierra, plásticos…

Tareas habituales

Destacar que la visión artificial como cualquier inteligencia artificial, son muy buenas en cumplir tareas específicas, no para hacer cualquier cosa.

Entre las tareas más habituales que se realizan con visión artificial se pueden encontrar las siguientes:

- Detección de objetos: Es la capacidad de detectar o identificar objetos en una imagen determinada correctamente junto con su posición espacial en la imagen dada, en forma de cuadros rectangulares (conocidos como cuadros delimitadores) que limitan el objeto dentro de ella.

- Clasificación de imágenes: Básicamente significa identificar a qué clase pertenece el objeto identificado.

- Detección de relaciones visuales: Es la capacidad de descifrar qué relación comparten los dos objetos.

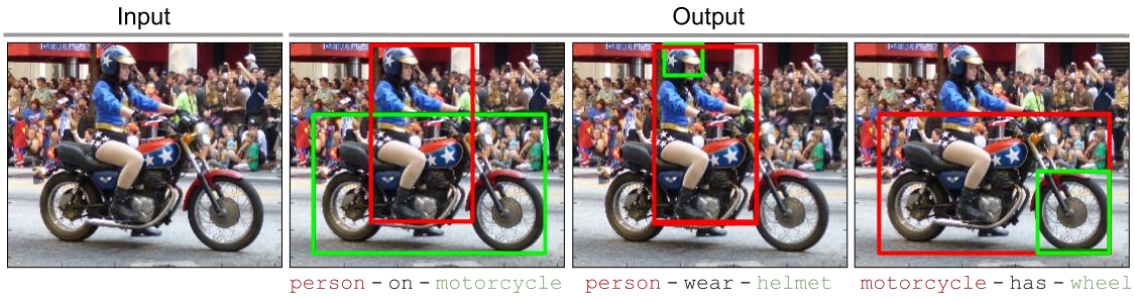


Ilustración 11. Detección de relaciones visuales (Lu, C., 2020)

- Descripción de imágenes: Describir lo que está sucediendo en la imagen mediante anotaciones, etiquetas o descripciones.

- Reconstrucción de imágenes: Es la capacidad de identificar lo que falta en una imagen para poder reconstruirla.

- Reconocimiento facial.

- Segmentación de instancias: Esta tarea, identifica instancias dadas en una imagen con sus límites intactos.

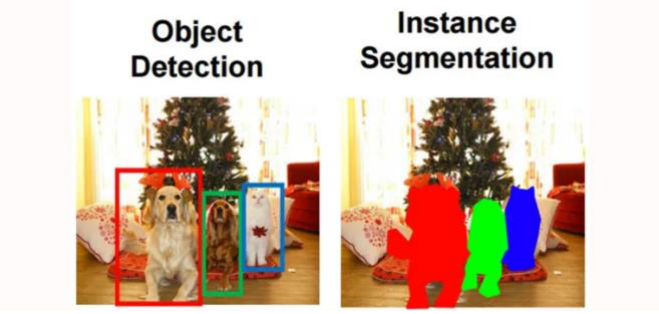


Ilustración 12. Segmentación de instancias vs detección de objetos. (Banerjee, A., 2019)

- Segmentación semántica: Trata de identificar objetos similares en el objeto que pertenecen a la misma clase a nivel de píxel. Por ejemplo, la imagen a continuación muestra que el modelo ha tratado de identificar objetos similares como árboles, llantas en la bicicleta, césped, edificios, etc. y los ha codificado con el mismo color para simbolizar que pertenecen a la misma clase.

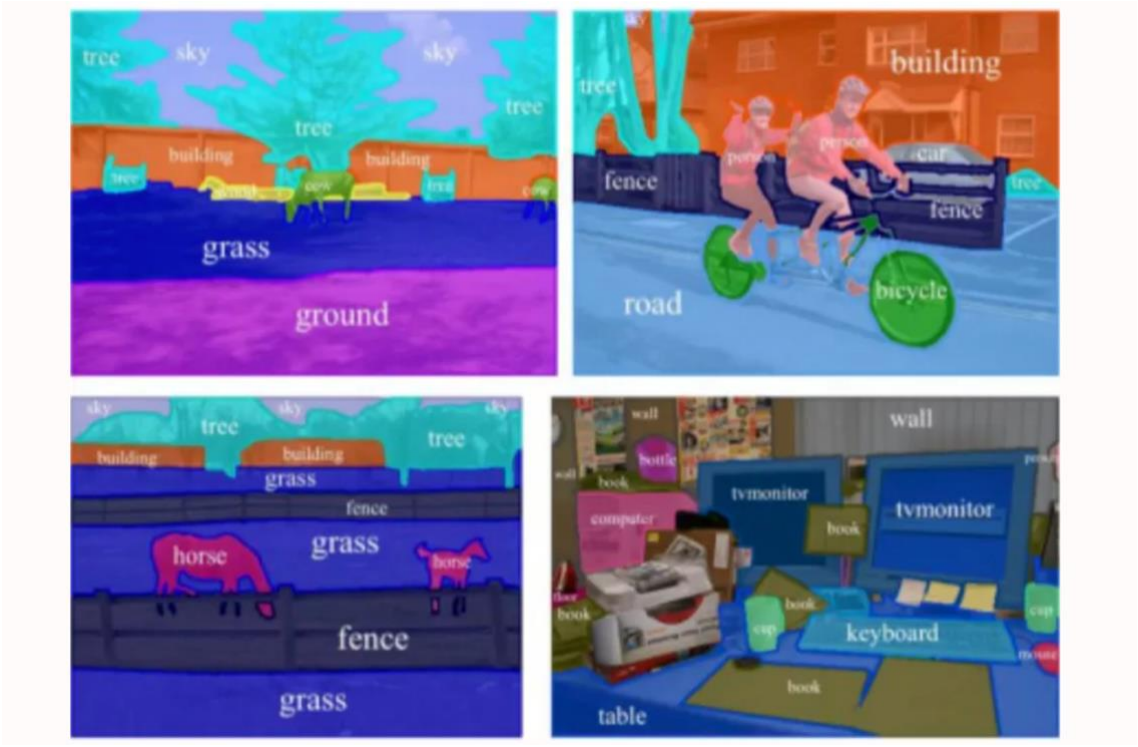


Ilustración 13. Segmentación semántica (Banerjee, A., 2019)

Ejemplos de aplicación de la visión artificial

En procesos industriales como una cadena de montaje podemos encontrar los siguientes ejemplos:

- Sistemas *pick and place*: Realizan tareas consistentes en coger un producto y depositarlo en otro lugar. Hay que tener en cuenta que esos productos no siempre avanzan de modo ordenado, por lo que este tipo de robots deben ser capaces de identificar el producto y su posición y calcular trayectorias y velocidad.

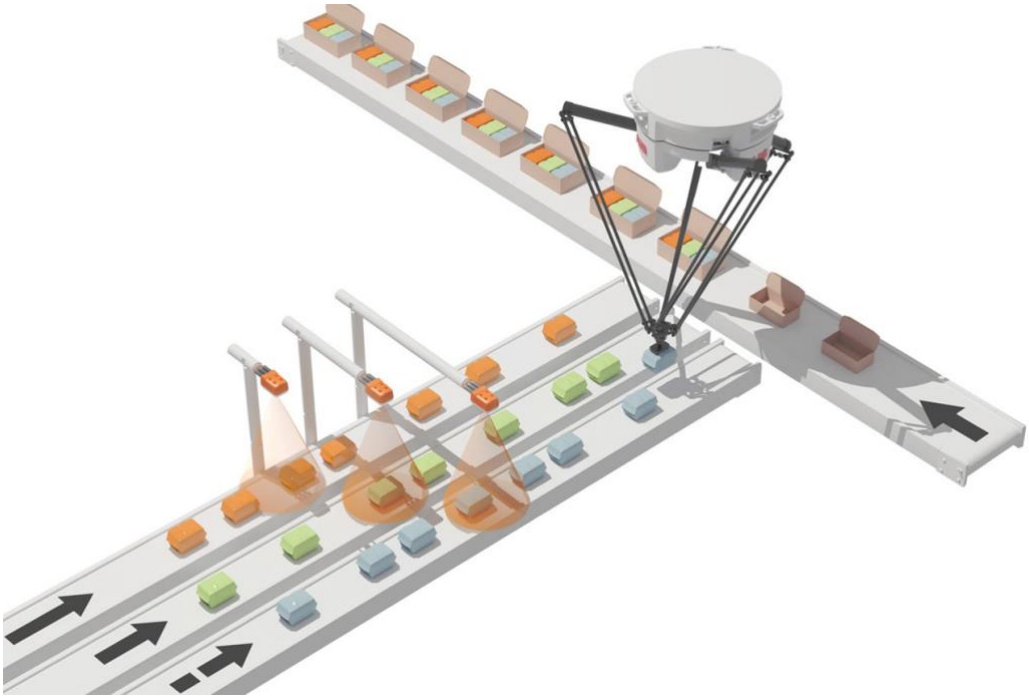


Ilustración 14. Sistema pick and place. (Edimar)

- Comprobación de montajes: Los sistemas de visión artificial al final de una línea de ensamblaje pueden comprobar rápidamente que cada pieza ocupa el lugar que le corresponde.

- Sistemas de medición: La visión artificial garantiza la precisión exigida para realizar cualquier medición (tamaños, distancias, ubicaciones…) siempre que se haya realizado un ajuste referencial.

- Localización de defectos: En cualquier proceso de producción aparecen defectos, bien por el propio proceso en sí o por las condiciones a las que ha estado expuesto el producto. Las cámaras de visión artificial permiten detectar estos defectos de manera automatizada y rápidamente, retirando de la línea de producción la pieza (puede ser un alimento en mal estado o que presente algún golpe).

- Lectura de códigos: La trazabilidad de un producto se basa en el uso de diferentes códigos. Los más comunes son los códigos de barras, pero

también se emplean códigos matriz 2D y OCR (Optical Character Recognition). La visión artificial, gracias al sistema de algoritmos de reconocimiento de caracteres, facilita su lectura, comprobando que coincida con el producto.

- Detección de elementos intrusos: Una función especialmente útil en el sector de la alimentación. La visión hiperespectral permite clasificar elementos atendiendo a su composición química, por lo que resulta fácil detectar un intruso en la fase de empaquetado de alimentos, por ejemplo.

En el campo agroalimentario podríamos encontrar los siguientes ejemplos:

- Análisis del punto de maduración de frutas y hortalizas: La visión artificial permite conocer el punto de maduración de una fruta o vegeta. La identificación mediante cámaras se puede realizar tanto en árbol, permitiendo así identificar si éstas se pueden recolectar, o en almacén, para decidir el destino más óptimo para el producto ya recolectado.

Ilustración 15. Detección punto de maduración mediante visión artificial (Xueping Ni et all, 2020)

- Detección de malezas: Dado que los sistemas de visión artificial escanean granjas con frecuencia, las plantas adventicias se pueden detectar prácticamente en tiempo real. La visión artificial también ayuda a detectar la efectividad de los herbicidas empleados y encontrar malezas resistentes en los cultivos.

- Control de plagas y enfermedades: Mediante la visión artificial es posible identificar y categorizar las principales enfermedades relevantes, así como detectar daños físicos causados por insectos y plagas casi en el momento en el que se originan.

- Categorización y precio de compra para materias primas naturales: La determinación del precio de materia prima como la uva, oliva, fruta, etc. para muchas de las industrias alimentarias depende de una inspección visual, juntamente con una cata para determinar el precio de compra. La visión por computador y la inteligencia artificial permiten realizar esta inspección de forma más ágil y precisa determinando así el grado de

maduración, posibles defectos estéticos o incluso el porcentaje de agua de los productos.

- Supervisión y diagnóstico de la salud del ganado: El software de visión artificial también es útil para realizar un seguimiento del crecimiento y las características físicas del ganado. Las vacas se escanean individualmente utilizando un sensor 3D y se miden sus lomos durante su alimentación mediante un sensor de tiempo de vuelo, combinado con procesamiento de imágenes en 3D. De esta forma se pueden determinar sus características físicas con gran precisión y obtener datos valiosos sobre su estado de salud.

Redes neuronales

Los pájaros nos inspiraron a volar, las plantas de bardana inspiraron el velcro y la naturaleza ha inspirado innumerables inventos más. Parece lógico, entonces, observar la arquitectura del cerebro en busca de inspiración sobre cómo construir una máquina inteligente. Esta es la lógica que desencadenó las redes neuronales artificiales (Artificial Neural Networks, ANN), modelos de aprendizaje automático inspirados en las redes de neuronas biológicas que se encuentran en nuestros cerebros. Sin embargo, aunque los aviones se inspiraron en las aves, no es necesario que aleteen para volar. De manera similar, las ANN se han vuelto gradualmente bastante diferentes de sus primos biológicos. Algunos investigadores incluso argumentan que deberíamos abandonar la analogía biológica por completo (p. ej., diciendo "unidades" en lugar de "neuronas"), para no restringir nuestra creatividad a sistemas biológicamente plausibles.

Las ANN están en el centro mismo del aprendizaje profundo. Son versátiles, potentes y escalables, lo que los hace ideales para abordar tareas de aprendizaje automático grandes y muy complejas, como clasificar miles de millones de imágenes (p. ej., Google Images), potenciar los servicios de reconocimiento de voz (p. ej., Siri de Apple), recomendar los mejores videos a ver a cientos de millones de usuarios todos los días (por ejemplo, YouTube), o aprender a vencer al campeón mundial en el juego de Go (AlphaGo de DeepMind).

Una red neuronal básica tiene neuronas artificiales interconectadas en tres capas: (Amazon- What is a neural network?)

- Capa de entrada: La información del mundo exterior entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa.
- Capa oculta: Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa.
- Capa de salida: La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos. Por ejemplo, si tenemos un problema de clasificación binaria (sí/no), la capa de salida tendrá un nodo de salida que dará como resultado 1 o 0. Sin embargo, si tenemos un problema de clasificación multiclase, la capa de salida puede estar formada por más de un nodo de salida.

Si nos fijamos en las redes neuronales utilizadas para el procesamiento de imágenes, encontramos los siguientes tipos:

Convolutional neural networks (CNN)

Se distinguen de otras redes neuronales por su rendimiento superior con entradas de señales de imagen, voz o audio. Tienen tres tipos principales de capas, que son:

- Capa convolucional
- Capa de agrupación
- Capa totalmente conectada (Fully Connected, FC)

La capa convolucional es la primera capa de una red convolucional. Si bien las capas convolucionales pueden ir seguidas de capas convolucionales adicionales o capas de agrupación, la capa totalmente conectada es la capa final.

Con cada capa, la CNN aumenta en su complejidad, identificando porciones más grandes de la imagen. Las capas anteriores se enfocan en características simples, como colores y bordes. A medida que los datos de la imagen avanzan a través de las capas de la CNN, comienza a reconocer elementos o formas más grandes del objeto hasta que finalmente identifica el objeto deseado.

