

Unidad 3 - Entrenamiento de modelos de visión

UNR - TUIA - Procesamiento de Imágenes y Visión por Computadora

Docente teoría: Juan Pablo Manson

Docentes de práctica: Lucas Brugé, Constantino Ferrucci

Cuaderno con ejercicios prácticos:

Google Colab

🔗 https://colab.research.google.com/drive/1Fxo15uyAYEPQhl98UlvoJSlL_M0oeVO?usp=sharing



1. Datasets

La visión por computadora es un campo de la inteligencia artificial que permite a las máquinas interpretar y comprender el mundo visual a través de imágenes y videos, emulando la capacidad del sistema visual humano. En el corazón de este campo se encuentran los datasets, conjuntos estructurados de datos que son esenciales para el entrenamiento, evaluación y desarrollo de algoritmos de visión por computadora.

Importancia de los Datasets en la Visión por Computadora

Los datasets son fundamentales en la visión por computadora por varias razones fundamentales:

1. **Entrenamiento de Modelos:** Los modelos de aprendizaje profundo, especialmente las redes neuronales convolucionales (CNN), requieren grandes volúmenes de datos etiquetados para aprender a reconocer patrones y características en las imágenes. Sin datasets adecuados, el desarrollo de modelos precisos y robustos sería imposible.
2. **Evaluación y Benchmarking:** Los datasets estándar permiten la evaluación objetiva del rendimiento de diferentes algoritmos y modelos. Proporcionan un marco común para comparar los resultados y establecer benchmarks que impulsan el progreso en el campo.
3. **Diversidad y Generalización:** Los datasets diversos, que abarcan una amplia variedad de escenarios, objetos y condiciones, son esenciales para asegurar que los modelos desarrollados sean capaces de generalizar bien a nuevas situaciones y no se limiten a los datos específicos con los que fueron entrenados.

Tipos de Datasets

Existen varios tipos de datasets utilizados en la visión por computadora, cada uno adecuado para diferentes tareas y aplicaciones. Los más comunes son:

1. **Datasets de Clasificación:** Utilizados para entrenar modelos que asignan etiquetas a imágenes completas. Ejemplos destacados incluyen ImageNet y CIFAR-10.
2. **Datasets de Detección de Objetos:** Diseñados para entrenar modelos que identifican y localizan objetos dentro de las imágenes. Ejemplos incluyen MS COCO y PASCAL VOC.
3. **Datasets de Segmentación:** Estos datasets contienen anotaciones detalladas que permiten a los modelos delinear y segmentar objetos dentro de una imagen. MS COCO también es utilizado para este propósito, junto con Cityscapes para la segmentación urbana.
4. **Datasets de Tracking de Múltiples Objetos:** Los datasets de tracking de múltiples objetos son esenciales para desarrollar y evaluar algoritmos capaces de seguir varios objetos en movimiento a través de secuencias de video. Entre los más destacados se encuentran **MOTChallenge**, que proporciona una serie de escenarios de vigilancia urbana y entornos públicos para el seguimiento de peatones; **UA-DETRAC**, diseñado específicamente para el seguimiento de

vehículos en condiciones de tráfico diverso; y **KITTI Tracking**, que ofrece datos capturados desde un vehículo en movimiento, enfocándose en el seguimiento de coches, peatones y ciclistas en entornos urbanos.

5. **Datasets Especializados:** Algunos datasets están diseñados para tareas muy específicas, como la detección de rostros (Labeled Faces in the Wild) o la clasificación de escenas (MIT Places).

Desafíos y Consideraciones

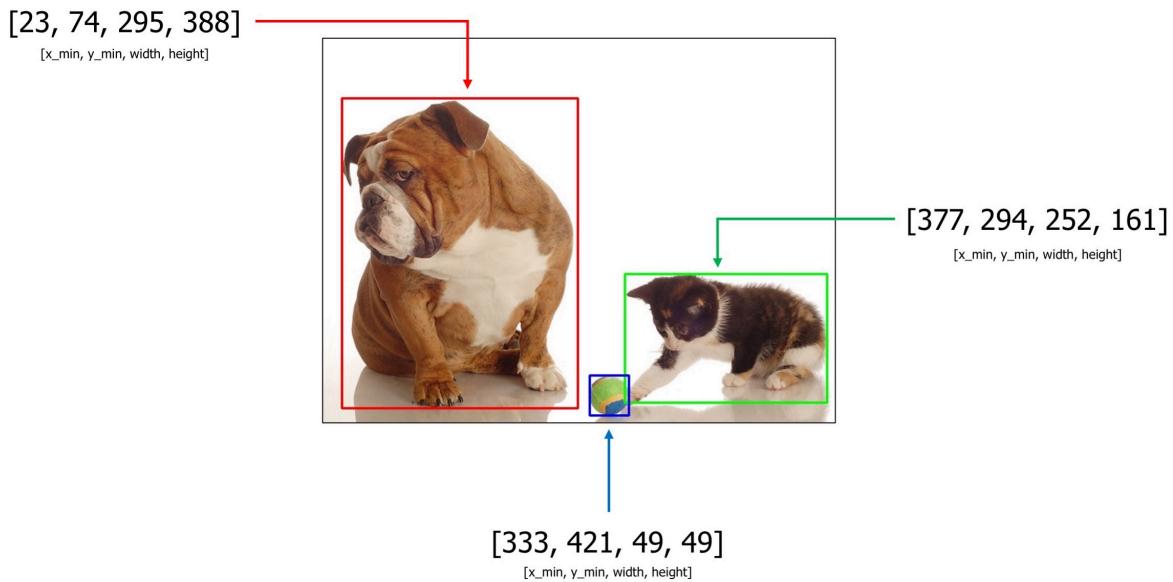
A pesar de su importancia, el uso de datasets en la visión por computadora presenta varios desafíos:

1. **Calidad de los Datos:** La precisión de los modelos depende en gran medida de la calidad de los datos de entrenamiento. Los datos mal etiquetados o ruidosos pueden llevar a modelos inexactos.
2. **Tamaño y Accesibilidad:** Los datasets grandes requieren considerables recursos de almacenamiento y procesamiento. Además, no todos los datasets están disponibles públicamente, lo que puede limitar la investigación.
3. **Ética y Privacidad:** El uso de imágenes que contienen información personal plantea preocupaciones sobre la privacidad y la ética. Es importante asegurar que los datasets cumplan con las normativas y estándares éticos.
4. **Sesgo en los Datasets:** Muchos datasets pueden contener sesgos inherentes debido a la forma en que se recolectan o anotan los datos. Estos sesgos pueden llevar a modelos que perpetúan o amplifican discriminaciones, lo que resulta en aplicaciones injustas o poco precisas. Es fundamental identificar y mitigar estos sesgos para desarrollar modelos equitativos y justos.

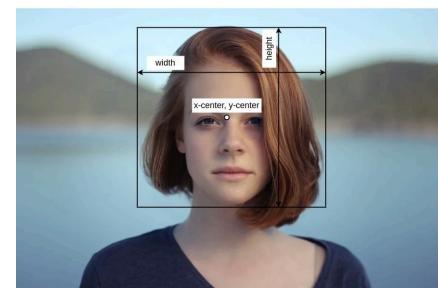
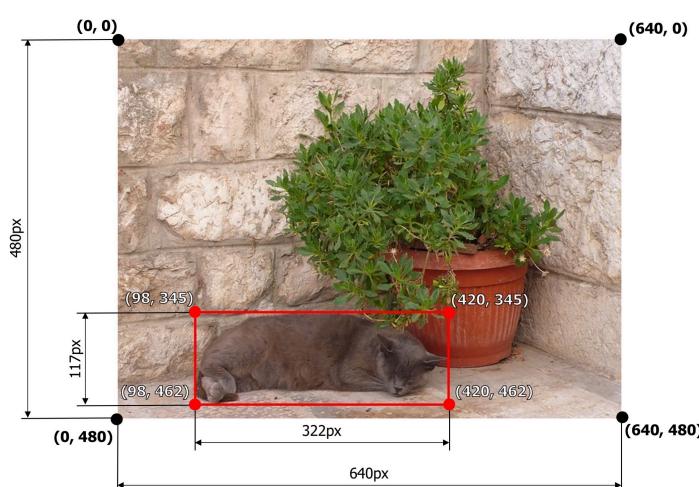
Tipos de anotación o etiquetado

Bounding boxes

Las cajas delimitadoras son el tipo de anotación más comúnmente usado en la visión por computadora. Son cajas rectangulares que definen la ubicación del objeto objetivo. Se determinan mediante las coordenadas x e y en la esquina superior izquierda y las coordenadas x e y en la esquina inferior derecha del rectángulo. Las cajas delimitadoras se utilizan generalmente en tareas de detección y localización de objetos.



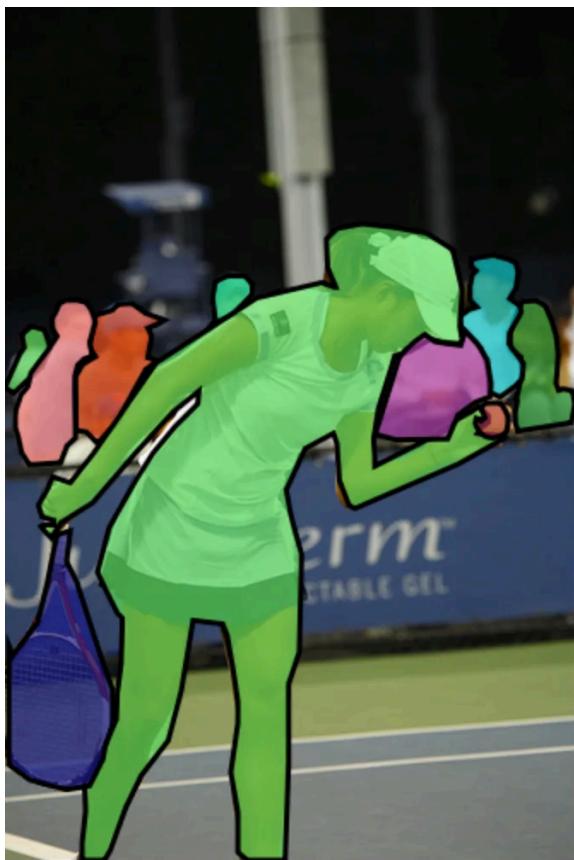
Las cajas delimitadoras suelen representarse por dos coordenadas (x_1, y_1) y (x_2, y_2) o por una coordenada (x_1, y_1) y el ancho (w) y la altura (h) de la caja delimitadora. Generalmente, se utiliza la esquina superior izquierda como x_1, y_1 , y la esquina inferior derecha como x_2, y_2 , aunque eso depende del criterio del dataset.



Algunos formatos como Yolo, trabajan con el centro de la caja delimitadora, el ancho y la altura, para definir el rectángulo.