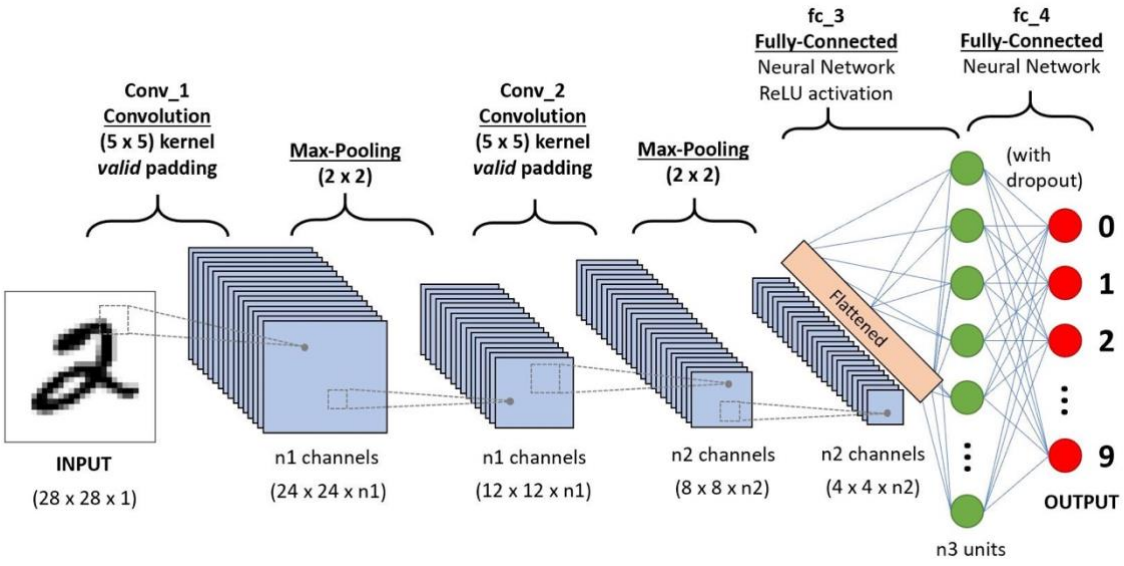


Ilustración 16. CNN para clasificación de números (Sumit, S., 2018)

Region based convolutional neural networks (R-CNN)

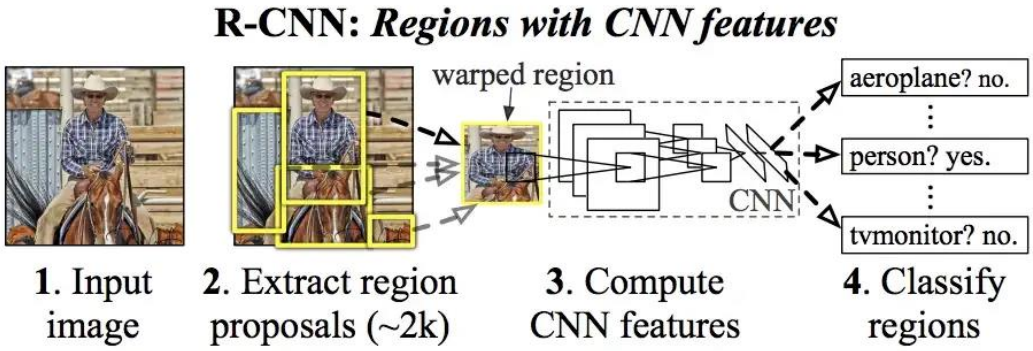


Ilustración 17. Regions with CNN features (Gandhi, R., 2018)

Se obtienen 2000 propuestas de regiones candidatas que se deforman en un cuadrado y se alimentan a una red neuronal convolucional que produce un vector de características de 4096 dimensiones como salida. La CNN actúa como un extractor de características y la capa densa de salida consta de las características extraídas de la imagen, las cuales alimentan a una SVM para clasificar la presencia del objeto dentro de esa propuesta de región candidata.

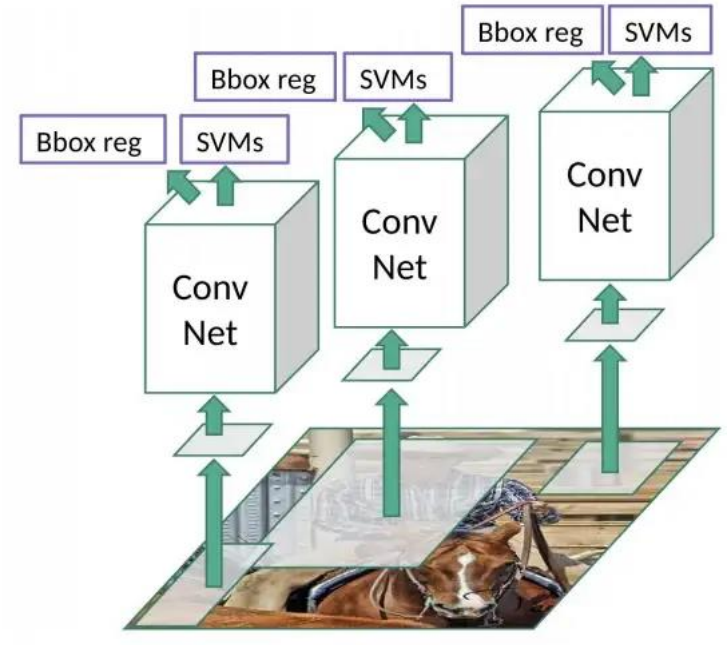


Ilustración 18. R-CNN extractor de características y SVM (Gandhi, R., 2018)

Fast region based convolutional neural networks (Fast R-CNN)

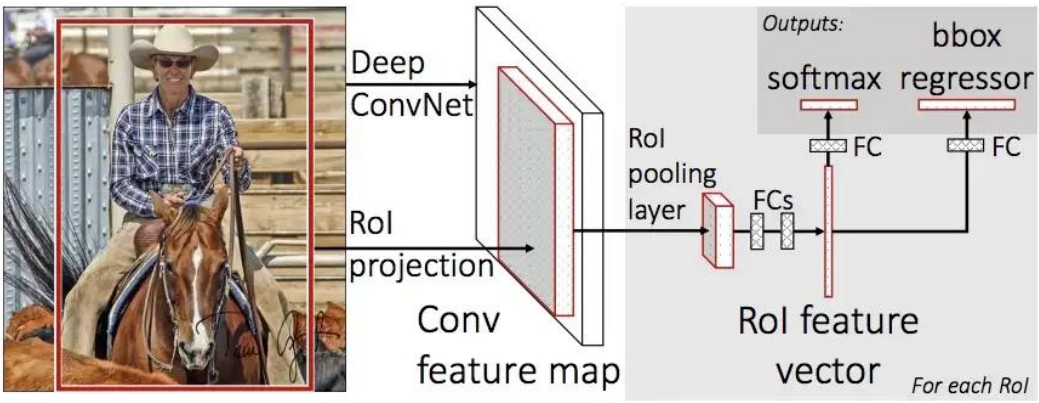


Ilustración 19. Fast R-CNN diagrama (Gandhi, R., 2018)

Esta red soluciona algunos de los problemas de las R-CNN como por ejemplo que necesitan una gran cantidad de tiempo para ser entrenadas ya que tiene que clasificar 2000 propuestas de regiones por imagen, o que no se pueda implementar en tiempo real pues tarda unos 50 segundos en predecir cada imagen.

El enfoque de esta nueva red es similar a la R-CNN pero en lugar de enviar las regiones propuestas a la CNN, enviamos la imagen entrada para generar el mapa de características convolucionales. De este mapa, identificamos las regiones propuestas y las envolvemos en cuadrados y, usando una capa de agrupación region of interest (RoI), las deformamos en un tamaño fijo para que puedan alimentarse en una capa completamente conectada. Desde el vector de características RoI, usamos una capa SoftMax para predecir la clase de la región propuesta, así como los límites del cuadro delimitador.

Faster region based convolutional neural networks (faster R-CNN)

Esta, elimina el algoritmo de búsqueda selectiva y permite que la propia red aprenda las propuestas de región, lo que la hace mucho más rápida.

Similar a Fast R-CNN, la imagen se proporciona como entrada a una red convolucional que proporciona un mapa de características convolucionales. En lugar de usar un algoritmo de búsqueda selectiva en el mapa de características para identificar las propuestas de región, se usa una red separada para predecir las propuestas de región. Las propuestas de región predichas luego se remodelan usando una capa de agrupación de RoI que luego se usa para clasificar la imagen dentro de la región propuesta y predecir los límites de los cuadros delimitadores.

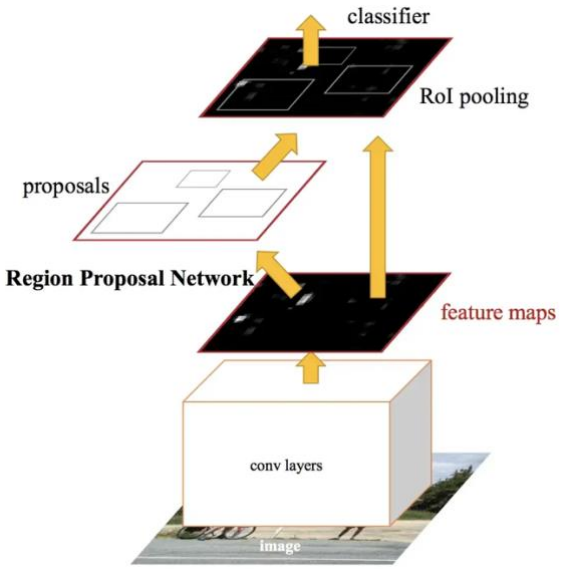


Ilustración 20. Faster R-CNN diagram (Gandhi, R., 2018)

YOLO (You Only Look Once)

A diferencia de los algoritmos anteriores, YOLO mira la imagen completa en vez de usar regiones para localizar el objeto dentro de la imagen. YOLO es un algoritmo de detección de objetos muy diferente a los basados en regiones, este, en una única red convolucional predice las fronteras de los cuadros donde están los objetos, así como su probabilidad.

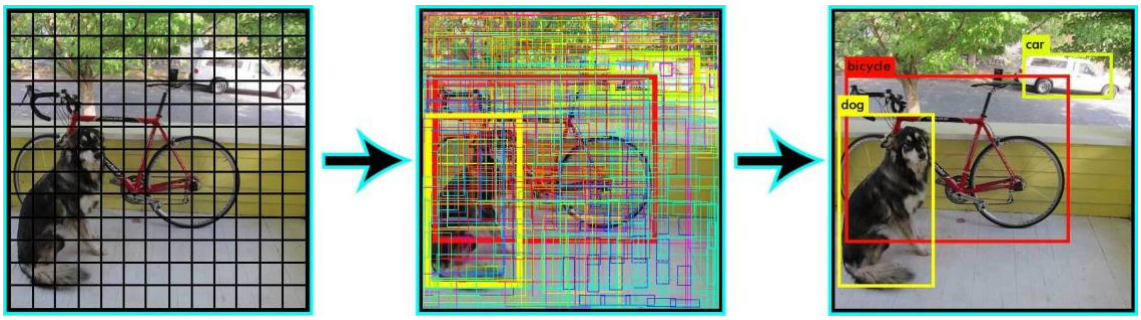


Ilustración 21. YOLO example. (Khandewal, R., 2019)

YOLO divide la imagen en una maya de SxS, para cada segmento de la cuadrícula tomamos m cuadros delimitadores. Para cada uno de los cuadros, la red genera la probabilidad de la clase, así como los límites de los cuadros. Solo nos quedamos con los cuadros que tengan una probabilidad por encima de un umbral definido por el usuario.

Este algoritmo es mucho más rápido (hasta 45 fotogramas por segundo) que otros algoritmos de detección de objetos.

Conclusión

Dado que lo que queremos en detectar objetos (mala hierba) en tiempo real, lo más interesante es usar el algoritmo YOLO para la detección de las mismas. La idea sería entrenar YOLO con un dataset propio para que pueda aprender a reconocer donde hay mala hierba.

YOLO

La motivación del autor es construir un modelo unificado de todas las fases en una red neuronal. Con la imagen de entrada que contiene (o no) los objetos, después de pasar a través de una sola red neuronal de múltiples redes convolucionales, el sistema produce vectores predictivos correspondientes a cada objeto que aparece en la imagen. En lugar de repetir el proceso de clasificación de diferentes regiones en la imagen, el sistema YOLO calcula todas las características de la imagen y hace predicciones para todos objetos al mismo tiempo. Esa es la idea de "You Only Look Once". (Redmon, et al., 2016)

YOLO explotó el campo de la detección de objetos en 2015 cuando salió su primera versión publicada por Joseph Redmon. Hasta ahora en combinación con las ideas más innovadoras surgidas por la visión artificial, YOLO se ha actualizado en 5 versiones y se ha considerado uno de los algoritmos de detección de objetos más destacados. La 5ª generación de YOLO, YOLOv5, es la última versión no desarrollada por el autor original de YOLO, sin embargo, el rendimiento es superior que YOLOv4 tanto en precisión como en velocidad.

Conceptos

La idea principal de YOLOv1 es aplicar una celda de cuadrícula con el tamaño de $S \times S$ (7 × 7 por defecto) en una imagen. Si el centro de un objeto cae en una celda de cuadrícula, esa celda de cuadrícula es responsable de detectar ese objeto (Ilustración 22). Por lo tanto, todas las demás células ignoran incluso esa apariencia de objeto revelada en múltiples celdas.

Para implementar la detección de objetos, cada celda de la cuadrícula predice B cuadros delimitadores con sus parámetros y puntuaciones de confianza para esos cuadros (Ilustración 23) (V Thatte, A., 2020). Estos puntajes de confianza reflejan la presencia o ausencia de un objeto en el cuadro delimitador. La puntuación de confianza se define como:

$$confidence\ score = p(Object) * IOU_{pred}^{truth}$$

siendo $p(Object)$ la probabilidad de que haya un objeto dentro de la celda y IOU_{pred}^{truth} es la intersección sobre la unión del cuadro de predicción y el cuadro de verdad fundamental. $p(Object)$ está en el rango 0 y 1, por lo que la puntuación de confianza es cercana a 0 si no existe ningún objeto en esa celda.

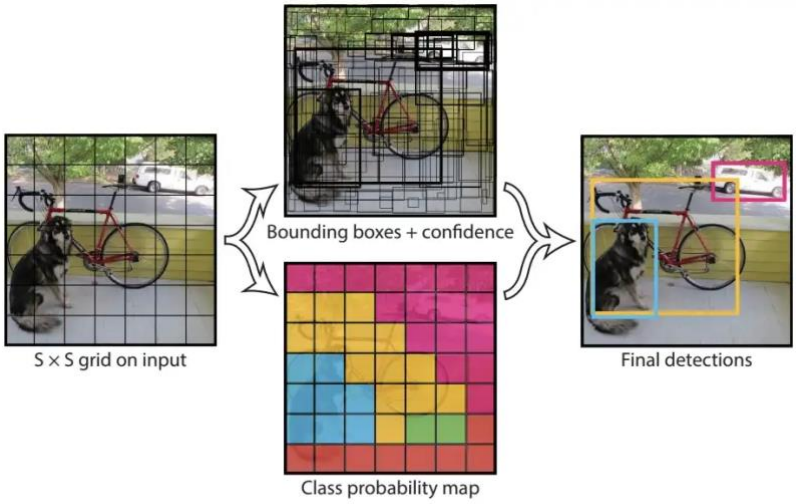


Ilustración 22. YOLO con maya de 7x7 celdas aplicadas a una imagen de entrada. (Redmon, et al,. 2016)

Además, cada cuadro delimitador consta de otros 4 parámetros (x, y, w, h) correspondientes a ($center\ coordinate$ (x, y), $width$, $height$ (Ilustración 23). Combinando con la puntuación de confianza, cada cuadro delimitador consta de 5 parámetros.

$$x = \frac{(220 - 149)}{149} = 0.48$$

$$y = \frac{(170 - 149)}{149} = 0.14$$

$$w = \frac{120}{448} = 0.27$$

$$h = \frac{160}{448} = 0.38$$

Ilustración 23 . Parámetros para un cuadro delimitador en una celda de cuadrícula de 3x3.

La probabilidad de un objeto predicho para cada clase en una celda de cuadrícula se denota por $p(Class_i|Object)$. Los valores de probabilidad para la clase C producirán una salida de C para cada celda de la cuadrícula. El cuadro delimitador B de la misma celda de cuadrícula comparte un conjunto común de predicciones sobre la clase del objeto, lo que significa que todos los cuadros delimitadores en la misma celda de cuadrícula tienen la misma clase. Como se muestra en la Ilustración 24, por ejemplo, que hay 2 cuadros de predicción en la celda central. Tienen diferentes parámetros (x, y, w, h, $confidence\ score$). Sin embargo, tienen las mismas 3 clases de predicción.

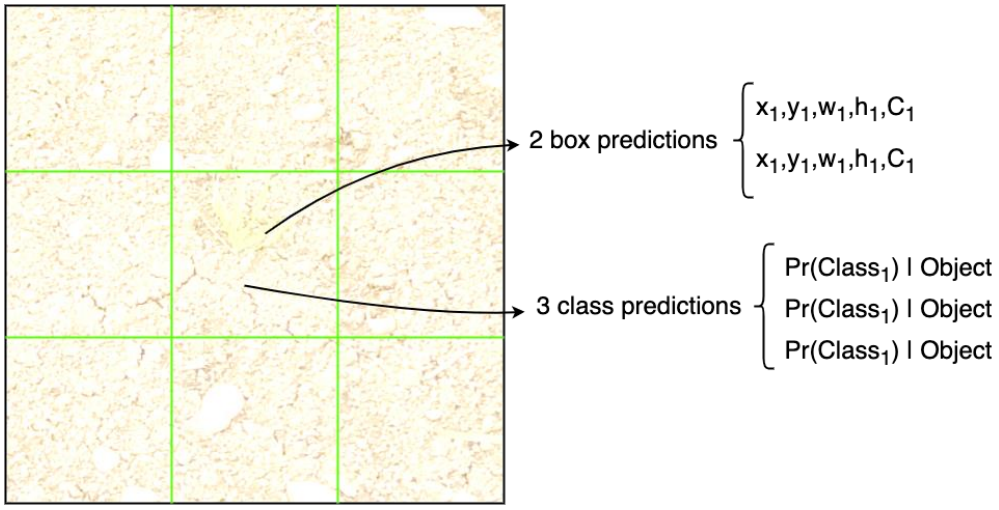


Ilustración 24. Los cuadros delimitadores de la misma celda de cuadrícula comparten el conjunto de clases de predicción. (Menegaz, M., 2018)

Por lo tanto, el modelo tiene $S \times S$ celdas de cuadrícula para una imagen. Cada celda predice B cuadros delimitadores que consta de 5 parámetros y comparte probabilidades de predicción de C clases. La salida total de YOLO de los parámetros del modelo serán $S \times S \times (5 * B + C)$ (Menegaz, M., 2018). Por ejemplo, al evaluar YOLO con el famoso conjunto de datos COCO (Common objects in context) que contiene 80 clases y establece que cada celda predice 2 cuadros delimitadores, los parámetros de salida totales son $7 \times 7 \times (5 * 2 + 80)$.

El propósito del algoritmo YOLO es detectar un objeto al predecir con precisión el cuadro delimitador que contiene ese objeto y localizar el objeto en función de las coordenadas del cuadro delimitador. Por lo tanto, los vectores del cuadro delimitador pronosticados corresponden al vector de salida $\hat{y}$ y el cuadro delimitador de verdad básica de los vectores correspondientes a la etiqueta del vector y. La etiqueta del vector y y el vector predicho $\hat{y}$ se pueden indicar como en la Ilustación 20 donde la celda morada no tiene ningún objeto, la puntuación de confianza de los cuadros delimitadores en la celda morada igual a 0, todos los parámetros restantes serán ignorados.

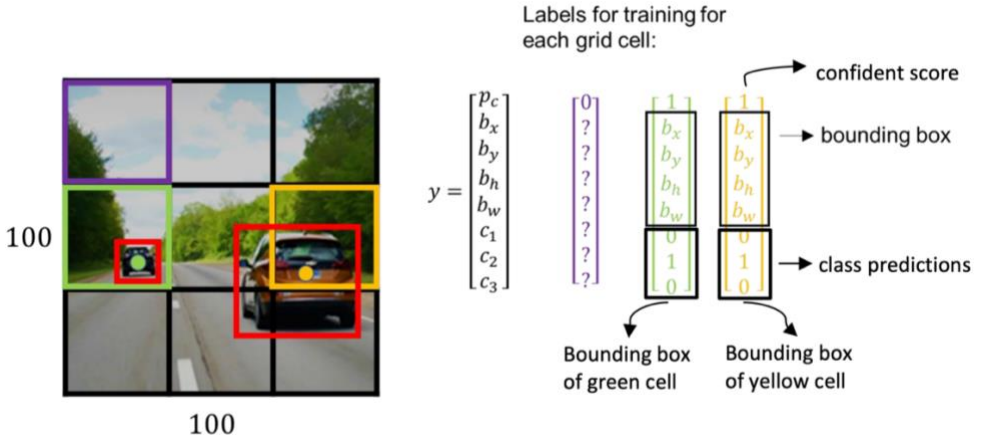


Ilustración 25. Especificación del vector de etiquetas y en un modelo YOLO que tiene celdas de cuadrícula de 3x3. (datahacker, 2018)

Finalmente, YOLO aplica supresión no máxima (NMS) para limpiar todos los cuadros delimitadores que no contienen ningún objeto o que contienen el mismo objeto que otros cuadros delimitadores. Eligiendo un valor umbral, NMS elimina todos los cuadros delimitadores superpuestos que tienen intersección sobre la unión (IOU) valor superior al valor de umbral. (ODSC Science, 2018)

Arquitectura YOLOv1

El modelo YOLO está diseñado para abarcar una arquitectura que procesa todas las características de la imagen (los autores lo llamaron arquitectura Darknet) y seguido de 2 capas completamente conectadas que realizan predicciones de las delimitaciones de los objetos (Ilustación 21). Este modelo fue evaluado en el conjunto de datos Pascal VOC, donde los autores usaron S = 7, B = 2 y C = 20. Esto explica por qué los mapas de características finales son 7 × 7, y el tamaño de salida fue (7 × 7 × (2 ∗ 5 + 20)).

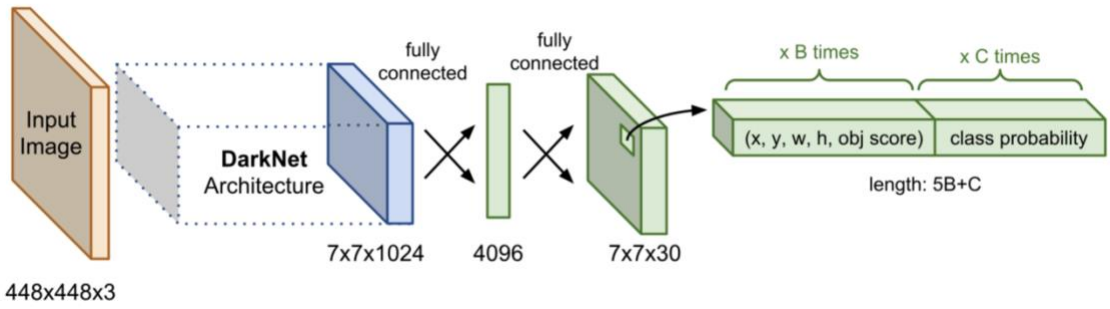


Ilustración 26. Arquitectura preliminar de YOLOv1.

Los autores han introducido el modelo fast-YOLO con 9 capas CNN en arquitectura Darknet para conjuntos de datos sin complicaciones y el modelo YOLO normal con 24 capas CNN en arquitectura Darknet que puede manejar conjuntos de datos más complejos que producen una mayor precisión (Ilustración 27 y 28). Las secuencias de 1x1 y las capas convolucionales 3x3 se inspiraron en el modelo GoogLeNet (Inception) que ayuda a reducir el espacio de características

de las capas anteriores (Menegaz, M., 2018). La capa final utiliza una función de activación lineal en lugar de la ReLU con fugas (ReLU con fugas) como todas las demás capas.

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

Ilustración 27. Arquitectura de Normal-YOLOv1 con 24 capas convolucionales y 2 capas completamente conectadas

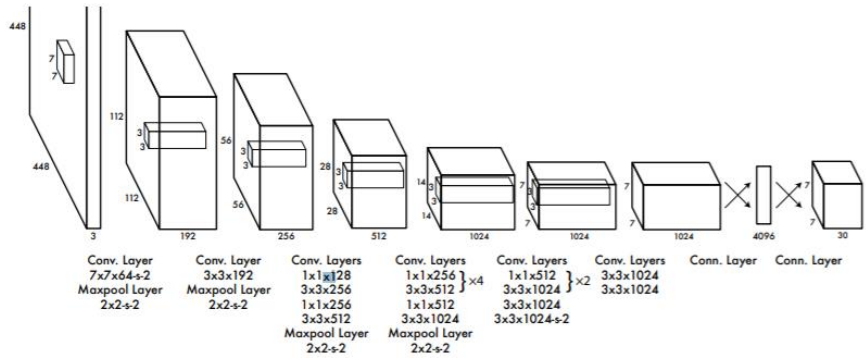


Ilustración 28.Arquitectura final de YOLOv1. (Redmon, et al, 2016)

Función de pérdida

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$
$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2$$
$$+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

La primera parte de la ecuación calcula la pérdida relacionada con la posición predicha del cuadro delimitador y posición del cuadro delimitador de la realidad basada en coordenadas (x_{center}, y_{center}). $\mathbb{I}_{ij}^{obj}$ se define como 1 si el objeto está dentro de j^{th} cuadro delimitador predicho en i^{th} celda, y 0 para lo contrario. El cuadro delimitador predicho será "responsable" de predecir un objeto en función de qué predicción la IOU más alta con la verdadera.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

La segunda parte de la ecuación calcula el error en la predicción del cuadro delimitador en anchura y altura similar a la primera parte de la ecuación. Sin embargo, la magnitud del error en cuadros grandes afecta a la ecuación menos que los cuadros pequeños. Debido a que la altura y la anchura están normalizadas entre 0 y 1, sus raíces cuadradas incrementan las diferencias para los valores pequeños más que para los valores más grandes, ese es el motivo por el cual se usan raíces cuadradas en lugar de la anchura y altura directamente.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

La puntuación de pérdida de confianza se calcula en ambos casos tanto si el objeto está presente en el cuadro delimitador o no. La función de pérdida solo penaliza el error de confianza del objeto si ese predictor es responsable de la caja de verdad de tierra. $\mathbb{I}_{ij}^{obj}$ es igual a 1 cuando hay un objeto en la celda, y 0 de lo contrario. $\mathbb{I}_{ij}^{noobj}$ es lo contrario.

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2$$

La última parte de la función de pérdida es similar a la función de pérdida de clasificación normal que calcula la pérdida de probabilidad de clase, excepto por el término $\mathbb{1}_{ij}^{obj}$. Este, se utiliza porque YOLO no penaliza errores de clasificación incluso cuando no hay objetos presentes en la celda.

$$\sum_{i=0}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

Evolución de YOLO

Existen más de 5 versiones de YOLO publicadas hasta ahora. Cada versión ha sido actualizada e integrada con las ideas más destacadas en el momento en el campo de la visión artificial. Además, algunas ideas han sido eliminadas debido a la incapacidad de alcanzar el rendimiento y la precisión requerida. Eso hace que YOLO sea uno de los algoritmos de predicción de objetos más potentes a día de hoy. Veamos las mejoras más remarcables en las versiones posteriores a la original.

YOLOv2

Añade normalización por lotes (Batch Normalization):

La normalización por lotes es uno de los métodos de normalización más populares en modelos de Deep Learning, este permite entrenamientos más rápidos y estables de redes neuronales profundas, estabilizando la distribución de las capas de entrada durante el entrenamiento. La meta es normalizar las características a una distribución de media 0 y desviación típica 1.

Se aplica la normalización por lotes en todas las capas de convolución de YOLOv2. Esta técnica no solo reduce el tiempo de entrenamiento, sino que también aumenta la generalización de la red. En YOLOv2, con la normalización en lote, aumentó mAP (precisión promedio media) en aproximadamente un 2 % (Redmon, et al., 2016). La red tampoco necesita usar abandonos adicionales para evitar el sobre entrenamiento.

Clasificación de alta resolución

En la primera versión de YOLO, las primeras 20 capas convolucionales eran usadas para entrenar el extractor de características con una imagen input de 224 x 224. Después las restantes 4 capas convolucionales y las 2 capas completamente conectadas incrementan la resolución.

Mientras que en YOLOv2, después de completar la fase de extracción de características, el modelo continúa con el entrenamiento de extracción de características por 10 épocas más con una imagen de tamaño 448 x 448. Esto ayuda al modelo a "adaptarse" a altas resoluciones de 448 x 448 en vez de aumentar el tamaño de la imagen de repente. Esta mejora de aumento de resolución hizo que aumentara en casi un 4% mAP.

Cuadros anclados

La idea de YOLOv1 es usar una celda de cuadrícula para que sea la responsable de detectar un objeto cuyo centro está dentro de esa celda. Por lo que, cuando dos o más objetos que tienen el centro dentro de la misma celda, la detección puede ser errónea. Para resolver este problema, el autor intentó permitir a una celda predecir más de un objeto. En YOLOv2, el autor introdujo una arquitectura de "cuadros anclados" para predecir los cuadros delimitadores en vez de usar capas completamente conectadas como en YOLOv1 (Redmon, et al., 2016). Los cuadros anclados son una lista de cuadros predefinidos que mejor encajan con los objetos deseados. Los cuadros delimitadores no solo fueron predichos en base a los cuadros de verdad, sino que también por los *k* cuadros anclados predefinidos.

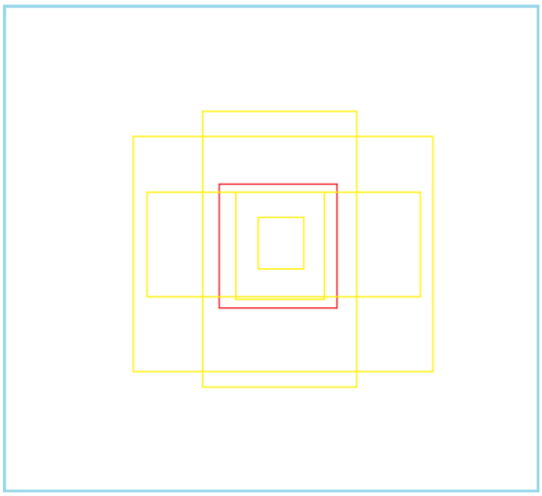


Ilustración 29. Para cada celda (roja), el modelo predice 5 marcos delimitadores basado en 5 marcos anclados (amarillo)

YOLOv3

Red más grande con ResNet

YOLOv2 usaba 30 capas convolucionales profundas para la arquitectura DarkNet. Para redes neuronales profundas, más capas significan más precisión. Sin embargo, la imagen de entrada se reducía cuando se reenviaba a capas más profundas, lo que provocaba la pérdida de características de granularidad fina. Es por eso que YOLOv2 a menudo tenía problemas con la detección de objetos pequeños. ResNet trajo la idea de omitir conexiones para ayudar a que las activaciones se propaguen a través de capas más profundas sin que desaparezca el gradiente. (Ilustración 30) (He,k . et al., 2015).