Segmentación Poligonal

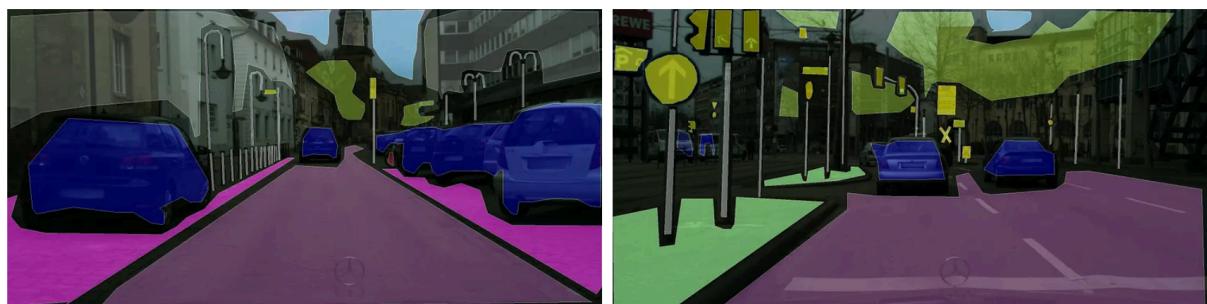
Los objetos no siempre tienen forma rectangular. Con esta idea, la segmentación poligonal es otro tipo de anotación de datos donde se usan polígonos complejos en lugar de rectángulos para definir la forma y ubicación del objeto de manera mucho más precisa.



Segmentación Semántica

La segmentación semántica es una anotación a nivel de píxel, donde cada píxel de la imagen se asigna a una clase. Estas clases pueden ser peatón, coche, autobús, carretera, vereda, etc., y cada píxel tiene un significado semántico.

La segmentación semántica se utiliza principalmente en casos donde el contexto ambiental es muy importante. Por ejemplo, se utiliza en coches autónomos y robótica porque los modelos necesitan entender el entorno en el que operan.



Cuboides 3D (3D cuboid)

Los cuboides 3D son similares a las cajas delimitadoras, pero con información adicional de profundidad sobre el objeto. Con los cuboides 3D, se puede obtener una representación tridimensional del objeto, lo que permite a los sistemas distinguir características como el volumen y la posición en un espacio 3D.

Un caso de uso de los cuboides 3D es en los coches autónomos, donde se puede utilizar la información de profundidad para medir la distancia de los objetos desde el automóvil.



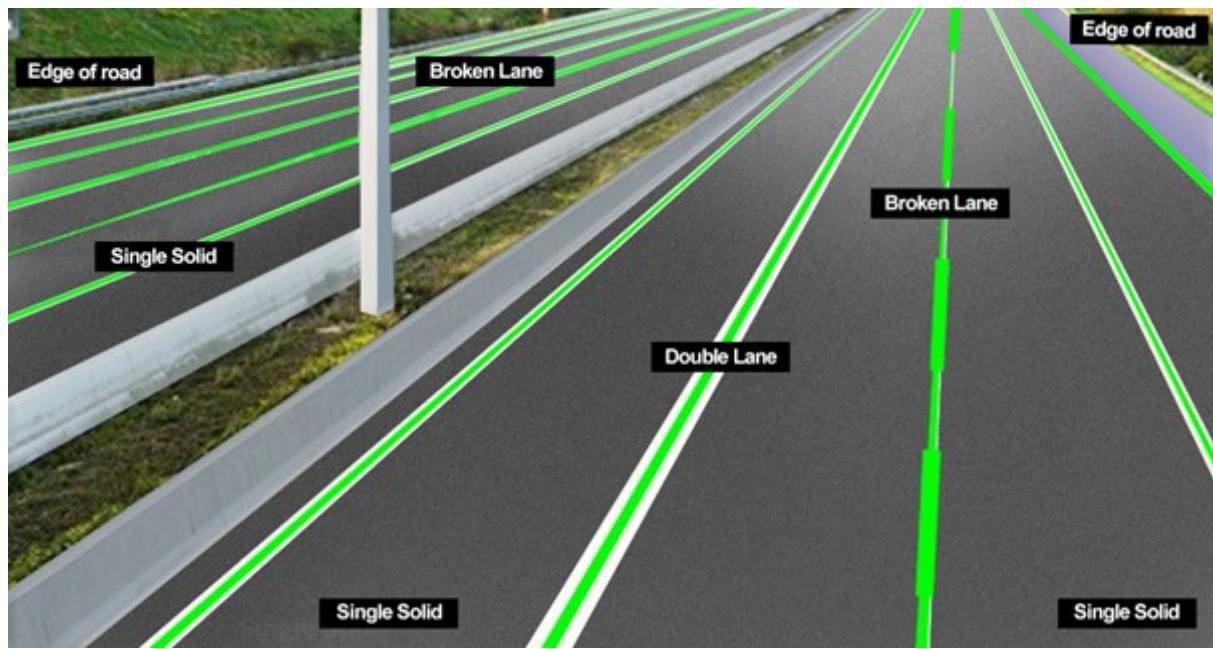
Puntos Clave y Puntos de Referencia

La anotación de puntos clave y puntos de referencia se utiliza para detectar objetos pequeños, estructuras y variantes de forma mediante la creación de puntos en la imagen. Este tipo de anotación es útil para detectar características faciales, expresiones faciales, partes del cuerpo humano y poses.



Líneas y Splines

Como su nombre indica, este tipo de anotación se crea utilizando líneas y splines. Las splines son funciones matemáticas utilizadas para crear curvas suaves que pasan por un conjunto de puntos de control. Se usa comúnmente en vehículos autónomos para la detección y reconocimiento de carriles.

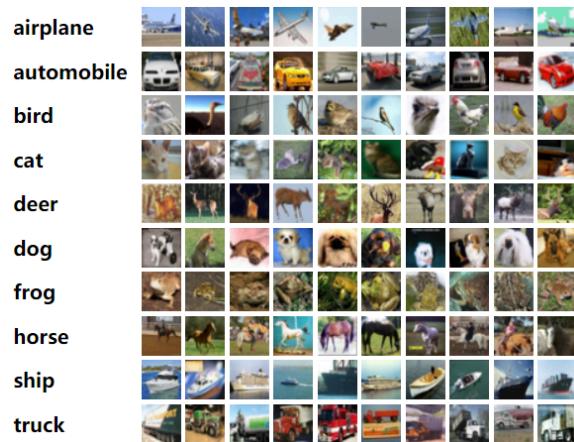


Datasets populares

CIFAR-10 (Canadian Institute for Advanced Research, 10 classes)

Origen: <https://www.cs.toronto.edu/~kriz/cifar.html> - Krizhevsky et al. Learning multiple layers of features from tiny images

El conjunto de datos CIFAR-10 (Canadian Institute for Advanced Research, 10 clases) es un subconjunto del conjunto de datos Tiny Images y consta de 60,000 imágenes en color de 32×32 . Las imágenes están etiquetadas con una de las 10 clases mutuamente excluyentes: avión, automóvil (pero no camión o camioneta), pájaro, gato, ciervo, perro, rana, caballo, barco y camión (pero no camioneta). Hay 6000 imágenes por clase, con 5000 imágenes de entrenamiento y 1000 imágenes de prueba por clase.



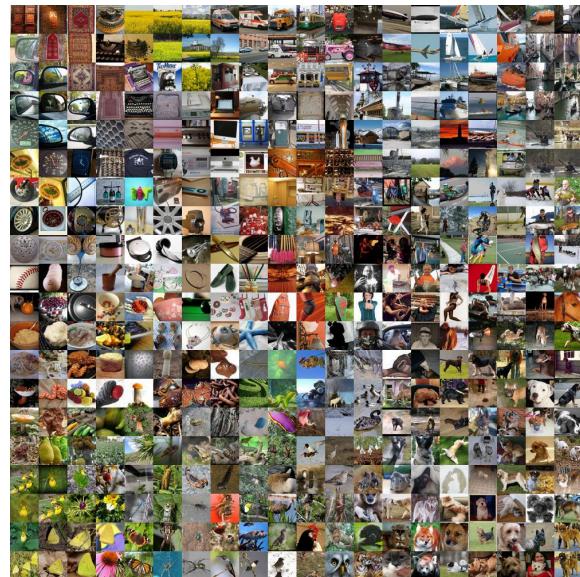
Los criterios para decidir si una imagen pertenece a una clase fueron los siguientes:

1. El nombre de la clase debería estar en lo alto de la lista de respuestas probables a la pregunta "¿Qué hay en esta imagen?"
2. La imagen debería ser fotorrealista. Se instruyó a los etiquetadores que rechazaran los dibujos lineales.
3. La imagen debería contener solo una instancia prominente del objeto al que se refiere la clase. El objeto puede estar parcialmente oculto o visto desde un punto de vista inusual, siempre y cuando su identidad sea clara para el etiquetador.

ImageNet

Origen: Jia Deng et al. in ImageNet: A large-scale hierarchical image database

El conjunto de datos ImageNet contiene 14,197,122 imágenes anotadas según la jerarquía de WordNet. Desde 2010, el conjunto de datos se utiliza en el ImageNet Large Scale Visual Recognition Challenge (ILSVRC), un punto de referencia en la clasificación de imágenes y la detección de objetos. El conjunto de datos publicado contiene un conjunto de imágenes de entrenamiento anotadas manualmente. También se publica un conjunto de imágenes de prueba, cuyas anotaciones manuales se mantienen en reserva.



Las anotaciones del ILSVRC se dividen en dos categorías: (1) anotación a nivel de imagen de una etiqueta binaria para la presencia o ausencia de una clase de objeto en la imagen, por ejemplo, "hay coches en esta imagen" pero "no hay tigres", y (2) anotación a nivel de objeto de una caja delimitadora ajustada y una etiqueta de clase alrededor de una instancia de objeto en la imagen, por ejemplo, "hay un destornillador centrado en la posición (20,25) con un ancho de 50 píxeles y una altura de 30 píxeles". El proyecto ImageNet no posee los derechos de autor de las imágenes, por lo tanto, solo se proporcionan miniaturas y URLs de las imágenes.

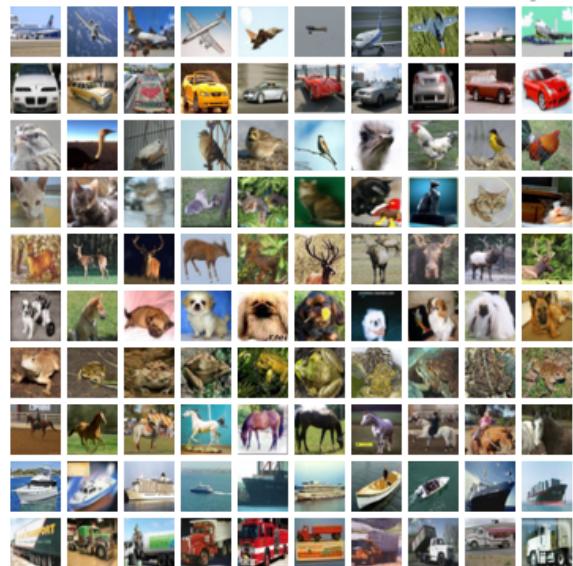
- Número total de sinónimos de WordNet no vacíos: 21841
- Número total de imágenes: 14,197,122
- Número de imágenes con anotaciones de caja delimitadora: 1,034,908
- Número de sinónimos con características SIFT: 1000
- Número de imágenes con características SIFT: 1,2 millones

MS COCO (Microsoft Common Objects in Context)

Origen: Lin et al. in Microsoft COCO: Common Objects in Context

El conjunto de datos MS COCO (Microsoft Common Objects in Context) es un conjunto de datos a gran escala para la detección de objetos, segmentación, detección de puntos clave y generación de descripciones (captioning). El conjunto de datos consta de 328,000 imágenes.