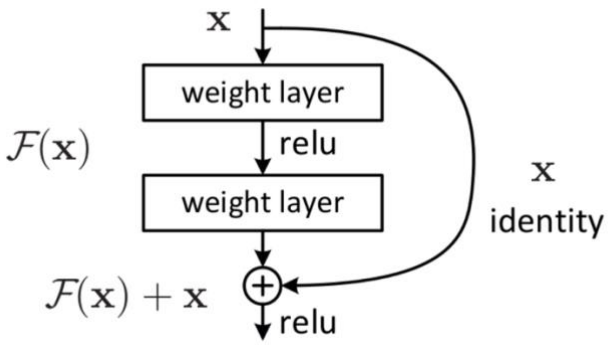


Ilustración 30. ResNet con saltar conexión arquitectura.

YOLOv3 vino con una mejor arquitectura donde el extractor de características era un híbrido entre YOLOv2, DarkNet-53 (53 capas convolucionales), y Residual networks (ResNet) (Redmon, J& Farhadi, A., 2018).

El modelo usó la arquitectura Darknet-53 que originalmente tiene 53 capas para el entrenamiento del extractor de características. Después de eso, se apilaron 53 capas más para la cabeza para el entrenamiento (head) de la detección del objeto, haciendo qué YOLOv3 tenga un total de 106 capas.

Detección multi-escala

En las 2 versiones anteriores de YOLO, después de entrenar el extractor de características con arquitectura Darknet, la entrada se reenviaba a algunas capas más y finalmente hacía las predicciones en las últimas capas del detector de objetos. Sin embargo, YOLOv3 agregó las capas de predicción a un lado de la red en lugar de apilarlas en las últimas capas como antes (Ilustración 31). La característica más notable de YOLOv3 es que realiza detecciones en 3 escalas diferentes (Redmon, J., & Farhadi, A., 2018). Las características de los últimos 3 bloques residuales se utilizan para 3 detectores de escala diferentes.

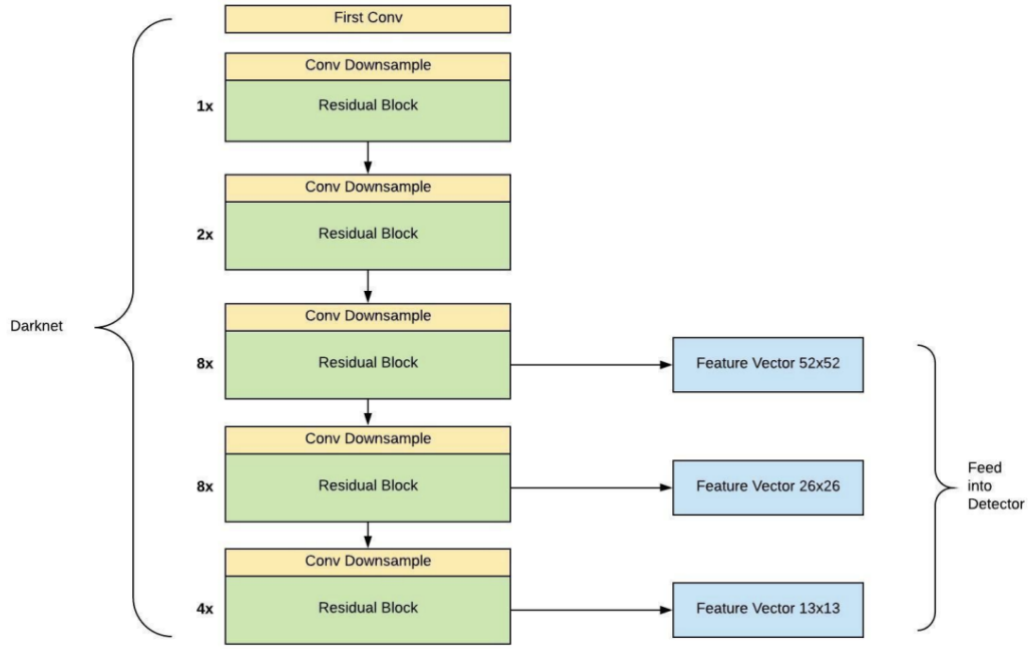


Ilustración 31. Detector multi-escala se añade al lado de la red para hacer 3 detecciones a 3 escalas diferentes. (Yanjia, Ei, 2019)

Más específicamente, YOLOv3 hace predicciones a 3 escalas en las capas 82,94 y 106.

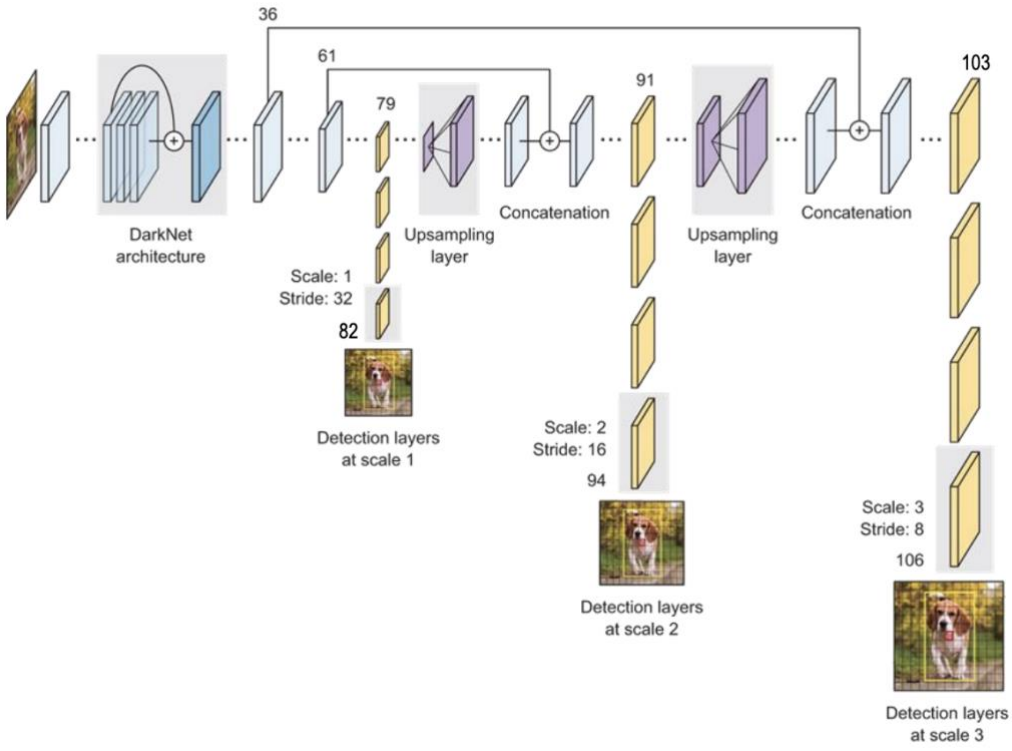


Ilustración 32. YOLOv3 arquitectura de la red combinando ambos el extractor de características y el detector de objetos (Ayoosh, K., 2018)

YOLOv4

El algoritmo original YOLO fue escrito por Joseph Redmon. Después de 5 años de investigación y desarrollo de la 3ª generación de YOLO (YOLOv3), Joseph Redmon anunció su retirada en el campo de la visión artificial se desvinculaba del desarrollo del algoritmo YOLO debido a que sus investigaciones serán mal usadas en aplicaciones militares.

En abril de 2020, Alexey Bochkovsky, un investigador ruso e ingeniero que desarrolló el framework de Darknet y las 3 versiones previas de la arquitectura YOLO en C basadas en los conceptos teóricos de Joseph Redmon, colaboró junto con Yao y Hon-Yuan para publicar YOLOv4 (Bochkovskiy, A., 2020)

Arquitectura en la detección de objetos

Junto con el desarrollo de YOLO, varios algoritmos de identificación de objetos que utilizan diversas metodologías han logrado resultados sobresalientes. Desde entonces, han surgido dos ideas de arquitectura para detección de objetos: detector de una etapa (one-stage detector) y detector de dos etapas (two-stage detector).

El extractor de características (Backbone) comprime las características de la imagen de entrada y luego las envía al detector de objetos (que contiene el cuello de detección y el cabezal de detección), como se muestra a continuación. El cuello de detección (Neck) funciona como una agregación de características, mezclando y combinando las características creadas en el Backbone para prepararse para el proceso de detección en el cabezal de detección (Head).

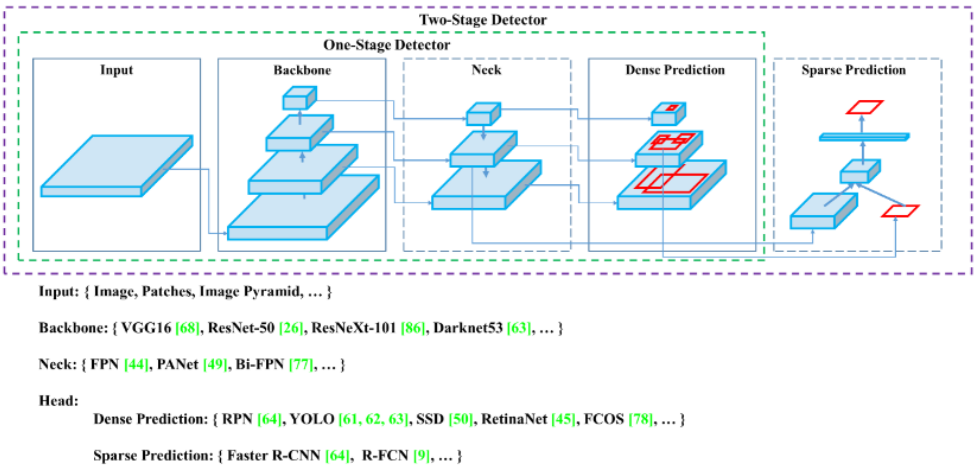


Ilustración 33. Detector de objetos. (Solawetz, J., 2020)

Backbone CSPDarknet53

Los autores investigaron tres alternativas para el Backbone del modelo YOLOv4 (extractor de características): CSPResNext53, CSPDarknet53 y EfficientNet-B3, la red convolucional más sofisticada en ese momento. Se

consideró que la red neuronal CSP Darknet53 era el mejor modelo óptimo según la justificación teórica y varias pruebas, como se muestra a continuación.

Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

Ilustración 34. Tabla comparando 3 redes backbone. (Huang, G., Liu, Z., & Maaten, L. v., 2018)

Las arquitecturas CSPResNext50 y CSPDarknet53 (CSP significa Cross Stage Partial) se desarrollan a partir del diseño DenseNet, que toma la entrada anterior y la concatena con la entrada actual antes de pasar a la capa densa. DenseNet se creó para vincular capas en una red neuronal extremadamente profunda con el fin de resolver desafíos de gradientes que se desvanecen (como ResNet).

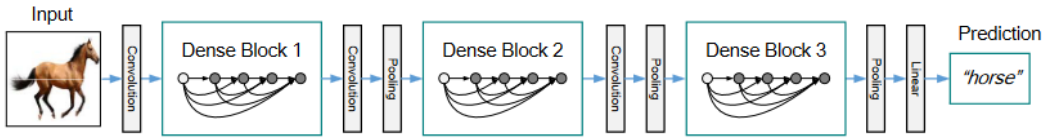


Ilustración 35. Arquitectura DenseNet con 3 bloques densos. Las capas entre 2 bloques son las capas de transición. (Huang, G., Liu, Z., & Maaten, L. v., 2018)

Para ser más específicos, cada etapa de DenseNet se compone de un bloque denso y una capa de transición, como se muestra arriba, y cada bloque denso se compone de *k* capas densas. Después de pasar por el bloque denso, la entrada se enrutará a la capa de transición, que cambiará el tamaño (disminuir o aumentar la muestra) mediante convolución y agrupación. La salida de la capa densa i^{th} se concatenará con su propia entrada para generar la entrada para la siguiente capa $(i + 1)^{th}$. Por ejemplo, en la primera capa densa, la entrada x_0 ha creado la salida x_1 después de avanzar a través de las capas convolucionales. La salida x_1 luego se concatena con su propia entrada x_0, y el resultado de esta concatenación se convierte en la entrada de la segunda capa densa (ver más abajo).

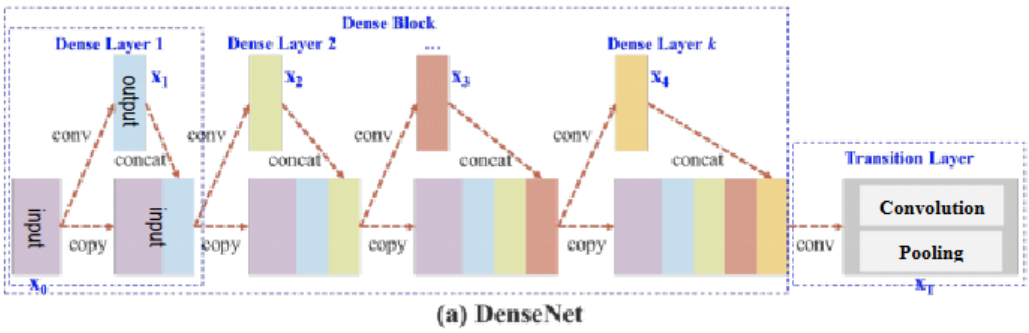


Ilustración 36. Proceso del procesado de un input en un bloque denso. (Wang, C. et al., 2019)

El CSP (Cross Stage Partial) se basa en la misma premisa que DenseNet descrita anteriormente, excepto que en lugar de utilizar el mapa de características de entrada de tamaño completo en la capa base, la entrada se dividirá en dos mitades. Un fragmento se enviará a través del bloque denso de forma normal, mientras que otro se enrutará directamente al siguiente paso sin ser procesado (como se ve a continuación).

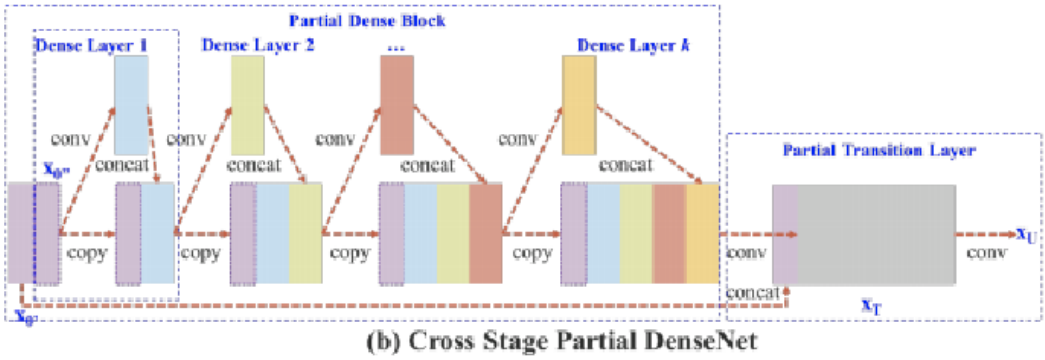


Ilustración 37. Proceso del procesado de un input en un bloque parcialmente denso. (Wang, C. et al., 2019)

Cuando estos conceptos se combinaron con el diseño Darknet-53 en YOLOv3, los bloques residuales se reemplazaron con bloques densos. CSP conserva las características a través de la propagación, estimula la red para que reutilice características, reduce la cantidad de parámetros de red y ayuda a conservar las características detalladas para un reenvío más eficiente a capas más profundas. Dado que aumentar el número de capas convolucionales densamente vinculadas puede resultar en una caída en la velocidad de detección, solo el bloque convolucional final en la red troncal Darknet-53 que puede extraer características semánticas más ricas se mejora para ser un bloque.