Divisiones (Splits): La primera versión del conjunto de datos MS COCO se lanzó en 2014. Contiene 164,000 imágenes divididas en conjuntos de entrenamiento (83,000), validación (41,000) y prueba (41,000). En 2015 se lanzó un conjunto de prueba adicional de 81,000 imágenes, que incluye todas las imágenes de prueba anteriores y 40,000 imágenes nuevas. Basado en la retroalimentación de la comunidad, en 2017 la división de entrenamiento/validación cambió de 83,000/41,000 a 118,000/5,000. La nueva división utiliza las mismas imágenes y anotaciones.



<https://cocodataset.org/>

El conjunto de prueba de 2017 es un subconjunto de 41,000 imágenes del conjunto de prueba de 2015. Además, el lanzamiento de 2017 contiene un nuevo conjunto de datos no anotado de 123,000 imágenes.

Anotaciones: El conjunto de datos tiene anotaciones para:

- **Detección de objetos:** Cajas delimitadoras y máscaras de segmentación por instancia con 80 categorías de objetos.
- **Generación de descripciones (captioning):** Descripciones en lenguaje natural de las imágenes (ver MS COCO Captions).
- **Detección de puntos clave:** Contiene más de 200,000 imágenes y 250,000 instancias de personas etiquetadas con puntos clave (17 puntos clave posibles, como ojo izquierdo, nariz, cadera derecha, tobillo derecho).
- **Segmentación de imágenes de "cosas":** Máscaras de segmentación por píxel con 91 categorías de "cosas", como césped, pared, cielo (ver MS COCO Stuff).
- **Panóptico:** Segmentación completa de escenas, con 80 categorías de "cosas" ("things", como persona, bicicleta, elefante) y un subconjunto de 91 categorías de "cosas" ("stuff", césped, cielo, camino).
- **Dense Pose:** Más de 39,000 imágenes y 56,000 instancias de personas etiquetadas con anotaciones DensePose. Cada persona etiquetada está anotada con un id de instancia y un mapeo entre los píxeles de la imagen que

pertenecen al cuerpo de esa persona y un modelo 3D de plantilla. Las anotaciones están disponibles públicamente solo para imágenes de entrenamiento y validación.

Podemos ver las imágenes de coco con el siguiente buscador:

COCO - Common Objects in Context

 <https://cocodataset.org/#explore>

Google Colab

 <https://colab.research.google.com/drive/1esxekYdBU4QkMxcpWWCjr6qTt-mTL97i?usp=sharing>

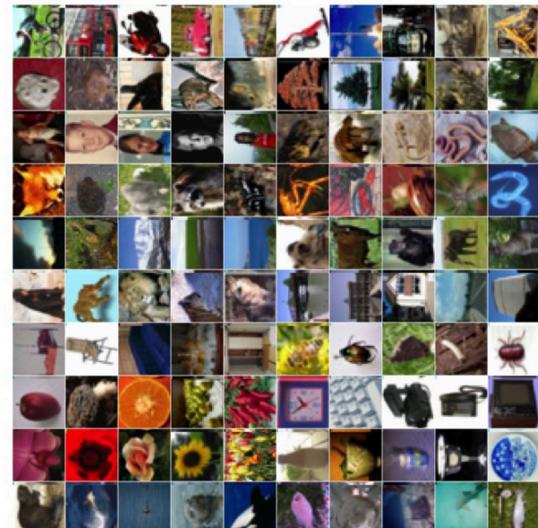


CIFAR-100 (Canadian Institute for Advanced Research, 10 classes)

Origen: Krizhevsky et al. [Learning multiple layers of features from tiny images](#)

El conjunto de datos CIFAR-100 (Canadian Institute for Advanced Research, 100 clases) es un subconjunto del conjunto de datos Tiny Images y consta de 60000 imágenes en color de 32×32 píxeles. Las 100 clases en el CIFAR-100 están agrupadas en 20 superclases. Hay 600 imágenes por clase. Cada imagen viene con una etiqueta "fina" (la clase a la que pertenece) y una etiqueta "gruesa" (la superclase a la que pertenece). Hay 500 imágenes de entrenamiento y 100 imágenes de prueba por clase.

Los criterios para decidir si una imagen pertenece a una clase fueron los siguientes:



<https://www.cs.toronto.edu/~kriz/cifar.html>

1. El nombre de la clase debería estar en lo alto de la lista de respuestas probables a la pregunta “¿Qué hay en esta imagen?”
2. La imagen debería ser fotorrealista. Se instruyó a los etiquetadores que rechazaran los dibujos lineales.
3. La imagen debería contener solo una instancia prominente del objeto al que se refiere la clase. El objeto puede estar parcialmente oculto o visto desde un punto de vista inusual, siempre y cuando su identidad sea clara para el etiquetador.

MNIST

Origen: LeCun et al. in [Gradient-based learning applied to document recognition](#) - <http://yann.lecun.com/exdb/mnist/>

La base de datos MNIST (Modified National Institute of Standards and Technology database) es una gran colección de dígitos manuscritos. Tiene un conjunto de entrenamiento de 60,000 ejemplos y un conjunto de prueba de 10,000 ejemplos.

Es un subconjunto de una base de datos más grande, NIST Special Database 3 (dígitos escritos por empleados de la Oficina del Censo de los Estados Unidos) y Special Database 1 (dígitos escritos por estudiantes de secundaria), que contienen imágenes monocromáticas de dígitos manuscritos.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

Los dígitos han sido normalizados en tamaño y centrados en una imagen de tamaño fijo. Las imágenes originales en blanco y negro (bilevel) del NIST fueron normalizadas en tamaño para caber en una caja de 20×20 píxeles, preservando su relación de aspecto. Las imágenes resultantes contienen niveles de gris como resultado de la técnica de suavizado (anti-aliasing) utilizada por el algoritmo de normalización. Las imágenes fueron centradas en una imagen de 28×28 píxeles mediante el cálculo del centro de masa de los píxeles y trasladando la imagen para posicionar este punto en el centro del campo de 28×28 píxeles.

Cityscapes

Origen: Cordts et al. in [The Cityscapes Dataset for Semantic Urban Scene Understanding](https://www.cityscapes-dataset.com/dataset-overview/) - <https://www.cityscapes-dataset.com/dataset-overview/>

Cityscapes es una base de datos a gran escala que se centra en la comprensión semántica de escenas urbanas callejeras. Proporciona anotaciones semánticas, por instancia y de píxeles densos para 30 clases agrupadas en 8 categorías (superficies planas, humanos, vehículos, construcciones, objetos, naturaleza, cielo y vacío).



<https://www.cityscapes-dataset.com/dataset-overview/>

El conjunto de datos consta de aproximadamente 5000 imágenes anotadas finamente y 20000 imágenes anotadas de manera gruesa. Los datos fueron capturados en 50 ciudades durante varios meses, en distintos momentos del día y en condiciones climáticas favorables. Originalmente, se grabaron como video, por lo que los fotogramas se seleccionaron manualmente para tener las siguientes características: gran número de objetos dinámicos, disposición variable de la escena y fondo variable.

KITTI

Origen: Andreas Geiger et al. in [Are we ready for autonomous driving? The KITTI vision benchmark suite](http://www.cvlibs.net/datasets/kitti/) - <http://www.cvlibs.net/datasets/kitti/>

KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) es uno de los conjuntos de datos más populares para su uso en robótica móvil y conducción autónoma. Consiste en horas de escenarios de tráfico grabados con una variedad de modalidades de sensores, incluyendo cámaras RGB de alta resolución, cámaras estéreo en escala de grises y un escáner láser 3D.



A pesar de su popularidad, el conjunto de datos en sí no contiene verdades fundamentales (ground truth) para la segmentación semántica. Sin embargo, varios investigadores han anotado manualmente partes del conjunto de datos para adaptarlas a sus necesidades.

Álvarez et al. generaron verdades fundamentales para 323 imágenes del desafío de detección de carreteras con tres clases: carretera, vertical y cielo. Zhang et al. anotaron 252 adquisiciones (140 para entrenamiento y 112 para prueba), incluyendo escaneos RGB y Velodyne, del desafío de seguimiento para diez categorías de objetos: edificio, cielo, carretera, vegetación, acera, coche, peatón, ciclista, señal/polo y valla. Ros et al. etiquetaron 170 imágenes de entrenamiento y 46 imágenes de prueba (del desafío de odometría visual) con 11 clases: edificio, árbol, cielo, coche, señal, carretera, peatón, valla, poste, acera y ciclista.

| https://www.cvlibs.net/datasets/kitti/eval_semantics.php

SVHN (Street View House Numbers)

Origen: Netzer et al. in [Reading digits in natural images with unsupervised feature learning - http://ufldl.stanford.edu/housenumbers/](http://ufldl.stanford.edu/housenumbers/)

Street View House Numbers (SVHN) es un conjunto de datos de referencia para la clasificación de dígitos que contiene 600,000 imágenes RGB de 32×32 de dígitos impresos (del 0 al 9) recortadas de fotografías de placas de números de casas. Las imágenes recortadas están centradas en el dígito de interés, pero los dígitos cercanos y otros distractores se mantienen en la imagen. SVHN tiene tres conjuntos: conjuntos de entrenamiento, prueba y un conjunto adicional con 530,000 imágenes que son menos difíciles y pueden utilizarse para ayudar en el proceso de entrenamiento.



<http://ufldl.stanford.edu/housenumbers/>

CelebA (CelebFaces Attributes Dataset)

Origen: Liu et al. in [Deep Learning Face Attributes in the Wild](#) -
<https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

El conjunto de datos CelebFaces
Attributes contiene 202,599 imágenes
de rostros de tamaño 178×218 de
10,177 celebridades, cada una anotada
con 40 etiquetas binarias que indican
atributos faciales como color de
cabello, género y edad.



nuScenes

Origen: Caesar et al. in [nuScenes: A multimodal dataset for autonomous driving](#) -
<https://www.nuscenes.org/>

El conjunto de datos nuScenes es un
conjunto de datos a gran escala para la
conducción autónoma. El conjunto de
datos tiene cajas delimitadoras 3D para
1000 escenas recolectadas en Boston
y Singapur. Cada escena dura 20
segundos y está anotada a 2 Hz.



Esto resulta en un total de 28,130 muestras para entrenamiento, 6,019 muestras
para validación y 6,008 muestras para prueba. El conjunto de datos tiene la suite
completa de datos de vehículos autónomos: LiDAR de 32 haces, 6 cámaras y
radares con cobertura completa de 360°. El desafío de detección de objetos 3D
evalúa el rendimiento en 10 clases: coches, camiones, autobuses, remolques,
vehículos de construcción, peatones, motocicletas, bicicletas, conos de tráfico y
barreras.