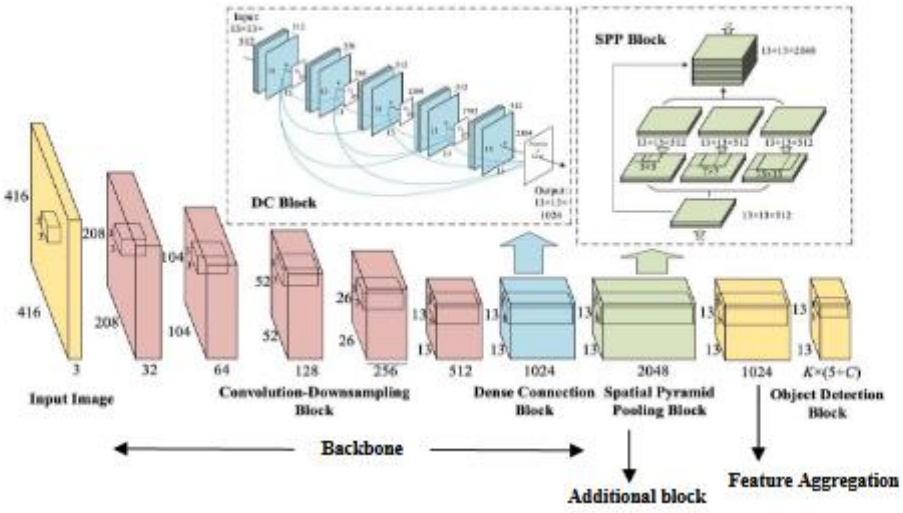


Ilustración 38. CSPDarknet53 con bloque adicional. (Deval, S., 2020)

Neck – Bloque adicional – Bloque SSP

Los mapas de características de salida de la red troncal CSPDarknet53 se envían a un bloque adicional (bloque de agrupación piramidal espacial) para ampliar el campo receptivo y separar las características más relevantes antes de enviarlas a la arquitectura de agregación de características en el cuello (Ilustración 38).

Muchos modelos basados en CNN (red neuronal convolucional) tienen capas totalmente conectadas que toman como entrada solo imágenes de ciertas dimensiones. SPP se creó con el objetivo de producir una salida de tamaño fijo independientemente del tamaño de la entrada. SPP también ayuda en la extracción de características esenciales al agrupar versiones de múltiples escalas de sí mismo. Dado que las capas completamente conectadas se eliminaron de YOLOv2, el algoritmo de YOLO ha evolucionado a un modelo basado en FCN (fully convolution network) que acepta imágenes de diferentes dimensiones como entrada.

Además, YOLO debe pronosticar y localizar las ubicaciones de los cuadros delimitadores en función de la celda de cuadrícula $S \times S$ que se muestra en la imagen. Como resultado, convertir mapas de características bidimensionales en un vector unidimensional de tamaño fijo no siempre es una buena idea. Como resultado, el bloque SPP se actualizó para mantener la dimensión espacial de salida. El nuevo bloque SPP esta situado cerca de la red troncal.

Se emplea una convolución 1X1 entre la red Backbone y el bloque SPP para reducir la cantidad de mapas de características de entrada entregados al bloque SPP. Después de eso, los mapas de características de entrada se duplican y agrupan en múltiples escalas utilizando el mismo enfoque que el bloque SPP tradicional, excepto que se emplea el relleno para conservar los mapas de características de salida en un tamaño constante, de modo que tres mapas de características siguen siendo del tamaño de $sizefmap \times sizefmap \times 512$..

Neck – Agregación de características – PANet

Después de pasar por el Backbone, las características de la imagen de entrada se convierten en características semánticas (o características aprendidas). En otras palabras, a medida que la imagen de entrada avanza a través de las capas de bajo nivel, la complejidad de las características semánticas aumenta, pero la resolución espacial de los mapas de características disminuye debido a la reducción de la resolución. Como resultado, se pierde información espacial y características detalladas. Para el cuello de YOLOv3, Joseph Redmon usó la arquitectura Feature Pyramid Network (FPN) para mantener estas características de grano fino. El diseño de FPN usó un enfoque de arriba hacia abajo para transmitir información semántica (desde la capa de alto nivel) y luego concatenarla en características detalladas (desde la capa de bajo nivel de la columna vertebral) para predecir objetos diminutos en el detector a gran escala. Path Aggregation Network (PAN) es una versión más avanzada de FPN.

Debido a que el flujo en la arquitectura FPN es de arriba hacia abajo, solo el detector a gran escala de las capas de bajo nivel en FPN puede recibir información semántica de las capas de alto nivel y las características detalladas de las capas de bajo nivel en el Backbone lateral al mismo tiempo. Actualmente, el detector a pequeña escala de capas de alto nivel en FPN detecta objetos utilizando únicamente información semántica. El hecho de concatenar características semánticas con características de grano fino en capas de alto nivel fue sugerido para aumentar el rendimiento de los detectores de pequeña y mediana escala.

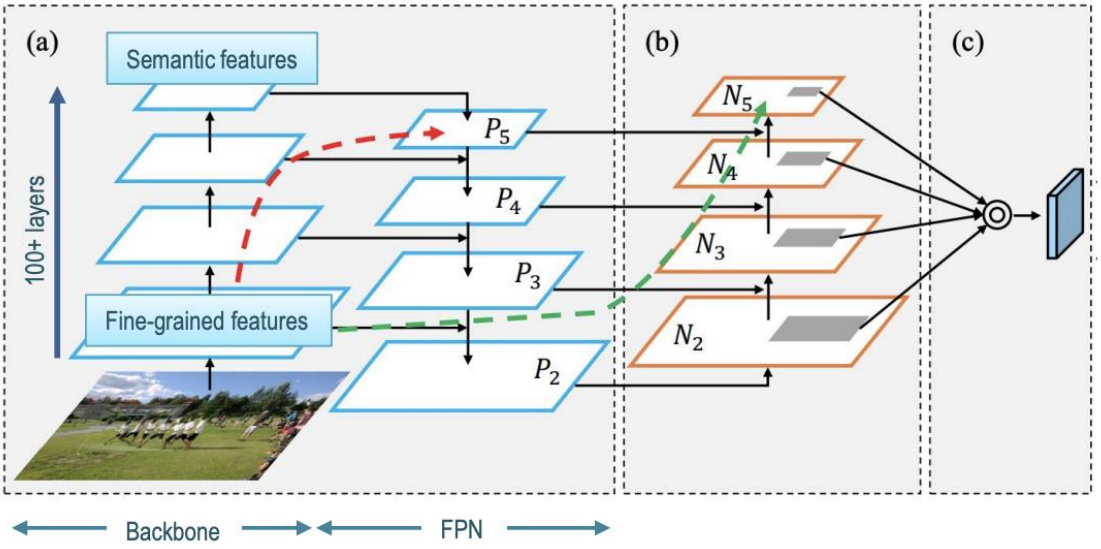


Ilustración 39. Arquitectura PANet incluyendo (a) FPN backbone, (b) bottom-up path augmentation, (c) agregación de características. (Liu, S., 2018)

Sin embargo, el Backbone de las redes neuronales profundas de hoy en dia comprende una gran cantidad de capas (pueden ser más de 100 capas). Como resultado, en FPN, las entidades de granularidad fina deben atravesar un largo viaje desde las capas de bajo nivel hasta las de alto nivel. Los desarrolladores de la arquitectura PAN ofrecieron un enfoque de aumento de abajo hacia arriba además del de arriba hacia abajo utilizado en FPN. Como resultado, se creó un "atajo" para vincular las características detalladas de las capas de nivel inferior a las capas de nivel superior. Este "atajo" tiene menos de diez capas, lo que permite un flujo de información más fácil.

Como se muestra en la Ilustración 39, la arquitectura de Backbone y FPN se presenta de la misma manera. Pero en realidad, el Backbone contiene muchas capas, las 4 capas dibujadas representan las capas que fueron concatenados a detectores de múltiples escalas en la arquitectura FPN, no representan para todo el Backbone. Es la razón por la cual el “atajo” del camino verde (Ilustración 12) pasa por menos de 10 capas, aunque más largo que el camino rojo en la misma imagen.

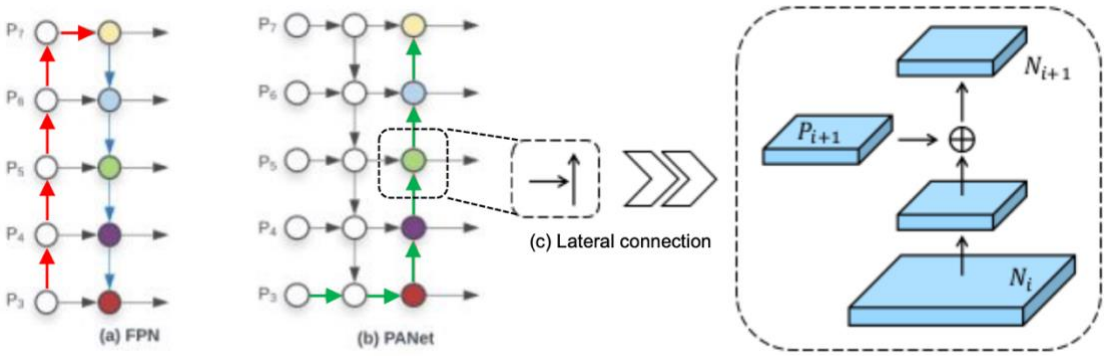


Ilustración 40. (a) FPN architecture. (b) PAN architecture. (c) Connection in bottom-up augmentation. (Solawetz, J., 2020)

La ruta de aumento de abajo hacia arriba se puede observar como una réplica de la ruta de arriba hacia abajo FPN en la que cada etapa contiene capas que producen mapas de características con los mismos tamaños espaciales. Estos mapas de características se conectan a la arquitectura lateral mediante la operación de suma por elementos (Ilustración 41a), mientras que en la arquitectura PAN modificada para YOLOv4, los autores la reemplazaron con la operación de concatenación (Ilustración 41b). Esto ayuda a que el flujo de información no se pierda ni las características de FPN ni las características de la ruta de aumento de abajo hacia arriba.

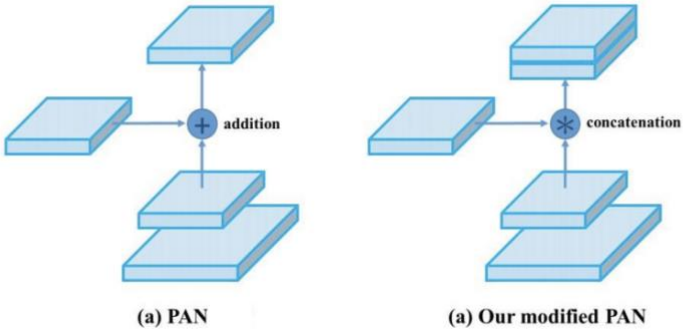


Ilustración 41. (a) Original PAN. (b) Modificada PAN para YOLOv4. (Bochkovskiy, A., 2020)

YOLOv5

Un mes después del lanzamiento de YOLOv4, el investigador Glenn y su equipo publicaron una nueva versión de la familia YOLO, llamada YOLOv5 (Jocher, G., 2020). Glenn Jocher es investigador y director ejecutivo de Ultralystics LLC. Los modelos YOLO se desarrollaron en un framework personalizado de Darknet que está escrito principalmente en C por Alexey Bochkovsky. Ultralystic es la empresa que convierte versiones anteriores de YOLO en uno de los marcos más famosos en el campo del Deep Learning, PyTorch, que está escrito en el lenguaje Python.

Descripción general

Glenn Jocher es también el inventor de la técnica de aumento de datos "Moisac", reconocido por Alexey Bochkovsky en el artículo de YOLOv4 (Bochkovskiy, A., 2020). Sin embargo, su modelo YOLOv5 causó mucha controversia en la comunidad de visión artificial debido a su nombre y mejoras.

A pesar de que fue publicado un mes después de YOLOv4, el inicio de la investigación de YOLOv4 y YOLOv5 fue bastante cercana (marzo – abril de 2020). Para evitar la colisión, Glenn decidió nombrar su versión de YOLO, YOLOv5. Así, básicamente, ambos investigadores aplicaron las innovaciones más avanzadas en el campo de visión artificial en ese momento. Eso hace que la arquitectura de YOLOv4 y YOLOv5 sea muy similar y hace que muchas personas no estén satisfechas con el nombre YOLOv5 (quinta generación de YOLO) cuando no contiene múltiples mejoras destacadas respecto a la versión anterior YOLOv4. Además, Glenn no publicó ningún artículo para YOLOv5, lo que provocó más sospechas sobre YOLOv5.

Sin embargo, YOLOv5 poseía ventajas en su ingeniería. YOLOv5 está escrito en Python en lugar de C como versiones anteriores. Eso hace que la instalación e integración en dispositivos IoT sea más fácil. Además, la comunidad PyTorch es más grande que la de Darknet, lo que significa que PyTorch recibirá más contribuciones y, por tanto, tiene más potencial de crecimiento en el futuro. Debido a que está escrito en 2 idiomas diferentes y 2 frameworks diferentes, comparar el rendimiento entre YOLOv4 y YOLOv5 es difícil pero después de un tiempo, YOLOv5 ha demostrado un mayor rendimiento que YOLOv4 bajo ciertas circunstancias y en parte ganó confianza en la comunidad de visión artificial.

Diferencias notables – Cuadros de anclaje adaptativos

Como se mencionó anteriormente, la arquitectura YOLOv5 ha integrado las últimas innovaciones similares a la arquitectura YOLOv4, por lo tanto, no hay muchas diferencias en teoría. El autor no publicó un documento detallado, solo lanzó un repositorio en Github y actualizaciones de mejoras en él. Al diseccionar la estructura de su código en el archivo *yaml*, el modelo YOLOv5 se puede resumir como lo siguiente (Jocher, G., 2020):

- The backbone consta de una red CSP.
- Neck: SPP block, PANet
- Head: Head: YOLOv3 head using GIoU-loss

El punto notable mencionado por el autor de YOLOv5 es una diferencia de ingeniería. Joseph Redmon introdujo la estructura del cuadro de anclaje en YOLOv2 y un procedimiento para seleccionar el tamaño y la forma de los cuadros de que se asemejan mucho a las cajas delimitadoras de la realidad en el conjunto de entrenamiento. Al usar el algoritmo de agrupación en clústeres k-means con diferentes valores de k, los autores eligieron los 5 que mejor se ajustaban cuadros

de anclaje para el conjunto de datos COCO (que contiene 80 clases) y utilícelos como predeterminados. Eso reduce el tiempo de entrenamiento y aumenta la precisión de la red.

Sin embargo, al aplicar estos 5 cuadros de anclaje a un conjunto de datos único (que contiene una clase a la que no pertenece a 80 clases en el conjunto de datos COCO), estos cuadros de anclaje no pueden adaptarse rápidamente a la realidad de los cuadros delimitadores de este conjunto de datos. Por ejemplo, un conjunto de datos de jirafas prefiere los cuadros de anclaje con la forma delgada y más alta que una caja cuadrada. Para abordar este problema, generalmente se ejecuta el algoritmo de agrupación en clústeres k-means en el conjunto de datos único para obtener el mejor ajuste de los cuadros de anclaje para los datos. Luego, estos parámetros serán configurados manualmente en la arquitectura YOLO.

Glenn Jocher propuso integrar el proceso de selección del cuadro de anclaje en YOLOv5. Como resultado, la red no tiene que considerar ninguno de los conjuntos de datos que se utilizarán como entrada, automáticamente "aprenderá" los mejores cuadros de anclaje para ese conjunto de datos y los utilizará durante el entrenamiento. (Solawetz, J., 2020)

Limitaciones

Aunque YOLO parece ser el mejor algoritmo para usar si tiene que resolver un problema de detección de objetos, tiene varias limitaciones.

- YOLO se esfuerza por detectar y segmentar objetos pequeños en imágenes que aparecen en grupos, ya que cada cuadrícula está restringida para detectar un solo objeto. Los objetos pequeños que naturalmente vienen en grupos, como una fila de hormigas, son por lo tanto difíciles de detectar y localizar para YOLO.
- También se caracteriza por una menor precisión en comparación con algoritmos de detección de objetos mucho más lentos como Fast RCNN.

Entrenamiento con dataset propio

Para entrenar YOLO con un dataset propio, debemos seguir el siguiente formato (YOLOv5-GitHub, 2023):

Para cada imagen, tenemos que crear un fichero TXT con las etiquetas asociadas a los objetos de esa imagen. Este fichero ha de cumplir la siguiente especificación:

- Una fila por objeto
- Cada fila tiene el formato <clase> <centro_x> <centro_y> <anchura> <altura>
- Las coordenadas deben estar normalizadas (0-1)
- Las clases empiezan desde el número 0

Por ejemplo, dada esta imagen:

Ilustración 42. Imagen con medidas relativas sobre los objetos. (YOLOv5-GitHub, 2023)

El TXT asociado a esta imagen contendría la siguiente información

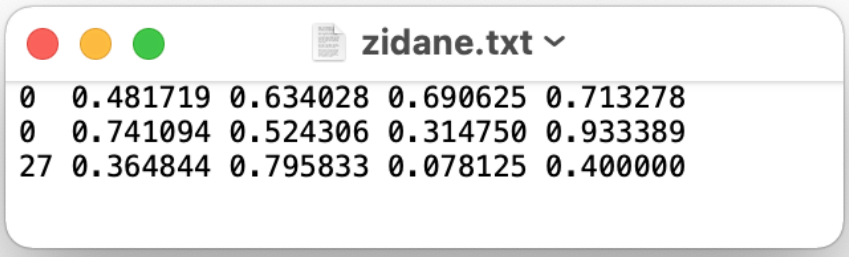


Ilustración 43. Fichero de etiquetas asociado a la Ilustración 42. (YOLOv5-GitHub, 2023)

Siendo el 0 la clase persona y la clase 27 la clase corbata.

Generación de imágenes sintéticas

Las imágenes sintéticas son imágenes creadas artificialmente que pretenden ser lo más fieles a la realidad posible, se utilizan cuando no hay suficientes datos. De esta forma, a partir de imágenes reales, se simulan nuevas imágenes diferentes a las reales para que la red aprenda también de estas nuevas situaciones.

En nuestro caso se han conseguido imágenes de cultivo de girasol en estado temprano. Las imágenes son del siguiente estilo:

Ilustración 44. Fotografía de suelo con cultivo y mala hierba

Donde, como se puede apreciar, se diferencia el cultivo (hojas gruesas) de la mala hierba (hojas finas y alargadas)

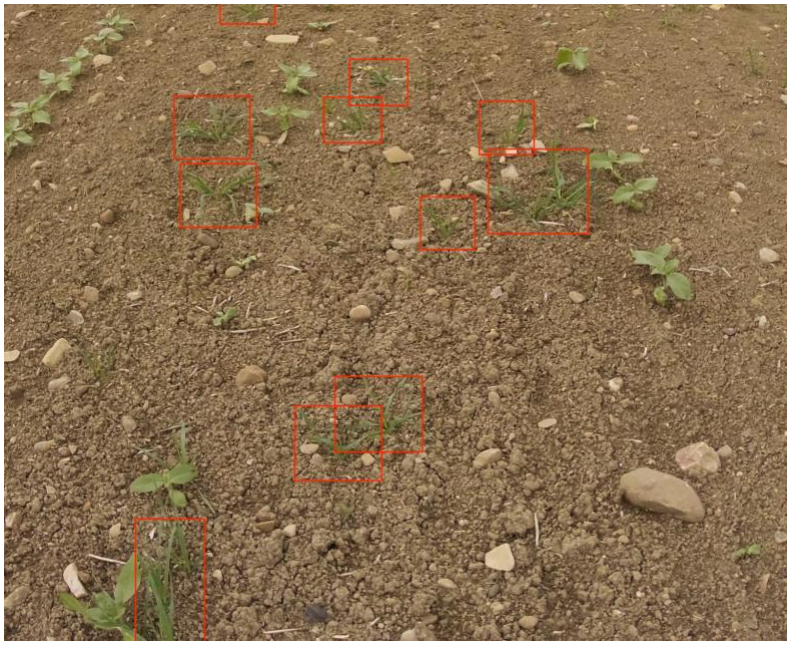


Ilustración 45. Fotografía de suelo con cultivo y mala hierba recuadrada en rojo

Dado que no se tienen suficientes imágenes reales (unas 200), generaremos nuestras propias imágenes.

Algoritmo

Previamente a explicar el algoritmo de generación de imágenes sintéticas, necesitamos:

Obtener máscaras Alpha de malas hierbas, algunas de ellas:

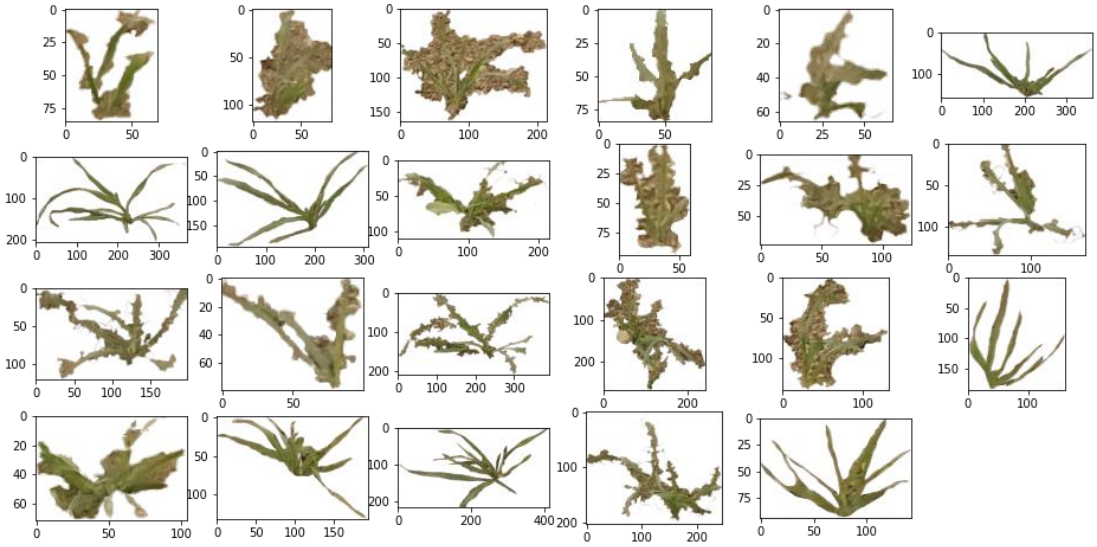


Ilustración 46. Máscaras de malas hierbas

De la misma forma, es necesario obtener los "backgrounds" de nuestras imágenes sintéticas, estos "backgrounds" son imágenes del cultivo SIN malas hierbas.

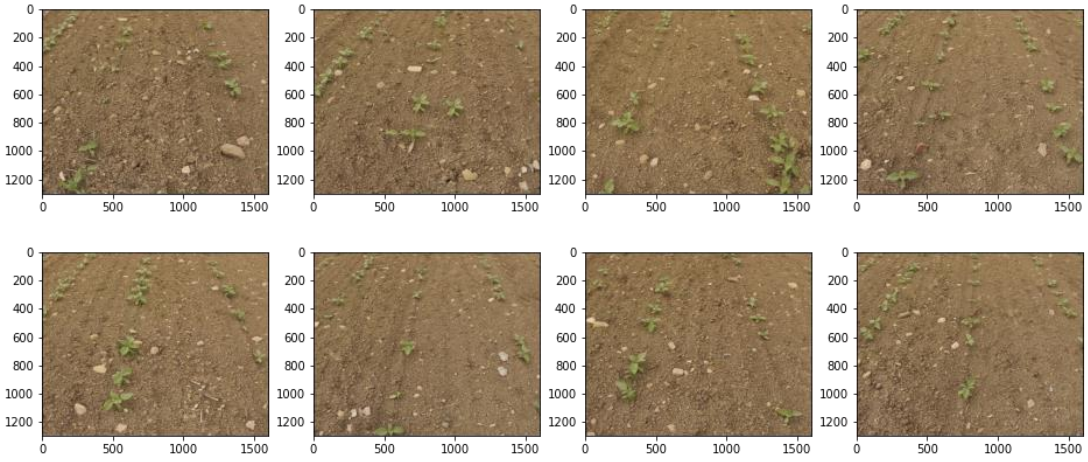


Ilustración 47. Fotos de suelo con cultivo sin mala hierba

Una vez tenemos la base de nuestras imágenes sintéticas, correremos el siguente algoritmo. Este, se reduce a posicionar *n* malas hierbas modificadas en tamaño y rotación en estos backgrounds.

Más concretamente:

Leemos de manera aleatoria una imagen "background" de las disponibles, de manera aleatoria también leemos *n* imágenes alpha de malas hierbas que modificamos en escala y rotación de manera aleatoria, seguidamente colocamos

en posiciones aleatorias de la imagen “background” estás *n* imágenes de malas hierbas modificadas.

En pseudocódigo:

obtener imagen background aleatoria de las disponibles

n loop, n = random int (1-8):

obtener imagen alpha de weed aleatoria

rescalar la imagen weed leida a una escala aleatoria (x0.6 , x1.4)

rotar la imagen weed un angulo aleatorio (-90 , 90)º

obtener posición x,y aleatoria donde posicionaremos la mala hierba

pegar la imagen alpha weed modificada en la posición x,y de la imagen background

escribir en fichero posición normalizada donde se coloca la imagen de mala hierba en el background, tambien el tamaño de la imagen de la mala hierba

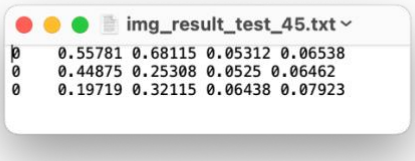
Repetimos este algoritmo tantas veces como imágenes queremos generar.

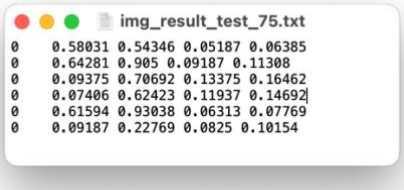
Algunas de las imágenes generadas son:

Se han recuadrado en rojo, donde se han posicionado las malas hierbas, solamente es algo visual, las imágenes generadas no contienen este recuadro rojo.

Conforme se van generando las imágenes, para cada una se genera también el txt asociado con las etiquetas de los objetos que contiene.

Por ejemplo:





Como vemos, todas las clases son la 0 ya que los únicos objetos que estamos detectando son mala hierba, tampoco estamos distinguiendo entre tipos de malas hierbas dado que para el caso de estudio solo se tiene un tipo de mala hierba. Cada linea de estos ficheros nos indica la posición del centro y el tamaño normalizado del objeto (mala hierba) en la imagen.

Teniendo en cuenta todo esto, podemos calcular el número de imágenes diferentes que podríamos generar artificialmente sin que haya dos imágenes iguales, asumiendo que dos imágenes son diferentes cuando tienen tanto diferente imagen de "background" como las *n* imágenes de malas hierbas.

$$número\ total\ de\ imágenes = \ q * \sum_{i=1}^{i \leq n} \frac{m!}{i!\,(m-i)!}$$

Siendo

- q: el número de imágenes background que tenemos disponibles.
- m: el número de malas hierbas disponibles entre las que elegir.
- n: el número máximo de malas hierbas que pondremos en la imagen

Se limita a que en cada imagen generada no se tengan más de 8 malas hierbas, es decir, $\mathrm{n} = 8$. Se disponen de 8 imágenes de "Background" ($\mathrm{q}=8$) y 25 imágenes alpha de malas hierbas ($\mathrm{m} = 25$)

$$8 * \sum_{i=1}^{i \leq 25} \frac{25!}{i!\,(25-i)!} = 8 * \ 33554431 = \ 268435448$$

Con 8 backgrounds y 25 malas hierbas, la combinatoria nos dice que se podrían llegar a generar hasta 268435448 imágenes diferentes. Para nuestro caso, nos basta con generar unas 800 imágenes. Utilizaremos 600 imágenes sintéticas para el entrenamiento, 200 imágenes sintéticas para la validación y 200 imágenes reales para test.

Entrenamiento de YOLO

Entorno

Google Colab (Google Collaboratory) es un entorno de desarrollo integrado (IDE) gratuito de Google para apoyar la investigación y el aprendizaje sobre la inteligencia artificial (IA). Colab proporciona un entorno para ejecución de código como Jupyter Notebook, de uso gratuito de procesamiento gráfico (GPU) así como de procesamiento de tensores (TPU). Google Colab tiene bibliotecas preinstaladas que son muy populares en la investigación de Deep Learning, como PyTorch, TensorFlow, Keras y OpenCV. Debido a que los algoritmos de DL/ML requieren que el sistema tenga alta velocidad y potencia de procesamiento (generalmente basada en GPU), las computadoras normales no están equipadas con GPU. Por lo tanto, Colab suministra GPU (Tesla T4) y TPU (TPUv2) en la nube, una de las GPU de mayor rendimiento en este momento, para brindar asistencia a los investigadores de IA.

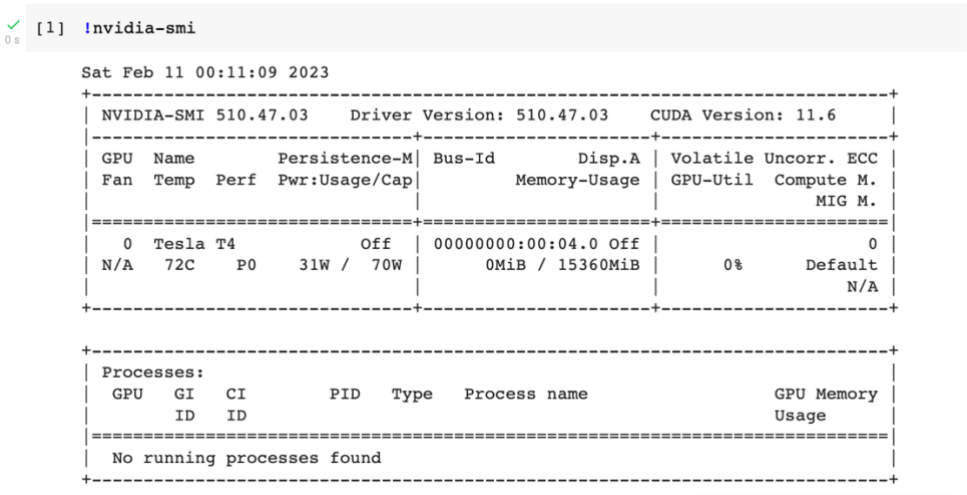


Ilustración 48.Colab provides Tesla T4 GPU on cloud for training deep learning models.