ShapeNet

Origen: Chang et al. in [ShapeNet: An Information-Rich 3D Model Repository](#)

ShapeNet es un repositorio a gran escala para modelos CAD 3D desarrollado por investigadores de la Universidad de Stanford, la Universidad de Princeton y el Instituto Tecnológico Toyota en Chicago, EE. UU. El repositorio contiene más de 300 millones de modelos, con 220,000 clasificados en 3,135 clases organizadas utilizando las relaciones hipónimo-hiperónimo de WordNet. El subconjunto ShapeNet Parts contiene 31,693 mallas categorizadas en 16 clases comunes de objetos (por ejemplo, mesa, silla, avión, etc.). La verdad fundamental de cada forma contiene de 2 a 5 partes (con un total de 50 clases de partes).



<https://www.shapenet.org/>

Visual Question Answering (VQA)

Origen: Agrawal et al. in VQA: Visual Question Answering

Visual Question Answering (VQA) es un conjunto de datos que contiene preguntas abiertas sobre imágenes. Estas preguntas requieren una comprensión de la visión, el lenguaje y el conocimiento común para ser respondidas. La primera versión del conjunto de datos se lanzó en octubre de 2015. VQA v2.0 se lanzó en abril de 2017.

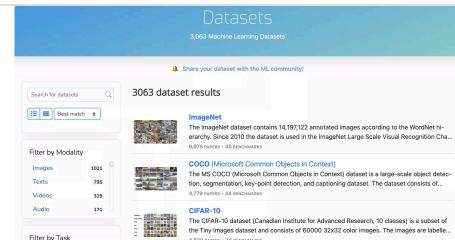


| Podemos encontrar más datasets en los siguientes links:

Papers with Code - Machine Learning Datasets

9843 datasets • 127513 papers with code.

🌐 <https://paperswithcode.com/datasets>



The screenshot shows a search interface with filters for Modality (Images, Texts, Videos, Audio) and Task. It displays 3063 dataset results, including ImageNet (14,197,132 images), MS COCO (82,779 images), and Tiny ImageNet (4,620 images). Each dataset entry includes a thumbnail, name, description, and a link to the dataset page.

Hugging Face – The AI community building the future.

We're on a journey to advance and democratize artificial intelligence through open source and open science.

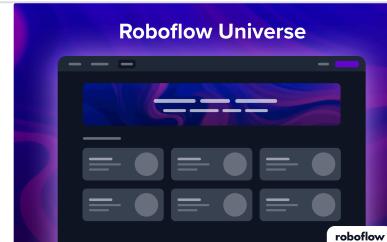
👉 <https://huggingface.co/datasets>



Roboflow Universe: Open Source Computer Vision Community

Download free, open source datasets and pre-trained computer vision machine learning models.

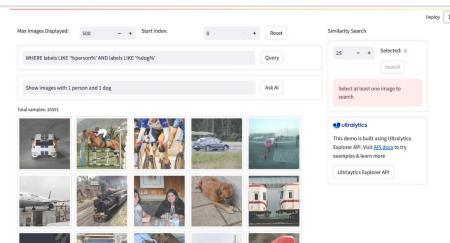
🌐 <https://universe.roboflow.com/>



Datasets Overview

Explore various computer vision datasets supported by Ultralytics for object detection, segmentation, pose estimation, image classification, and multi-object tracking.

🌐 <https://docs.ultralytics.com/datasets/>



List of datasets in computer vision and image processing

This is a list of datasets for machine learning research. It is part of the list of datasets for machine-learning research. These datasets consist primarily of images or videos for tasks such as object detection, facial recognition, and multi-label classification.

🌐 https://en.wikipedia.org/wiki/List_of_datasets_in_computer_vision_and_image_processing

Formatos de anotación

Si bien cada gran dataset generalmente utiliza su propio formato, la comunidad de desarrolladores suele adoptar algunos más que otros por su popularidad. A continuación se muestran los formatos más utilizados en la anotación de imágenes:

COCO

COCO tiene varios tipos de anotaciones: detección de objetos, detección de puntos clave, segmentación de objetos (stuff segmentation), segmentación panóptica, densepose y descripción de imágenes (image captioning). Las anotaciones se almacenan usando JSON. Tengamos en cuenta que la API de COCO descrita en la página de descarga puede ser utilizada para acceder y manipular todas las anotaciones. Todas las anotaciones comparten la misma estructura básica de datos a continuación:

```
{  
    "info": info,  
    "images": [image],  
    "annotations": [annotation],  
    "licenses": [license],  
}
```

Donde cada elemento mencionado contiene:

```
info {  
    "year": int,  
    "version": str,  
    "description": str,  
    "contributor": str,  
    "url": str,  
    "date_created": datetime,  
}
```

```
image {  
    "id": int,  
    "width": int, "height": int,  
    "file_name": str,  
    "license": int,  
    "flickr_url": str,  
    "coco_url": str,  
    "date_captured": datetime,  
}
```

```
license {  
    "id": int,  
    "name": str,
```

```
    "url": str,  
}
```

Detección de Objetos

Cada anotación de instancia de objeto contiene varios campos, incluyendo el ID de categoría y la máscara de segmentación del objeto. El formato de segmentación depende de si la instancia representa un solo objeto (`iscrowd=0`, usando polígonos) o una colección de objetos (`iscrowd=1`, usando [RLE](#)). Un solo objeto (`iscrowd=0`) puede necesitar múltiples polígonos si está oculto. Las anotaciones de multitudes (`iscrowd=1`) se usan para etiquetar grandes grupos de objetos (por ejemplo, una multitud de personas). Además, se proporciona una caja delimitadora para cada objeto, y el campo de categorías almacena el mapeo de ID de categoría a nombres de categoría y supercategoría.

```
annotation{  
    "id": int,  
    "image_id": int,  
    "category_id": int,  
    "segmentation": RLE or [polygon],  
    "area": float,  
    "bbox": [x,y,width,height],  
    "iscrowd": 0 or 1,  
}  
  
categories[  
    {  
        "id": int,  
        "name": str,  
        "supercategory": str,  
    }  
]
```

Keypoint Detection

Una anotación de punto clave contiene todos los datos de la anotación del objeto (incluyendo ID, bbox, etc.) y dos campos adicionales. Primero, "keypoints" es un array de longitud $3k$ donde k es el número total de puntos clave definidos para la categoría. Cada punto clave tiene una ubicación x,y indexada desde 0 y una

bandera de visibilidad v definida como $v=0$: no etiquetado (en cuyo caso $x=y=0$), $v=1$: etiquetado pero no visible, y $v=2$: etiquetado y visible. Un punto clave se considera visible si está dentro del segmento del objeto. "num_keypoints" indica el número de puntos clave etiquetados ($v>0$) para un objeto dado (muchos objetos, por ejemplo, multitudes y objetos pequeños, tendrán num_keypoints=0).

Finalmente, para cada categoría, la estructura de categorías tiene dos campos adicionales: "keypoints", que es un array de longitud k con los nombres de los puntos clave, y "skeleton", que define la conectividad mediante una lista de pares de bordes de puntos clave y se utiliza para visualización. Actualmente, los puntos clave solo están etiquetados para la categoría persona (para la mayoría de las instancias de personas medianas/grandes que no son multitudes).

```
annotation{  
    "keypoints": [x1,y1,v1,...],  
    "num_keypoints": int,  
    "[cloned)": ...,  
}  
  
categories[{  
    "keypoints": [str],  
    "skeleton": [edge],  
    "[cloned)": ...,  
}]  
  
"[cloned)": denotes fields copied from object detection annotations defined above
```

Stuff Segmentation

El formato de anotación de "stuff" es idéntico y completamente compatible con el formato de detección de objetos mencionado anteriormente (excepto que `iscrowd` no es necesario y se establece en 0 por defecto).



Proporcionamos anotaciones en formato JSON y PNG para facilitar el acceso, así como scripts de conversión entre ambos formatos. En el formato JSON, cada categoría presente en una imagen se codifica con una sola anotación RLE (ver la API de Mask para más detalles). El `category_id` representa el ID de la categoría

actual de "stuff". Para más detalles sobre las categorías de "stuff" y supercategorías, consulta la página de evaluación de "stuff".

Panoptic Segmentation

Para la tarea panóptica, cada estructura de anotación es por imagen en lugar de por objeto. Cada anotación por imagen tiene dos partes: (1) un PNG que almacena la segmentación de imagen sin clase y (2) una estructura JSON que almacena la información semántica para cada segmento de la imagen. En detalle:

1. Para emparejar una anotación con una imagen, usa el campo `image_id` (es decir, `annotation.image_id==image.id`).
2. Para cada anotación, los IDs de segmentos por píxel se almacenan como un único PNG en `annotation.file_name`. Los PNG están en una carpeta con el mismo nombre que el JSON, es decir, `annotations/name/` para `annotations/name.json`. Cada segmento (ya sea de "stuff" o "thing") recibe un ID único. Los píxeles no etiquetados (`void`) se asignan a un valor de 0. Cuando cargues el PNG como una imagen RGB, necesitarás calcular los IDs mediante `ids=R+G256+B256^2`.
3. Para cada anotación, la información por segmento se almacena en `annotation.segments_info`. `segment_info.id` almacena el ID único del segmento y se utiliza para recuperar la máscara correspondiente del PNG (`ids==segment_info.id`). `category_id` proporciona la categoría semántica e `iscrowd` indica que el segmento abarca un grupo de objetos (relevante solo para categorías de "thing"). Los campos `bbox` y `area` proporcionan información adicional sobre el segmento.
4. La tarea panóptica de COCO tiene las mismas categorías de "thing" que la tarea de detección, mientras que las categorías de "stuff" difieren de las de la tarea de "stuff" (para detalles, consulta la página de evaluación panóptica). Finalmente, cada estructura de categoría tiene dos campos adicionales: `isthing`, que distingue entre categorías de "stuff" y "thing", y `color`, útil para la visualización consistente.

```
annotation{
    "image_id": int,
    "file_name": str,
    "segments_info": [segment_info],
}
```

```

segment_info{
    "id": int,
    "category_id": int,
    "area": int,
    "bbox": [x,y,width,height],
    "iscrowd": 0 or 1,
}

categories[
{
    "id": int,
    "name": str,
    "supercategory": str,
    "isthing": 0 or 1,
    "color": [R,G,B],
}
]

```

Image Captioning

Estas anotaciones se utilizan para almacenar descripciones de imágenes. Cada descripción describe la imagen especificada y cada imagen tiene al menos 5 descripciones (algunas imágenes tienen más). Aquí podemos ver más sobre la tarea de [creación de descripciones](#).

```

annotation {
    "id": int,
    "image_id": int,
    "caption": str,
}

```

Ejemplo de un archivo JSON en formato COCO

Aquí vemos un ejemplo de anotación COCO, para tareas de detección de objetos, con los bounding box:

```

{
    "info": {
        "year": "2020",

```

```
"version": "1",
"description": "Exported from roboflow.ai",
"contributor": "Roboflow",
"url": "https://app.roboflow.ai/datasets/hard-hat-sample/1",
"date_created": "2000-01-01T00:00:00+00:00"
},
"licenses": [
{
    "id": 1,
    "url": "https://creativecommons.org/publicdomain/zero/1.0/",
    "name": "Public Domain"
}
],
"categories": [
{
    "id": 0,
    "name": "Workers",
    "supercategory": "none"
},
{
    "id": 1,
    "name": "head",
    "supercategory": "Workers"
},
{
    "id": 2,
    "name": "helmet",
    "supercategory": "Workers"
},
{
    "id": 3,
    "name": "person",
    "supercategory": "Workers"
}
],
"images": [
{
    "id": 0,
    "license": 1,
    "file_name": "0001.jpg",
    "width": 1280,
    "height": 720
}
]
```

```
        "height": 275,  
        "width": 490,  
        "date_captured": "2020-07-20T19:39:26+00:00"  
    }  
,  
    "annotations": [  
        {  
            "id": 0,  
            "image_id": 0,  
            "category_id": 2,  
            "bbox": [  
                45,  
                2,  
                85,  
                85  
            ],  
            "area": 7225,  
            "segmentation": [],  
            "iscrowd": 0  
        },  
        {  
            "id": 1,  
            "image_id": 0,  
            "category_id": 2,  
            "bbox": [  
                324,  
                29,  
                72,  
                81  
            ],  
            "area": 5832,  
            "segmentation": [],  
            "iscrowd": 0  
        }  
    ]  
}
```

Para más información sobre el formato COCO:

COCO - Common Objects in Context



<https://cocodataset.org/#format-data>

COCO Dataset Format - Complete Walkthrough

A detailed walkthrough of the COCO Dataset JSON Format, specifically for object detection (instance segmentations). 
CORRECTION BELOW 

▶ https://www.youtube.com/watch?v=h6s61a_pqfM



COCO Dataset: All You Need to Know to Get Started

COCO dataset is commonly used in machine learning—both for research and practical applications. Let's dive deeper into the COCO dataset and its significance for computer vision tasks.

V7 <https://www.v7labs.com/blog/coco-dataset-guide>



PASCAL VOC

El formato Pascal VOC XML es ampliamente utilizado para anotar imágenes en tareas de visión por computadora. Cada archivo XML corresponde a una imagen y contiene los siguientes elementos principales:

1. **annotation**: Elemento raíz que contiene toda la información.
 2. **folder**: Nombre de la carpeta que contiene la imagen.
 3. **filename**: Nombre del archivo de imagen.
 4. **path**: Ruta completa al archivo de imagen.
 5. **source**: Información sobre la base de datos.
 6. **size**: Dimensiones de la imagen, incluyendo ancho, alto y profundidad.
 7. **segmented**: Indica si la imagen está segmentada (0 o 1).
 8. **object**: Información sobre cada objeto en la imagen, incluyendo:
 - **name**: Nombre de la clase del objeto.
 - **pose**: Posición del objeto (opcional).
 - **truncated**: Indica si el objeto está truncado (0 o 1).
 - **difficult**: Indica si la detección del objeto es difícil (0 o 1).

- **bndbox**: Coordenadas de la caja delimitadora, incluyendo xmin, ymin, xmax, ymax.

Aquí tenemos un ejemplo de estructura Pascal VOC XML:

```
<annotation>
  <folder>vehicles</folder>
  <filename>ff9435ee-ba7e-4d32-93bb-d931b3d2aca7.jpg</filename>
  <path>E:\vehicles\ff9435ee-ba7e-4d32-93bb-d931b3d2aca7.jpg</path>
  <size>
    <width>800</width>
    <height>598</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>truck</name>
    <bndbox>
      <xmin>7</xmin>
      <ymin>119</ymin>
      <xmax>630</xmax>
      <ymax>468</ymax>
    </bndbox>
  </object>
  <object>
    <name>person</name>
    <bndbox>
      <xmin>40</xmin>
      <ymin>90</ymin>
      <xmax>100</xmax>
      <ymax>350</ymax>
    </bndbox>
  </object>
</annotation>
```

Para crear un archivo en este formato podemos hacerlo del siguiente modo con `pascal-voc-writer` (<https://github.com/AndrewCarterUK/pascal-voc-writer>):

```
!pip install pascal-voc-writer
# https://mlhive.com/2022/02/read-and-write-pascal-voc-xml-annotations-in-py
```

```
from pascal_voc_writer import Writer

# create pascal voc writer (image_path, width, height)
writer = Writer('path/to/img.jpg', 800, 598)

# add objects (class, xmin, ymin, xmax, ymax)
writer.addObject('truck', 1, 719, 630, 468)
writer.addObject('person', 40, 90, 100, 150)

# write to file
writer.save('path/to/img.xml')
```

YOLO

El formato de anotación YOLO (You Only Look Once) es ampliamente utilizado para entrenar modelos de detección de objetos.

Cada anotación está en un archivo de texto que corresponde a una imagen, y cada línea del archivo representa un objeto en la imagen. La estructura de cada línea es la siguiente:

```
# Formato de anotación YOLO Original (v1-v4)
<object-class> <x_center> <y_center> <width> <height>
```

Aquí se describe cada campo:

1. **object-class**: ID de la clase del objeto (entero, empezando desde 0).
2. **x_center**: Coordenada x del centro de la caja delimitadora, normalizada entre 0 y 1 (dividiendo la coordenada x por el ancho de la imagen).
3. **y_center**: Coordenada y del centro de la caja delimitadora, normalizada entre 0 y 1 (dividiendo la coordenada y por la altura de la imagen).
4. **width**: Ancho de la caja delimitadora, normalizado entre 0 y 1 (dividiendo el ancho de la caja por el ancho de la imagen).
5. **height**: Altura de la caja delimitadora, normalizada entre 0 y 1 (dividiendo la altura de la caja por la altura de la imagen).



Anotación YOLO Original (v1-v4):

```
<class_id> <x_center> <y_center> <width> <height>
```

Para una imagen con un objeto de clase 1, centrado en (0.5, 0.5) con un ancho de 0.2 y una altura de 0.3, la anotación sería:

```
1 0.5 0.5 0.2 0.3
```

Este formato es eficiente y sencillo, ideal para entrenar modelos YOLO que requieren procesar anotaciones rápidamente durante el entrenamiento:

- Valores normalizados entre 0 y 1
- Un objeto por línea
- Coordenadas son relativas al tamaño de la imagen

Anotación YOLO v5-v7

```
<class_id> <x_center> <y_center> <width> <height>
```

- Mantiene el mismo formato que v1-v3

- La diferencia principal está en la estructura de directorios y nombres de archivos

Anotación YOLO v8

```
<class_id> <x_center> <y_center> <width> <height> <confidence>
```

- Añade un valor opcional de confianza
- También normalizado entre 0 y 1

Herramientas de anotación

Los modelos de visión por computadora, dependen en gran medida de conjuntos de datos anotados, que sirven para entrenar modelos de aprendizaje automático. Las herramientas de anotación juegan un papel crucial en la creación de estos conjuntos de datos. Existen herramientas de anotación manual, semi-automáticas y automáticas. La anotación manual, aunque precisa, es laboriosa y consume mucho tiempo, mientras que las herramientas automáticas, impulsadas por algoritmos de inteligencia artificial, ofrecen una mayor eficiencia pero pueden ser menos precisas. Existen varias herramientas nativamente manuales, pero incorporan llamadas a modelos de detección o segmentación pre-entrenados, para hacer las anotaciones. De ese modo, una persona puede validar las anotaciones automáticas y corregirlas si es necesario.

La elección entre estos métodos depende del equilibrio deseado entre precisión y eficiencia.

Herramientas manuales o semi-automáticas.

Las herramientas manuales son aplicaciones con interfaz gráfica, que permiten interactuar con imágenes a través del uso del mouse principalmente, donde podemos anotarlas dibujando bounding boxes, polígonos, puntos, etc., según la tarea de anotación que estemos haciendo.

Algunas herramientas funcionan con interfaz web dentro del browser, y otras son aplicaciones de escritorio.

- **Labelme**

<https://github.com/labelmeai/labelme>

- **CVAT**

<https://github.com/cvat-ai/cvat>

- **Label Studio**

<https://github.com/HumanSignal/label-studio>

- **Anylabeling**

<https://github.com/vietanhdev/anylabeling>

- **X-AnyLabeling**

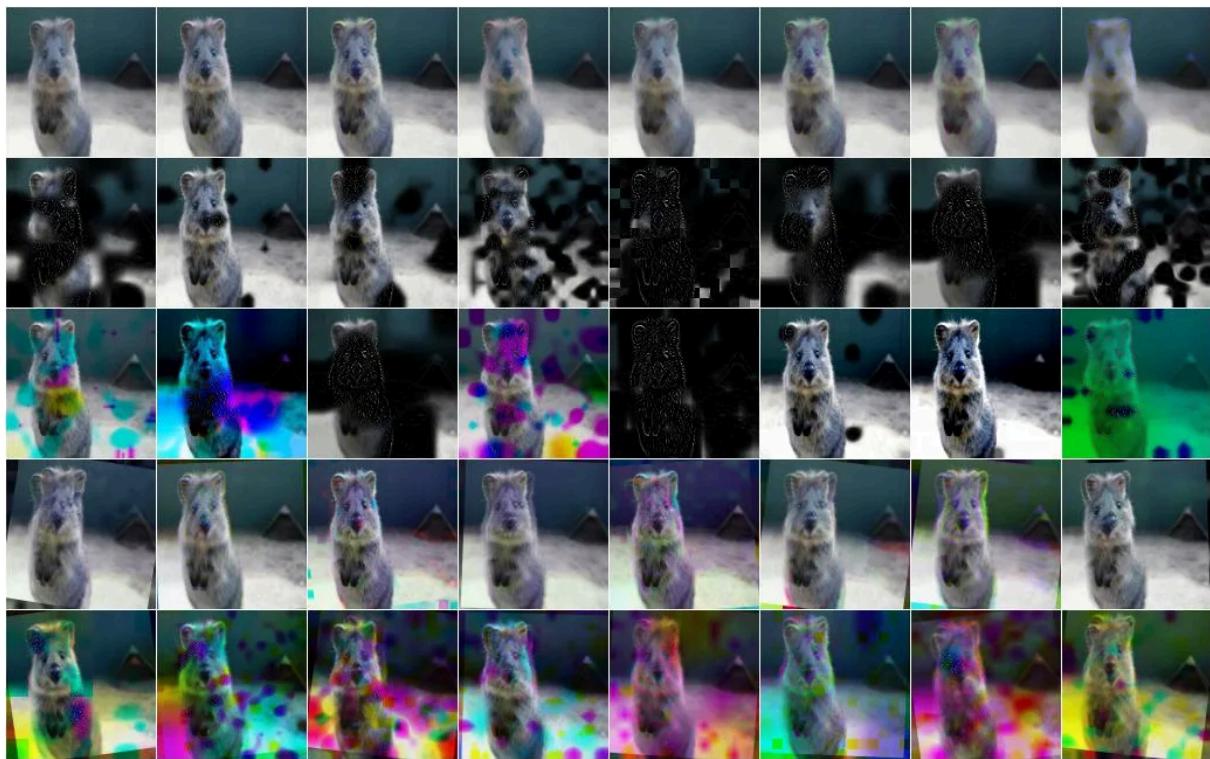
<https://github.com/CVHub520/X-AnyLabeling>

- **Autodistill (automáticas)**

<https://github.com/autodistill/autodistill>

2. Aumentación de datos (Data Augmentation)

El **data augmentation** o aumento de datos es una técnica utilizada en el campo del aprendizaje automático, especialmente en el aprendizaje profundo, para incrementar la cantidad y diversidad de los datos de entrenamiento sin necesidad de recolectar nuevos datos. Este proceso se realiza aplicando diversas transformaciones y modificaciones a los datos existentes, lo que permite crear nuevas variaciones de los mismos.