Al igual que con las versiones anteriores, la arquitectura YOLOv5 se construyó con base teórica y se lanzó a través de un repositorio en GitHub. Como se mencionó anteriormente, Ultralystic desarrolló YOLOv5 en el marco PyTorch, uno de los frameworks más populares en la comunidad de IA. Sin embargo, esta es solo una arquitectura preliminar, los investigadores pueden configurar la arquitectura para obtener los mejores resultados según sus problemas, como agregar capas, eliminar bloques, integrar métodos adicionales de procesamiento de imágenes, cambiar los métodos de optimización o las funciones de activación, etc.

```
[3] #clone YOLOv5 and
    !git clone https://github.com/ultralytics/yolov5  # clone repo
    %cd yolov5
    %pip install -qr requirements.txt # install dependencies
    %pip install -q roboflow

    import torch
    import os
    from IPython.display import Image, clear_output  # to display images

    print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_prope
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 15114, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 15114 (delta 32), reused 27 (delta 10), pack-reused 15045
Receiving objects: 100% (15114/15114), 14.13 MiB | 30.39 MiB/s, done.
Resolving deltas: 100% (10349/10349), done.
/content/yolov5/yolov5
Setup complete. Using torch 1.13.1+cu116 (Tesla T4)
```

Ilustración 49. Cloning and installing the YOLOv5 repository

Preparando el dataset para el entrenamiento

Dado que Colab es un servicio de Google, permite vincularse a una cuenta personal de Google Drive y obtener datos de Drive para usarlos en el entorno de desarrollo y posteriormente guardar los resultados.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Ilustración 50. Mounting to the Google Drive.

Como hemos comentado anteriormente, para entrenar YOLO, necesitamos:

- Imágenes
- Fichero de objetos (etiquetas) por cada imagen

Estos datos ya los hemos obtenidos mediante el algoritmo de generación de imágenes sintéticas y subido en un archivo comprimido a nuestra cuenta personal en Google Dive. A continuación, descomprimimos estos datos en el entorno de ejecución de Colab, esto hace los datos más accesibles.

```
!unzip -q /content/drive/MyDrive/agro/data.zip -d /content/
```

Ilustración 51. Unzip the dataset into Colab environment

El fichero comprimido contiene:

- Images/train: Carpeta con 600 imágenes para el entrenamiento.

- labels/train: Carpeta con 600 ficheros *txt* con las etiquetas de cada imagen de entrenamiento.

- Images/val: Carpeta con 200 imágenes para la validación.

- labels/val: Carpeta con 200 ficheros *txt* con las etiquetas de cada imagen de validación.

- Images/test: Carpeta con 200 imágenes para test

A continucación creamos un fichero de configuración de nuestro conjunto de datos (custom.yaml), para indicar posteriormente a YOLOv5 de donte tiene que obtener estos datos.

```
train: /content/data/images/train
val: /content/data/images/val
test:  /content/data/images/test
# Classes
names:
  0: weed
```

Como vemos, las rutas del fichero *yaml* comienzan por `/content/` porque es donde se han descomprimido los datos de Drive. Se indica además el nombre (weed) de la única clase que existe en el fichero de etiquetas (0).

Entrenamiento

Con la línea de comando que se muestra en la Ilustración 52, el modelo se entrenará mediante el archivo de compilación train.py junto con sus argumentos configurables.

```
!python train.py --img 640 --batch 4 --epochs 30
--data /content/yolov5/data/custom.yaml --weights yolov5x.pt --cache
```

Ilustración 52. Implementación del proceso de entrenamiento

Los input del comando de entrenamiento son los siguientes:

- img: Define el tamaño de las imágenes, Las imágenes originales tienen un tamaño de 1600 x 1300, comprimiendolas a un tamaño menor hace que el proceso de entrenamiento sea más rapido. Despues de varios experimentos, muchos investigadores están de acuerdo en que valores cercanos a 416 x 416 son ideales para usarse como input sin perder mucho detalle, para este caso se comprimirán a 640.

- batch: Determina el tamaño del lote. El reenvío de miles de imágenes a la red neuronal al mismo tiempo hace que la cantidad de pesos que el modelo

aprende de una vez (un batch) aumente mucho. Por lo tanto, el conjunto de datos generalmente se divide en múltiples lotes de n imágenes y se entrena lote por lote. Los resultados de cada lote se guardan en RAM y se agregan después de que se complete el entrenamiento para todos los lotes. Debido a que los pesos aprendidos de los lotes se almacenan en RAM, cuanto mayor sea el número de lotes, mayor será el consumo de memoria.

- epochs: Definir el número de épocas de entrenamiento. Una época es responsable de aprender todas las imágenes de entrada, en otras palabras, entrenar todas las entradas. Dado que el conjunto de datos se divide en varios lotes, una época será responsable de entrenar todos los lotes. El número de épocas representa el número de veces que el modelo entrena todas las entradas y actualiza los pesos para acercarse a las etiquetas de verdad. A menudo, este número es elegido en base a la experiencia y la intuición.

- data: el path del archivo *data.yaml* el cual contiene el resumen del dataset.

- cache: Cachear imágenes para un enrtrenamiento más rápido.

- weights: Especifica la ruta de los pesos. Se puede usar pesos preentrenados para ahorrar tiempo de entrenamiento. Si se deja en blanco, el modelo inicializará automáticamente los pesos de manera aleatoria.

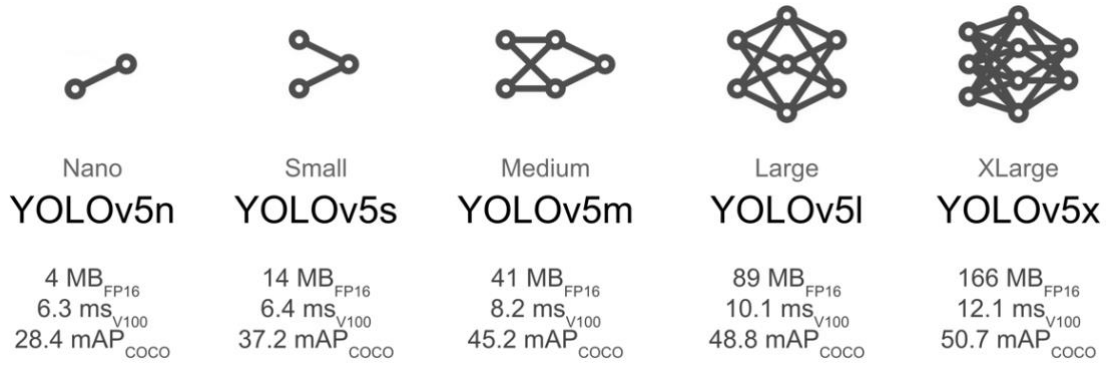


Ilustración 53. YOLOv5 modelos preentrenados

```
Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
22/29      4.95G    0.02121    0.02071          0         19        640: 100% 150/150 [00:45<00:00,  3.27it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.10it/s]
             all        200        885      0.997      0.969      0.988      0.795

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
23/29      4.95G    0.02037    0.02063          0         28        640: 100% 150/150 [00:45<00:00,  3.28it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.08it/s]
             all        200        885      0.995      0.967      0.991      0.814

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
24/29      4.95G    0.01983    0.02034          0         55        640: 100% 150/150 [00:45<00:00,  3.26it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.10it/s]
             all        200        885      0.992      0.979      0.993      0.832

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
25/29      4.95G    0.01918    0.01998          0         13        640: 100% 150/150 [00:45<00:00,  3.27it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.06it/s]
             all        200        885      0.991      0.976      0.992      0.826

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
26/29      4.95G    0.01839    0.01932          0          8        640: 100% 150/150 [00:45<00:00,  3.28it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.06it/s]
             all        200        885      0.994      0.974      0.993      0.843

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
27/29      4.95G    0.01811    0.01861          0         35        640: 100% 150/150 [00:46<00:00,  3.26it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.09it/s]
             all        200        885      0.985      0.982      0.993      0.842

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
28/29      4.95G    0.01778    0.01863          0         21        640: 100% 150/150 [00:45<00:00,  3.28it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.09it/s]
             all        200        885      0.994      0.977      0.994      0.859

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
29/29      4.95G       0.017    0.01842          0         22        640: 100% 150/150 [00:45<00:00,  3.26it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:04<00:00,  5.09it/s]
             all        200        885      0.984      0.986      0.994      0.861

30 epochs completed in 0.464 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 173.1MB
Optimizer stripped from runs/train/exp/weights/best.pt, 173.1MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 322 layers, 86173414 parameters, 0 gradients, 203.8 GFLOPs
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 25/25 [00:05<00:00,  4.30it/s]
             all        200        885      0.984      0.986      0.994      0.861
Results saved to runs/train/exp
```

Ilustración 54.Yolov5 proceso de entrenamiento

Métricas del entrenamiento

Antes de obtener las métricas del entrenamiento, conviene repasar ciertos conceptos:

- Intersección sobre Unión (IoU): La intersección sobre la unión es una métrica popular para medir la precisión de la localización y calcular los errores de localización en los modelos de detección de objetos.

  Para calcular el IoU con las predicciones y la realidad básica, primero tomamos el área de intersección entre los cuadros delimitadores de una predicción particular y los cuadros delimitadores reales de la misma área. Después de esto, calculamos el área total cubierta por los dos cuadros delimitadores, también conocidos como la *Unión*.

  La intersección dividida por la unión nos da la relación entre la superposición y el área total, lo que proporciona una buena estimación de qué tan cerca está el cuadro delimitador de la predicción original.

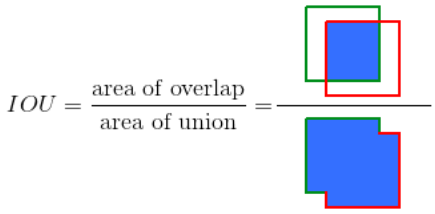


Ilustración 55. Intersección sobre unión

- Precisión media (AP): La precisión promedio se calcula como el área bajo una curva de precisión frente al recall para un conjunto de predicciones.

  El recall se calcula como la proporción de las predicciones totales realizadas por el modelo en una clase con un total de etiquetas existentes para la clase.

  Por otro lado, la Precisión se refiere a la relación de verdaderos positivos con respecto al total de predicciones realizadas por el modelo.

Definidas matemáticamente:

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

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

$$F1\ Score = 2 * \frac{precision * recall}{precision + recall}$$

$$TP = True\ positive$$

$$TN = True\ negative$$

$$FP = False\ positive$$

$$FN = False\ negative$$

El área bajo la curva de precisión frente al recall nos da la precisión promedio por clase para el modelo. El promedio de este valor, tomado en todas las clases, se denomina Precisión promedia media (mAP).

Una vez entrenado el modelo, podemos visualizar las estadísticas de las métricas que se han guardado en el fichero de logs. Para ello nos basta con utilizar los siguientes comandos:

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

Ilustración 56. Uso de TensorBoard para cargar el proceso entero de entrenamiento guardado en la carpeta runs

Lo que nos lanzará una ventana interactiva en la que veremos las siguientes métricas:

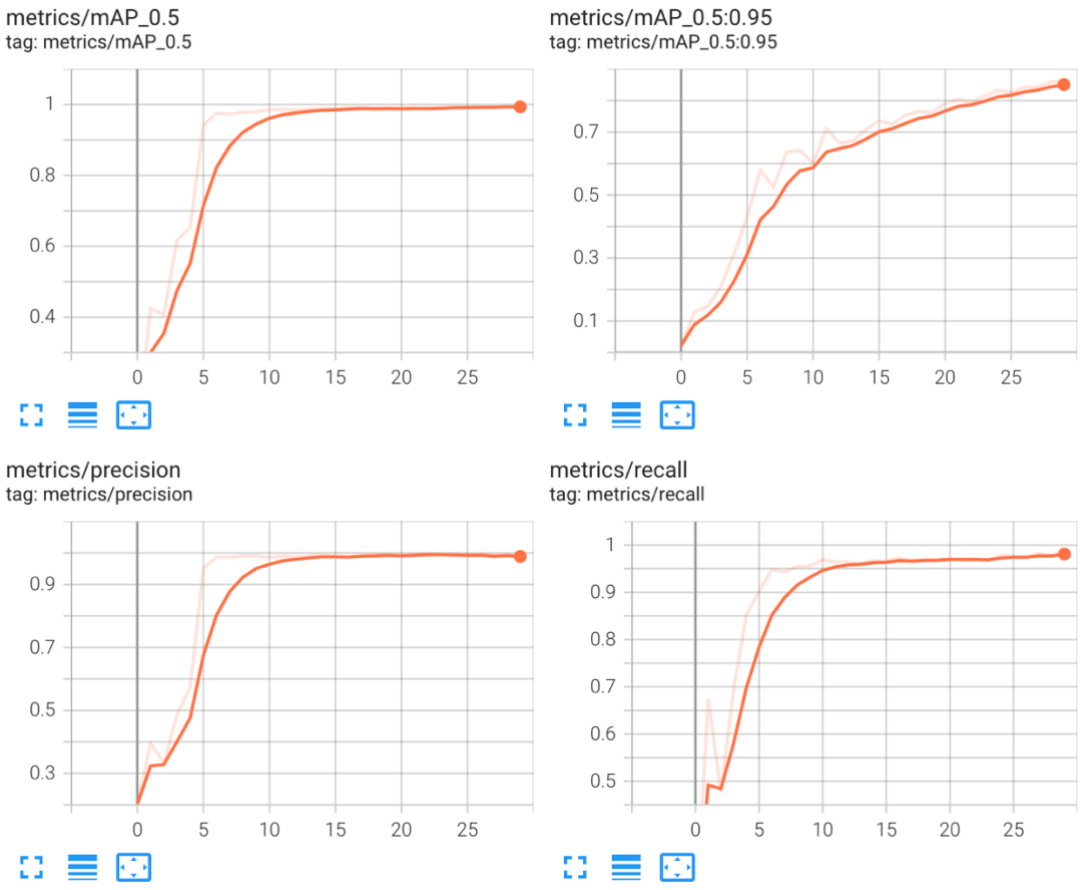


Ilustración 57. Evaluación de las métricas de las 30 épocas visualizadas gráficamente

mAP_0.5 corresponde al AP promedio para IoU de 0.5

mAP_0.5:0.95 corresponde al AP promedio para IoU de 0.5 a 0.95 con un tamaño de paso de 0.05. Es decir, es el promedio de todas las precisiones medias desde el AP_0.5 hasta AP_0.95 con incrementos de 0.05:

$$\frac{\sum_{n=0.5;n+=0.05}^{n\leq 0.95} AP\ con\ IoU = n}{10}$$

Como se muestra en la Ilustración 54 y la Ilustración 57, con un conjunto de datos que contiene 600 imágenes, el modelo tarda unos 50 segundos en completar una época, y solo con 30 épocas, la precisión del modelo (mAP_0.5) es de alrededor del 99%. Esto demuestra que, con la mera arquitectura original de YOLOv5, el modelo no solo es rápido, sino que la precisión también es alta sin ningún método de optimización integrado. Además, el modelo guarda 2 resultados de los pesos entrenados como archivo *pt*.

```
runs/train/exp/weights/best.pt
runs/train/exp/weights/last.pt
```

Como se muestra, el archivo *last.pt* es el peso en la última época y el archivo best.pt es el peso en la última época para obtener la máxima precisión. El tamaño de ambos archivos son de entorno a 100 MB, por lo que es muy fácil de integrarlo en los sistemas de IA (aplicación web, móvil, IoT…etc) manteniendo una precisión cercana al 99%.