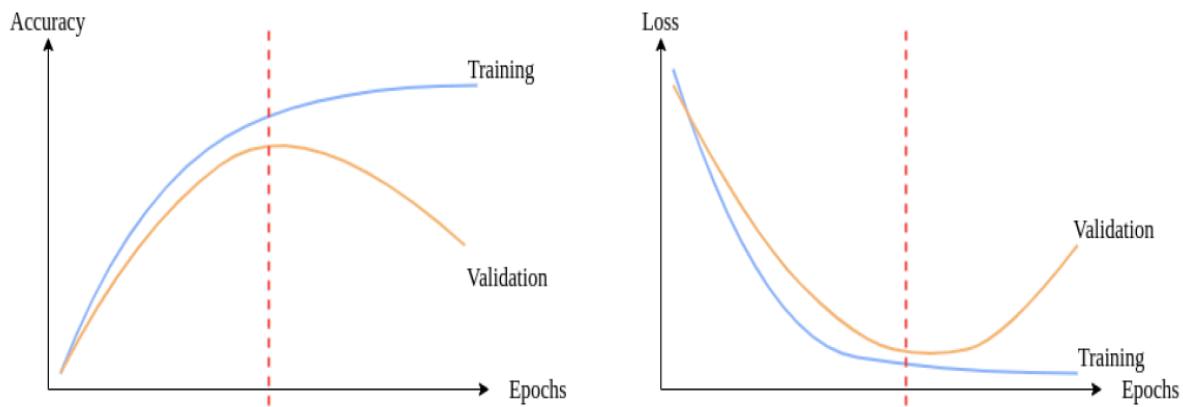


Aumentaciones conseguidas con la librería de python [imgaug](#)

Motivación del Data Augmentation

El data augmentation se originó como una solución a uno de los principales desafíos en el aprendizaje automático: la necesidad de grandes cantidades de datos para entrenar modelos efectivos. En el caso de las redes neuronales profundas, la cantidad de datos necesaria para lograr un buen rendimiento es especialmente alta. Recopilar estos datos puede ser costoso y a veces impracticable, especialmente en áreas como la medicina o la agricultura, donde obtener datos etiquetados de alta calidad puede ser complicado o de mucho trabajo manual. Uno de los motivos principales de la aumentación, es reducir el sobreajuste de los modelos, y lograr una mejor generalización, a partir de lograr mayor variedad en los datos.

Durante el entrenamiento de los modelos, podemos observar estos comportamientos que son indicativos de sobreajuste:



Problema de sobreajuste: En el lado izquierdo, se explica el sobreajuste en términos de precisión: después del punto de inflación (línea roja punteada), la precisión en el entrenamiento aumenta, pero la precisión en la validación disminuye. En el lado derecho, alternativamente en términos de pérdida: la pérdida en el entrenamiento disminuye, pero la pérdida en la validación aumenta después de la línea roja punteada ([Fuente](#)).

Importancia en el Computer Vision

En el campo de la visión, el data augmentation ha demostrado ser una técnica eficaz para mejorar el rendimiento de los modelos. Las transformaciones comunes incluyen operaciones geométricas (rotaciones, traslaciones, escalados), ajustes de color, adición de ruido, y técnicas más complejas como el uso de redes generativas adversariales (GANs) para generar nuevas imágenes a partir de las existentes.

Categorías de Data Augmentation

Hay diferentes maneras de categorizar la aumentación de datos según los métodos utilizados. A continuación se muestra un árbol de clasificación:

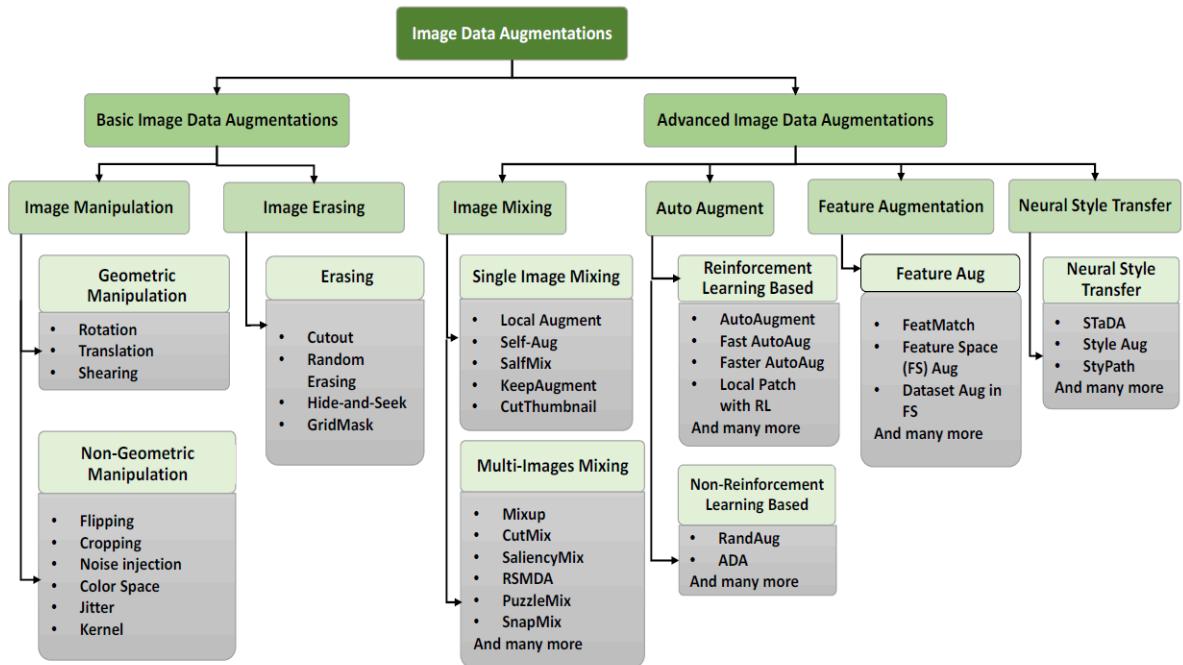


Image Data Augmentation Approaches: A Comprehensive Survey and Future directions

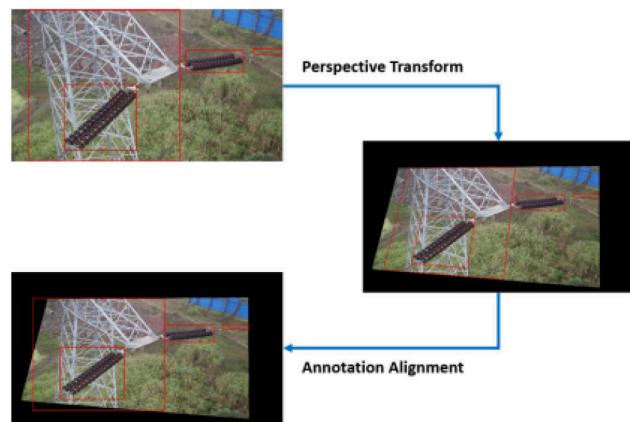
Los métodos básicos, involucran tareas de alteraciones de color, geométricas, máscaras, ruido, o blur. Los métodos avanzados, utilizan técnicas de redes profundas, GANs o RLs para lograr alteraciones de la imagen más complejas.



El data augmentation no solo aumenta el tamaño del conjunto de datos, sino que también mejora la capacidad del modelo para generalizar a datos no vistos, reduce el sobreajuste y mejora la robustez del modelo frente a variaciones y ruido en los datos de entrada. Es especialmente útil en aplicaciones como el reconocimiento de objetos, la clasificación de imágenes y la segmentación de imágenes.

Coherencia de las Anotaciones

Cuando se realiza aumentación de datos sobre imágenes anotadas previamente, es muy importante que las transformaciones aplicadas preserven la coherencia de las anotaciones originales. Por ejemplo, si se realiza un recorte o una rotación en una imagen de un dataset de detección de objetos, las coordenadas de los cuadros delimitadores (bounding boxes) deben ajustarse adecuadamente para reflejar la nueva posición y orientación del objeto.



De lo contrario, el modelo podría aprender información incorrecta, afectando negativamente su rendimiento.

Más información:

[Data augmentation: A comprehensive survey of modern approaches](#)

[A survey on Image Data Augmentation for Deep Learning](#)

Herramientas para realizar aumentación de datos

imgaug

<https://github.com/aleju/imgaug>

Albumentation

<https://github.com/albumentations-team/albumentations>

Demo: <https://demo.albumentations.ai/>

Augly

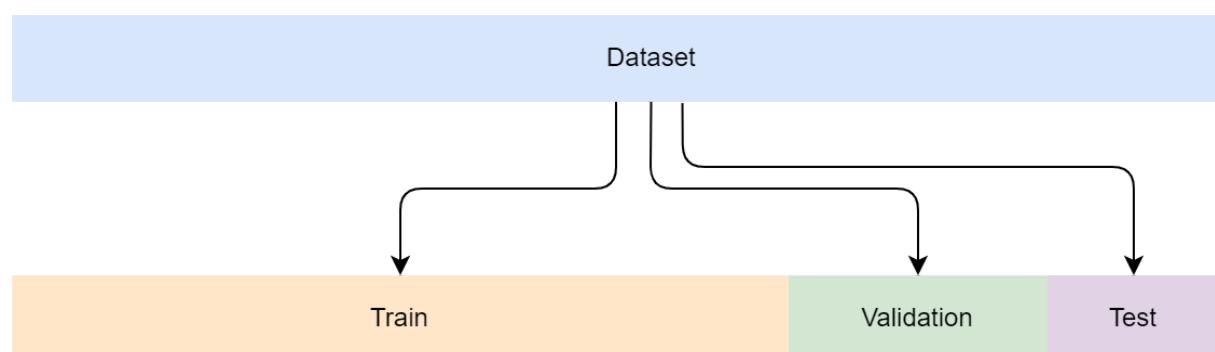
<https://github.com/facebookresearch/AugLy>

3. Entrenamiento

Para lograr modelos precisos y eficaces, es fundamental contar con un entrenamiento riguroso y un dataset de alta calidad. El proceso de entrenamiento permite que el modelo aprenda a reconocer patrones y características relevantes, mientras que un dataset bien etiquetado y representativo garantiza que el modelo generalice correctamente a datos nuevos. La combinación adecuada de ambos factores es esencial para el desarrollo de aplicaciones robustas y confiables en la visión por computadora.

Segmentación de datasets

En el campo de la visión por computadora, la segmentación de un conjunto de datos (dataset) en diferentes divisiones (splits), como entrenamiento (train), validación (validation) y prueba (test), es una práctica fundamental y ampliamente utilizada.



A continuación, se explica el propósito de esta segmentación y la justificación para utilizar cada división:

1. Entrenamiento (Train):

La división de entrenamiento es la parte más grande del conjunto de datos y se utiliza para entrenar el modelo de aprendizaje automático. Durante el proceso de entrenamiento, el modelo aprende a partir de los ejemplos proporcionados en esta división, ajustando sus parámetros internos para capturar los patrones y características relevantes de los datos. La razón principal para utilizar una división de entrenamiento es permitir que el modelo aprenda a generalizar a partir de una gran cantidad de ejemplos. Cuantos más datos de entrenamiento tenga el modelo, mejor podrá capturar la variabilidad y complejidad del problema, lo que conducirá a un mejor rendimiento en tareas futuras.

2. Validación (Validation):

La división de validación es una parte más pequeña del conjunto de datos que se separa del conjunto de entrenamiento. Esta división se utiliza durante el proceso de entrenamiento para evaluar el rendimiento del modelo en datos que no se han utilizado para el entrenamiento. El propósito principal de la división de validación es:

- a. Monitorear su progreso durante el entrenamiento. Generalmente podemos visualizar métricas que se calculan con esta división del dataset, luego de cada época de entrenamiento.
- b. Ajustar los hiperparámetros del modelo. Al evaluar el modelo con este split, se puede detectar si el modelo está empezando a sobreajustarse (overfitting) o subajustarse (underfitting) a los datos de entrenamiento. Esto permite ajustar los hiperparámetros, como la tasa de aprendizaje, la regularización o la arquitectura del modelo, para mejorar su rendimiento.

3. Prueba (Test):

La división de prueba es una parte del conjunto de datos que se mantiene completamente separada de los datos de entrenamiento y validación. Esta división se utiliza para evaluar el rendimiento final del modelo después de completar el entrenamiento y la validación. El propósito principal de la división de prueba es obtener una estimación imparcial del rendimiento del modelo en datos completamente nuevos que no se han utilizado en ninguna parte del proceso de entrenamiento o ajuste de hiperparámetros. Esto proporciona una evaluación más realista de cómo se comportará el modelo en el mundo real, donde se enfrentará a datos que nunca antes ha visto.

Mantener una división de prueba separada es muy importante para evitar el sobreajuste del modelo a los datos de entrenamiento y validación, lo que podría

dar una impresión engañosa de un buen rendimiento.

Datasets públicos

Es importante mencionar que no todos los conjuntos de datos (datasets) vienen separados en las divisiones de entrenamiento, validación y prueba. En algunos casos, el conjunto de datos está preparado de esa manera, pero en otros, es responsabilidad del investigador o desarrollador realizar esta separación, conocida como split.

- **Datasets con divisiones predefinidas:** Algunos conjuntos de datos populares y ampliamente utilizados en el campo de la visión por computadora, como ImageNet, COCO, PASCAL VOC, etc., ya vienen con divisiones predefinidas de entrenamiento, validación y prueba. Los creadores del conjunto de datos realizaron esta separación cuidadosa para facilitar el entrenamiento y la evaluación de modelos de manera consistente y comparable.
- **Datasets sin divisiones predefinidas:** Sin embargo, en muchos casos, el conjunto de datos no viene con estas divisiones predefinidas. Esto puede ocurrir cuando se trabaja con datos específicos de un dominio o problema particular, o cuando se recopilan datos personalizados. En estas situaciones, es responsabilidad del investigador o desarrollador realizar la separación del conjunto de datos en las divisiones apropiadas.

El proceso de separación del conjunto de datos en entrenamiento, validación y prueba, implica dividir aleatoriamente los datos en estas tres divisiones manteniendo una distribución representativa de las clases o características en cada una de ellas. Es una práctica común utilizar una proporción aproximada de 60-80% de los datos para entrenamiento, 10-20% para validación y el resto (10-20%) para la división de prueba.

Es importante realizar esta separación de manera cuidadosa y aleatoria para evitar sesgos y garantizar que las divisiones sean representativas del conjunto de datos completo. Además, es importante mantener la integridad de las divisiones y no mezclarlas durante el proceso de entrenamiento y evaluación del modelo.



Data Augmentation: Las aumentaciones de datos generalmente se aplican sobre la división de entrenamiento (**train split**) del conjunto de datos. La razón principal para hacer esto es aumentar la cantidad y variedad de los datos de entrenamiento, lo que puede mejorar el rendimiento y la capacidad de generalización del modelo. Esto puede incluir transformaciones como rotaciones, recortes, escalado, deformaciones, adición de ruido, entre otras. El objetivo es crear nuevas versiones sintéticas de las imágenes originales, ampliando así el conjunto de entrenamiento con más variaciones. Al entrenar el modelo con estos datos aumentados, se expone a una mayor diversidad de ejemplos, lo que puede ayudar a mejorar su capacidad para reconocer patrones y características en diferentes condiciones, como cambios en la orientación, iluminación, escala, etc. Esto puede reducir el riesgo de sobreajuste (overfitting) y mejorar la generalización del modelo a nuevos datos.

Proceso de entrenamiento

Generalmente, los modelos de aprendizaje profundo se entranen en grandes conjuntos de datos con miles o millones de muestras. Usar todo el conjunto de datos para cada actualización de gradiente sería costoso en términos de tiempo y recursos computacionales. Para solucionar esto, se utilizan lotes, porciones del conjunto de datos, para realizar actualizaciones de gradiente durante el entrenamiento.

Batch Size (Tamaño de Lote)

El tamaño de lote se refiere al número de muestras de entrenamiento en el lote. Por ejemplo, `batch_size=128` significa que hay 128 muestras en cada lote.

No debemos confundir el tamaño del lote con el número de lotes. El número de lotes se calcula de la siguiente manera:



No. of batches = (Size of the entire dataset / batch size) + 1

Está muy claro que el número de lotes depende de dos factores: el tamaño del conjunto de datos completo y el tamaño del lote. Además de eso, agregamos 1 para compensar cualquier parte fraccionaria si fuera el caso. Ejemplo:

Para un conjunto de datos de 60.000 instancias y un tamaño de lote de 128:

$$60.000 / 128 = 468.75$$

Sumamos 1 para compensar la parte fraccionaria, resultando en 469 lotes. En este escenario, 468 lotes tienen 128 muestras y el lote restante tiene 96.

$$\text{No. of batches} = 468 + 1 = 469$$

Pasos de Entrenamiento e Iteraciones

Durante el entrenamiento, los lotes de datos se pasan a través de la red neuronal y el error calculado por la función de pérdida se propaga hacia atrás (backpropagation) a través de la red para actualizar los parámetros, de modo que la red pueda hacer mejores predicciones en los siguientes pasos.

Discutamos cómo sucede esto detrás de escena. Imaginemos que los datos están listos para entrenar el modelo. Tenemos 60.000 instancias de entrenamiento y el tamaño del lote es 128.

Antes de comenzar a extraer lotes del conjunto de datos, el algoritmo generalmente baraja aleatoriamente los datos de entrenamiento (si así lo indican los parámetros de configuración).

A continuación, el algoritmo comienza a extraer lotes del conjunto de datos. Toma las primeras 128 instancias (primer lote) del conjunto de datos, entrena el modelo, calcula el error promedio y actualiza los parámetros una vez (realiza una actualización de gradiente). Esto completa un paso de entrenamiento (también llamado iteración).



Más precisamente, un paso de entrenamiento (iteración) es una actualización de gradiente.

Luego, el algoritmo toma las siguientes 128 instancias (segundo lote) del conjunto de datos, entrena el modelo, calcula el error promedio y actualiza los parámetros una vez (realiza otra actualización de gradiente). Esto completa otro nuevo paso de entrenamiento.

El algoritmo continúa con este procedimiento hasta que todos los lotes se hayan extraído del conjunto de datos de entrenamiento (eso es 469 veces según nuestro ejemplo). Eso concluye una época en el entrenamiento. En una época, se muestra todo el conjunto de datos al modelo.



Más específicamente, las épocas se refieren al número de veces que el modelo ve el conjunto de datos completo.

El número de actualizaciones de gradiente realizadas en una época es igual al número de pasos de entrenamiento (iteraciones) en esa época. Está muy claro que las épocas y las iteraciones son dos cosas diferentes.

Ahora, podemos escribir la siguiente afirmación considerando una época de entrenamiento:



No. of training steps = No. of batches = No. of gradient updates

Cuando el número de lotes es 469, como en nuestro ejemplo, hay 469 pasos de entrenamiento o 469 actualizaciones de gradiente en una época de entrenamiento.

Luego, el algoritmo está listo para la segunda época. Nuevamente, baraja aleatoriamente los datos de entrenamiento al inicio de esta época. Aquí también, el algoritmo sigue el mismo procedimiento que en la primera época.

Cuando el algoritmo completa las N épocas previstas al iniciar el proceso (todo el conjunto de datos se ha mostrado al modelo N veces), concluye todo el proceso de entrenamiento.

El número total de actualizaciones de gradiente o el número total de pasos de entrenamiento durante todo el proceso de entrenamiento se puede calcular de la siguiente manera.



No. of ALL gradient updates = No. of batches x No. of epochs

Más información:

[All You Need to Know about Batch Size, Epochs and Training Steps in a Neural Network](#)

4. Evaluación y métricas

En el ámbito del aprendizaje automático y la inteligencia artificial, entrenar modelos es solo una parte del proceso para alcanzar soluciones precisas y eficaces. La evaluación de estos modelos es crucial para entender su desempeño real en tareas específicas. Sin una evaluación adecuada, los modelos pueden presentar resultados engañosos, llevando a decisiones incorrectas. Herramientas como la

precisión promedio (AP) y la precisión promedio media (mAP) permiten medir y comparar la efectividad de los modelos de detección de objetos, garantizando su utilidad y robustez en aplicaciones prácticas. Evaluar de manera rigurosa asegura que los modelos no solo funcionen en entornos de entrenamiento, sino que también sean fiables en el mundo real, donde se enfrentan a datos no vistos y condiciones variables.

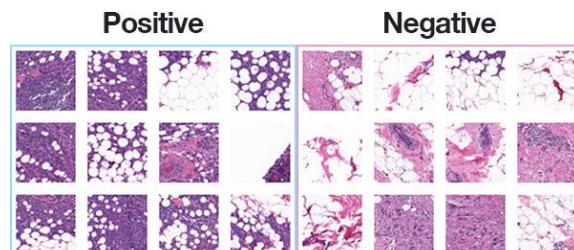
Métricas en un clasificador

Antes de analizar la forma de evaluar modelos de detección de objetos, vamos a ver primero cómo podemos evaluar un modelo más básico y de clasificación. La evaluación de un modelo de clasificación se realiza utilizando los valores de Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TN) y Falsos Negativos (FN), y permiten una evaluación completa del rendimiento del modelo desde diferentes perspectivas.



Salidas de Modelos de Clasificación

Muchos modelos de clasificación, especialmente aquellos basados en aprendizaje profundo o regresión logística, no emiten directamente una etiqueta de clase (ej: "cáncer" o "no cáncer"). En su lugar, suelen generar una **puntuación de probabilidad** o un valor numérico que indica la confianza



del modelo en que una instancia pertenece a una clase particular.

Salida de Probabilidad: Para un problema de clasificación binaria (como la detección de cáncer), el modelo podría generar un número entre 0 y 1.

- Un valor cercano a 1 sugiere una alta probabilidad de que la instancia sea positiva (ej. la imagen contiene células cancerosas).
- Un valor cercano a 0 sugiere una alta probabilidad de que la instancia sea negativa (ej. la imagen no contiene células cancerosas).