Inferencia con pesos entrenados

Utilizamos los pesos entrenados para identificar malas hierbas en nuevas imágenes. Si hay presencia de mala hierba, se dibujará un recuadro sobre el objeto, así como se indicará la clase y la probabilidad del mismo.

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 640
--conf 0.1 --source /content/data/images/test/
```

Donde *runs/train/exp/weights/best.pt* es la ruta del modelo que acabamos de entrenar.

Y, */content/data/images/test/* es la ruta que contiene las imágenes que queremos inferir.

Fijamos el umbral de confianza a 0.1 para que con poca confianza lo clasifique como mala hierba. Mejor clasificar y que este mal, a no clasificarlo cuando si. De esta forma no aplicaremos herbicida cuando estemos muy seguros de que lo visto no es mala hierba.

```
image 46/200 /content/test/DJI_0590.JPG: 544x640 13 weeds, 43.9ms
image 47/200 /content/test/DJI_0600.JPG: 544x640 13 weeds, 43.8ms
image 48/200 /content/test/DJI_0610.JPG: 544x640 10 weeds, 42.8ms
image 49/200 /content/test/DJI_0620.JPG: 544x640 5 weeds, 42.6ms
image 50/200 /content/test/DJI_0630.JPG: 544x640 6 weeds, 42.8ms
image 51/200 /content/test/DJI_0640.JPG: 544x640 11 weeds, 42.8ms
image 52/200 /content/test/DJI_0650.JPG: 544x640 8 weeds, 43.2ms
image 53/200 /content/test/DJI_0660.JPG: 544x640 11 weeds, 43.2ms
image 54/200 /content/test/DJI_0670.JPG: 544x640 10 weeds, 42.8ms
image 55/200 /content/test/DJI_0680.JPG: 544x640 8 weeds, 42.6ms
image 56/200 /content/test/DJI_0690.JPG: 544x640 3 weeds, 42.9ms
image 57/200 /content/test/DJI_0770.JPG: 544x640 13 weeds, 42.7ms
image 58/200 /content/test/DJI_0780.JPG: 544x640 17 weeds, 43.0ms
```

Ilustración 58. Infiriendo con pesos entrenados

Como se puede ver, la predicción en nuevas imágenes es muy rápida, en torno a 40 ms por imagen, lo que nos permite procesar imágenes a unos 30 fps, en tiempo real.

Una vez completada la detección, los recuadros delimitadores que cubren los objetos (malas hierbas) se dibujarán en la imagen. Estos resultados se guardarán en */content/yolov5/runs/detect/exp*.

```
#display inference on ALL test images

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'):
    display(Image(filename=imageName))
    print("\n")
```

Ilustración 59. Visualizar las inferencias

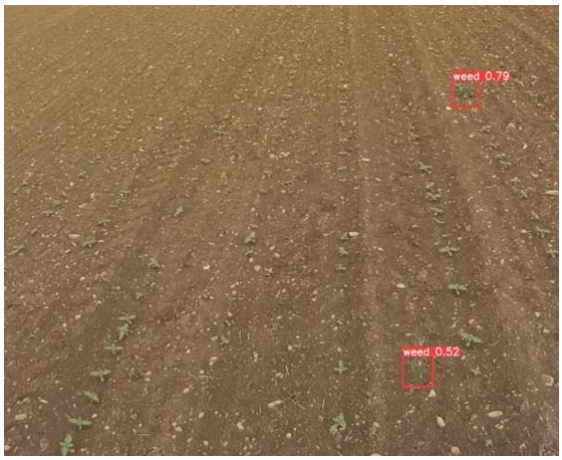
Algunas de las imágenes inferidas son:


weed 0.81
weed 0.77
weed 0.91
weed 0.50
weed 0.91
weed 0.60


weed 0.91
weed 0.75
weed 0.70
weed 0.40
weed 0.94


weed 0.19
weed 0.82
weed 0.60
weed 0.89
weed 0.62
weed 0.22
weed 0.48
weed 0.19
weed 0.68
weed 0.18
weed 0.64
weed 0.17
weed 0.90
weed 0.82
weed 0.54
weed 0.11
weed 0.79
weed 0.86


weed 0.86
weed 0.76
weed 0.68
weed 0.60
weed 0.50
weed 0.87
weed 0.62
weed 0.41
weed 0.80
weed 0.76
weed 0.87
weed 0.11
weed 0.80
weed 0.92
weed 0.84


weed 0.79
weed 0.52


weed 0.19
weed 0.82
weed 0.60
weed 0.62
weed 0.89
weed 0.22
weed 0.48
weed 0.19
weed 0.68
weed 0.18
weed 0.64
weed 0.17
weed 0.90
weed 0.82
weed 0.54
weed 0.11
weed 0.86

Ilustración 60.Visualización inferencias con el modelo entrenado

Podemos descargar el modelo para poder usarlo en nuestras aplicaciones DL/ML

```
#export your model's weights for future use
from google.colab import files
files.download('./runs/train/exp/weights/best.pt')
```

Ilustración 61. Descarga del modelo en formato PyTorch

Este comando descargará el modelo con el que acabamos de inferir las imágenes en formato PyTorch, para posteriormente, integrarlo en la aplicación DL/ML.

Conclusión y líneas futuras

Como se ha podido observar, YOLOv5 responde muy bien como solución al problema planteado. No es un modelo perfecto ya que no lo existe actualmente, pero funciona más que sobresaliente. Hoy en día, todavía hay mucha controversia sobre el nombre y las mejoras de YOLOv5 en la comunidad de visión artificial. Sin embargo, a parte del nombre, el rendimiento de YOLOv5 es superior al de YOLOv4 tanto en velocidad como en precisión. Gracias a que YOLOv5 fue desarrollado en el framework PyTorch, al ser fácil de utilizar y tener una comunidad muy grande, tiene ya dos versiones posteriores (YOLOv6 y YOLOv7).

Como se ha visto, el trabajo se ha realizado específicamente para el cultivo de girasol, pues son los datos que se disponían para entrenar el modelo DL/ML. Como líneas futuras, se pretende generalizar el modelo para que sea capaz de actuar sobre cualquier tipo de cultivo, es decir, que sea capaz de detectar cualquier tipo de mala hierba. Para ello es necesario reentrenar el modelo con imágenes de todo tipo de mala hierbas en los diferentes tipos de cultivos. Este nuevo modelo, además de predecir la posición de la mala hierba en la imagen, también predeciría el tipo de mala hierba, pudiendo así "aplicar un tipo de herbicida u otro en función del tipo de mala hierba" en tiempo real. Además de esto, se pretende realizar el mismo experimento con las versiones más recientes de YOLO para comprobar si aumenta el rendimiento del mismo.

Otro punto a mejorar es la generación de imágenes sintéticas. En el experimento realizado, YOLO se ha entrenado con imágenes sintéticas en las cuales cada imagen contiene un número aleatorio de malas hierbas entre 1 y 8. Para generar imágenes más fieles a la realidad, lo que se va a hacer es calcular sobre las imágenes reales el número medio de malas hierbas que contiene una imagen, es decir, se va a medir la densidad de malas hierbas por imagen. A partir de esto, mediante una función de probabilidad, se generarán imágenes sintéticas que tengan una densidad de mala hierba parecida a la real.

Sobre lo que no abarca este TFM, se realizará un estudio sobre en qué dispositivo debe ejecutarse el algoritmo, pues queremos que sea de bajo coste y con suficiente rendimiento como para procesar imágenes en tiempo real, así como los dispositivos input/output que necesita el sistema (Cámara para tomar las imágenes, y Válvula reguladora). Pudiendo tener una unidad del sistema (Dispositivo, Cámara y Válvula reguladora) un coste inferior a 100 euros. Cada unidad se colocaría sobre cada una de las boquillas que dispersan el pesticida como se aprecia en la ilustración 7. De esta manera, los agricultores comprarían tantas unidades del sistema como boquillas tenga su tractor pulverizador. Probablemente el sistema completo tendría un coste de alrededor de 1500€ dependiendo de la maquinaria del agricultor, pudiendo ser amortizados simplemente en la primera aplicación de pesticida con el sistema, y lo más importante, sin tener que reemplazar la maquinaria actual. Este sistema es muchísimo más barato a la alternativa de John Deree (See and spray) así como mucho más preciso a ecoSniper, pues solo se basa en ver color verde para pulverizar. Además como ya se ha mencionado anteriormente, cabría la

posibilidad (mediante un sistema un poco más complejo) de poder aplicar un tipo u otro de herbicida en función del tipo de mala hierba detectada.

Como futura mejora, además, se propone definir un benchmark que mida la capacidad computacional del sistema y en base a eso se defina la velocidad del tractor. Pudiendo haber así modelos del sistema más económicos que otros, siendo el de bajo coste un dispositivo con menor potencia computacional y por ello, definiendo una velocidad menor para el tractor. De lo contrario el dispositivo más caro, tendría más potencia computacional y por lo tanto podría procesar imágenes a mayor velocidad.

Referencias

Valdés. H (April 6, 2017). Agricultura de Precisión Ventajas – blogs iadb: https://blogs.iadb.org/sostenibilidad/es/agricultura-de-precision-una-posible-respuesta-al-cambio-climatico-y-a-la-seguridad-alimentaria-pero-es-asequible-para-todos-2/

Schrijver, R. Poppe, K. (Dec, 2016). La agricultura de precisión y el futuro sector agropecuario en Europa. Unidad de prospectiva científica (STOA): https://www.europarl.europa.eu/RegData/etudes/STUD/2016/581892/EPRS_STU(2016)581892_ES.pdf

Rodriguez, M, Cereceda, R. (Nov, 2018 ¿Qué país europeo consume más pesticidas? - euronews: https://es.euronews.com/2018/10/15/que-pais-europeo-consume-mas-pesticidas

(Feb, 2018) Ejemplos de adopción de agricultura de precisión – Hortalizas: https://www.hortalizas.com/tecnologia/ejemplos-de-adopcion-de-agricultura-de-precision/

Boto, Juan. (Feb, 2010) Equipos de aplicación de herbicidas para cultivos extensivos - phytoma: https://www.phytoma.com/la-revista/phytohemeroteca/216-febrero-2010/equipos-de-aplicacion-de-herbicidas-para-cultivos-extensivos

T40 – DJI (2021) Dron pulverizador – DJI: https://dronesforhire.com.au/shop/drone/DJI-Agras-T40

(2021) See and spray – John Deree. https://www.deere.com/en/sprayers/see-spray-ultimate

Berreta, Juan. (Dic 2020). Eco sniper - Agribio: https://www.agribio.com.ar/noticias/canada-prueban-una-caja-con-sensores-para-mapear-malezas-en-trigo#:~:text=El%20Eco%20Sniper%20es%20un,sobre%20la%20totalidad%20del%20lote

Edimar. Pick and Place. Edimar. https://edimar.com/maquina-pick-and-place-industrial/

Xueping Ni, Changying Li, Huanyu Jiang & Fumiomi Takeda. (July, 2020). Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. https://www.nature.com/articles/s41438-020-0323-3

Definición visión artificial – Sick : https://s.sick.com/es-es-soluciones-de-vision-artificial

CNN – IBM: CNN: https://www.ibm.com/cloud/learn/convolutional-neural-networks

YOLO – v7labs: YOLO: https://www.v7labs.com/blog/yolo-object-detection

Redmon, et al. (Dec, 2016) – YOLO9000: Better, Faster, Stronger: https://arxiv.org/pdf/1612.08242.pdf

V Thatte, A. (May, 2020). Evolution of YOLO — YOLO version 1. Medium.: https://towardsdatascience.com/evolution-of-yolo-yolo-version-1-afb8af302bd2

Menegaz, M. (March, 2018). Understanding YOLO. Hackernoon: https://hackernoon.com/understanding-yolo-f5a74bbc7967

ODSC Science. (2018). Overview of the YOLO Object Detection Algorithm: Medium. https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm7b52a745d3e0

He, K., Zhang, X., Ren, S., & Sun, J. (Dec, 2015). Deep Residual Learning for Image Recognition: https://arxiv.org/pdf/1512.03385.pdf

Redmon, J., & Farhadi, A. (Apr, 2018). YOLOv3: An Incremental Improvement: https://arxiv.org/pdf/1804.02767.pdf

Bochkovskiy, A. (April, 2020). Yolo v4, v3 and v2 for Windows and Linux. (GitHub): https://github.com/AlexeyAB/darknet

Jocher, G. (June, 2020). YOLOv5. (GitHub): https://zenodo.org/record/4418161#.X_iH_ugzaUk

Solawetz, J. (Jun, 2020). YOLOv5 New Version - Improvements And Evaluation. Roboflow: https://blog.roboflow.com/yolov5-improvements-and-evaluation/

YOLOv5 (Jan, 2023). YOLOv5 train custom data. (GigHub): https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

Amazon - What is a neural network?. Aws.amazon: https://aws.amazon.com/es/what-is/neural-network/

Sumit Saha. (Dec 16, 2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Towardsdatascience. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Rohith Gandhi (Jul 10, 2018). R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. Towards data science. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Renu Khandewal (Nov 23, 2019). Implementing YOLO on a custom dataset. Towards data science. https://towardsdatascience.com/implementing-yolo-on-a-custom-dataset-20101473ce53

Ethan Yanjia Li (Dec 31, 2019). Dive realty deep into YOLOv3: A beginner's guide. Towards data science. https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e

Ayoosh Kathuria (Apr 24, 2018). What's new in YOLOv3?. Towards data science. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

Huang, G., Liu, Z., & Maaten, L. v. (2018). Densely Connected Convolutional Networks. https://arxiv.org/pdf/1608.06993.pdf

Wang, C.-Y., Mark Liao, H.-Y., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019). CSPNET: A new backbone that can enhance learning capability of CNN. https://arxiv.org/pdf/1911.11929.pdf

Deval Shah (May 26, 2020). YOLOv4 – Version3: Proposed Workflow. Medium. https://medium.com/visionwizard/yolov4-version-3-proposed-workflow-e4fa175b902

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (Sep ,2018). Path Aggregation Network for Instance Segmentation. https://arxiv.org/pdf/1803.01534.pdf

(Nov 24, 2018). Data hacker. CNN Bounding Box Predictions. https://datahacker.rs/deep-learning-bounding-boxes/

Banerjee, A. (Oct 2019). Different Computer Vision Tasks. Medium. https://ananya-banerjee.medium.com/different-computer-vision-tasks-b3b49bbae891

Lu, C. Krishna, R. Bernstein, M. Li, F (2016). Visual Relationship Detection with Language priors. Standford. https://cs.stanford.edu/people/ranjaykrishna/vrd/