Por ejemplo, si alimentamos una imagen médica a nuestro clasificador de cáncer, podría devolver:

- Imagen A: **0.92** (alta probabilidad de cáncer)
- Imagen B: **0.15** (baja probabilidad de cáncer)
- Imagen C: **0.65** (probabilidad intermedia)

El Papel del Umbral (Threshold)

Para convertir estas probabilidades en una decisión binaria (positivo/negativo), necesitamos establecer un **umbral de decisión** (también conocido como *threshold* o punto de corte).

- **Definición:** El umbral es un valor predefinido. Si la probabilidad generada por el modelo para una instancia es mayor o igual que el umbral, la instancia se clasifica como positiva. Si es menor, se clasifica como negativa.
- **Umbral Común (pero no siempre óptimo):** Un umbral comúnmente utilizado por defecto es **0.5**.
 - Si probabilidad $\geq 0.5 \rightarrow$ Clase Positiva
 - Si probabilidad $< 0.5 \rightarrow$ Clase Negativa



Las métricas típicas de un modelo de clasificación que veremos a continuación, se calculan a partir de los TP, TN, FP y FN, es decir que la selección del umbral va a determinar los cálculos de las métricas

1. Sensibilidad (Recall o Tasa de Verdaderos Positivos)

- **Descripción:** La sensibilidad mide la proporción de los casos positivos reales que fueron correctamente identificados por el modelo. En otras palabras, de todos los que realmente eran positivos, ¿Cuántos detectó el modelo? Es una métrica crucial cuando el costo de los falsos negativos es alto (por ejemplo, en diagnósticos médicos, no detectar una enfermedad).

- **Fórmula:**

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

2. Especificidad (Specificity o Tasa de Verdaderos Negativos)

- **Descripción:** La especificidad mide la proporción de los casos negativos reales que fueron correctamente identificados por el modelo. Es decir, de todos los que realmente eran negativos, ¿Cuántos identificó correctamente el modelo como negativos? Es importante cuando el costo de los falsos positivos es alto (por ejemplo, clasificar a una persona sana como enferma, lo que podría llevar a tratamientos innecesarios).

- **Fórmula:**

$$\text{Especificidad} = \frac{TN}{TN + FP}$$

3. Precisión (Precision)

- **Descripción:** La precisión mide la proporción de las identificaciones positivas que fueron realmente correctas. En otras palabras, de todos los casos que el modelo predijo como positivos, ¿Cuántos eran realmente positivos? Es una métrica importante cuando se quiere estar muy seguro de las predicciones positivas, minimizando los falsos positivos.

- **Fórmula:**

$$\text{Precisión} = \frac{TP}{TP + FP}$$

4. Exactitud (Accuracy)

- **Descripción:** La exactitud mide la proporción de predicciones correctas (tanto positivas como negativas) entre el número total de casos evaluados. Es la métrica más intuitiva, pero puede ser engañosa en conjuntos de datos desbalanceados (cuando una clase es mucho más frecuente que la otra).

- **Fórmula:**

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

5. Puntaje F1 (F1 Score)

- **Descripción:** El puntaje F1 es la media armónica de la Precisión y la Sensibilidad. Proporciona una medida única que equilibra ambas métricas. Es muy útil cuando hay un desequilibrio en las clases o cuando tanto los falsos positivos como los falsos negativos son importantes. Un puntaje F1 alto indica que el modelo tiene tanto buena precisión como buena sensibilidad.
- **Fórmula:**

$$\text{Puntaje F1} = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

Impacto del umbral en las métricas de evaluación

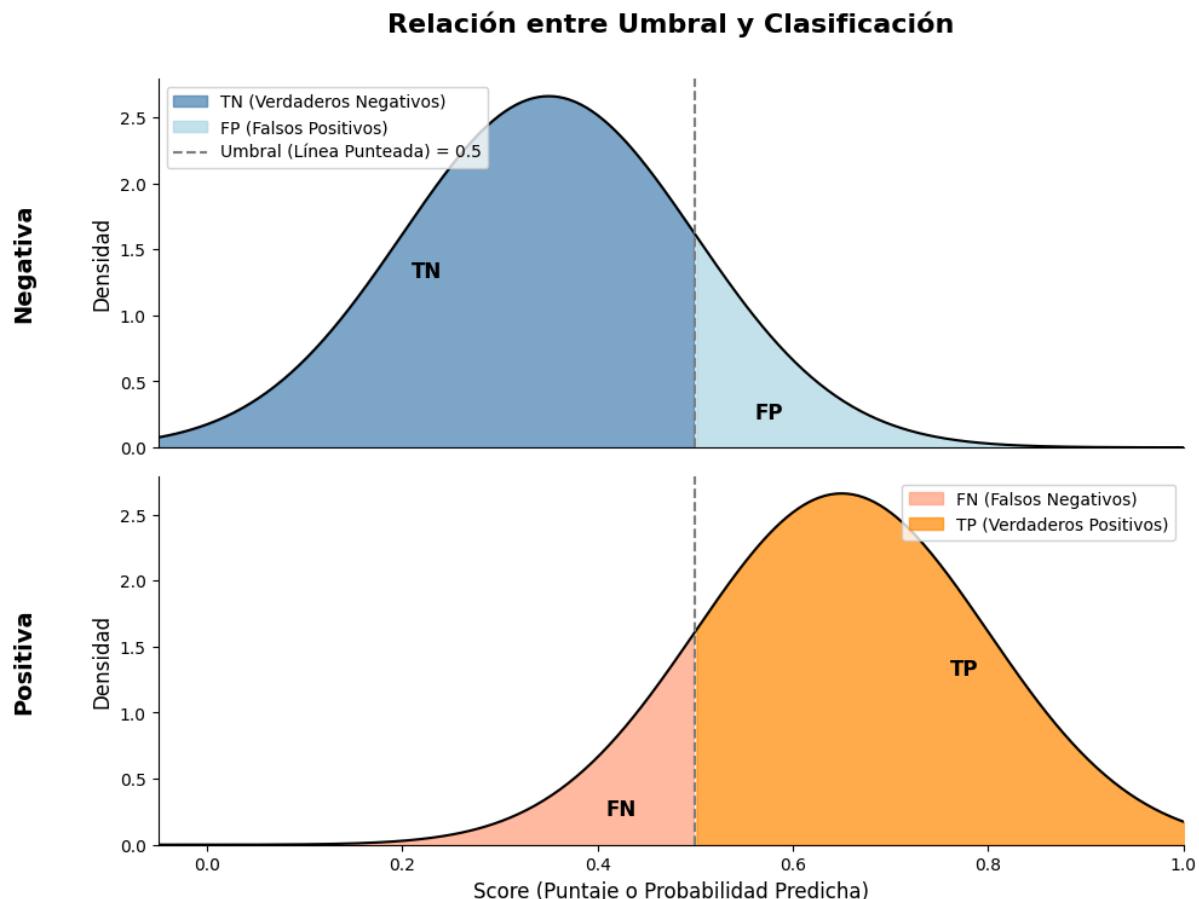
Como hemos mencionado, la elección del umbral tiene un impacto directo y significativo en las métricas de evaluación que discutimos anteriormente (Sensibilidad, Especificidad, Precisión, etc.). Veamos un ejemplo:

- **Umbral Bajo (ej. 0.2):**
 - **Mayor Sensibilidad (Recall):** Se identificarán más casos positivos reales. El modelo será más "sensible" a la detección de la enfermedad, incluso si la probabilidad es relativamente baja. Esto significa que se reducirán los Falsos Negativos (FN).
 - **Menor Especificidad:** Es probable que aumenten los Falsos Positivos (FP). Casos que en realidad son negativos podrían ser clasificados como positivos porque sus probabilidades, aunque bajas, superan el umbral.
 - **Mayor Precisión (Potencialmente):** Si la mayoría de los casos con baja probabilidad son realmente positivos, la precisión podría aumentar. Sin embargo, es más común que baje debido al aumento de FPs.
- **Umbral Alto (ej. 0.8):**
 - **Menor Sensibilidad (Recall):** Se identificarán menos casos positivos reales, ya que solo aquellos con una probabilidad muy alta serán clasificados como positivos. Esto significa que aumentarán los Falsos Negativos (FN).
 - **Mayor Especificidad:** Se reducirán los Falsos Positivos (FP), ya que el modelo será más "estricto" al clasificar algo como positivo.

- **Mayor Precisión:** Es más probable que las predicciones positivas sean correctas, ya que se requiere una alta confianza del modelo.



Existe un compromiso (trade-off) inherente entre la Sensibilidad y la Especificidad al ajustar el umbral. Aumentar una generalmente disminuye la otra.



Imaginemos nuestro modelo que analiza imágenes de tejido para detectar signos de cáncer. El modelo produce una probabilidad de que la imagen muestre malignidad.

- **Escenario 1: Umbral Bajo (ej. 0.3)**
 - **Objetivo:** Maximizar la detección de todos los posibles casos de cáncer (alta sensibilidad). No queremos pasar por alto ningún caso potencial.
 - **Consecuencia:**
 - **Ventaja:** Se identificarán muchos pacientes que realmente tienen cáncer. Se minimizan los Falsos Negativos (pacientes con cáncer no

detectados).

- **Desventaja:** Es probable que se clasifiquen como "potencialmente cancerosas" muchas imágenes de tejido sano (aumento de Falsos Positivos). Esto podría llevar a más biopsias innecesarias, pruebas adicionales, y ansiedad para los pacientes.
 - **Métricas afectadas:** Alta Sensibilidad, baja Especificidad.
- **Escenario 2: Umbral Alto (ej. 0.9)**
 - **Objetivo:** Estar muy seguro antes de declarar una imagen como cancerosa (alta precisión/especificidad). Queremos evitar alarmas innecesarias.
 - **Consecuencia:**
 - **Ventaja:** Cuando el modelo dice "cáncer", es muy probable que sea correcto. Se minimizan los Falsos Positivos (pacientes sanos diagnosticados erróneamente).
 - **Desventaja:** Es posible que algunos casos tempranos o sutiles de cáncer, donde la probabilidad del modelo no es extremadamente alta, no sean detectados (aumento de Falsos Negativos). Esto podría retrasar el tratamiento.
 - **Métricas afectadas:** Baja Sensibilidad, alta Especificidad, potencialmente alta Precisión.

¿Qué umbral elegir en la detección de cáncer?

Esta es una decisión crítica que a menudo involucra a expertos médicos y considera las consecuencias de los diferentes tipos de errores:

- **Costo de un Falso Negativo (FN):** Muy alto. No detectar un cáncer puede tener consecuencias fatales.
- **Costo de un Falso Positivo (FP):** Moderado a alto. Implica pruebas adicionales, costos económicos y estrés para el paciente, pero generalmente no es fatal.

Dado el alto costo de los Falsos Negativos en la detección de cáncer, a menudo se prefiere un **umbral más bajo** para maximizar la Sensibilidad, incluso a expensas de una menor Especificidad. El objetivo principal es no pasar por alto ningún caso. Los casos marcados como positivos (incluso algunos Falsos Positivos) pueden luego ser revisados por patólogos humanos para una confirmación definitiva.

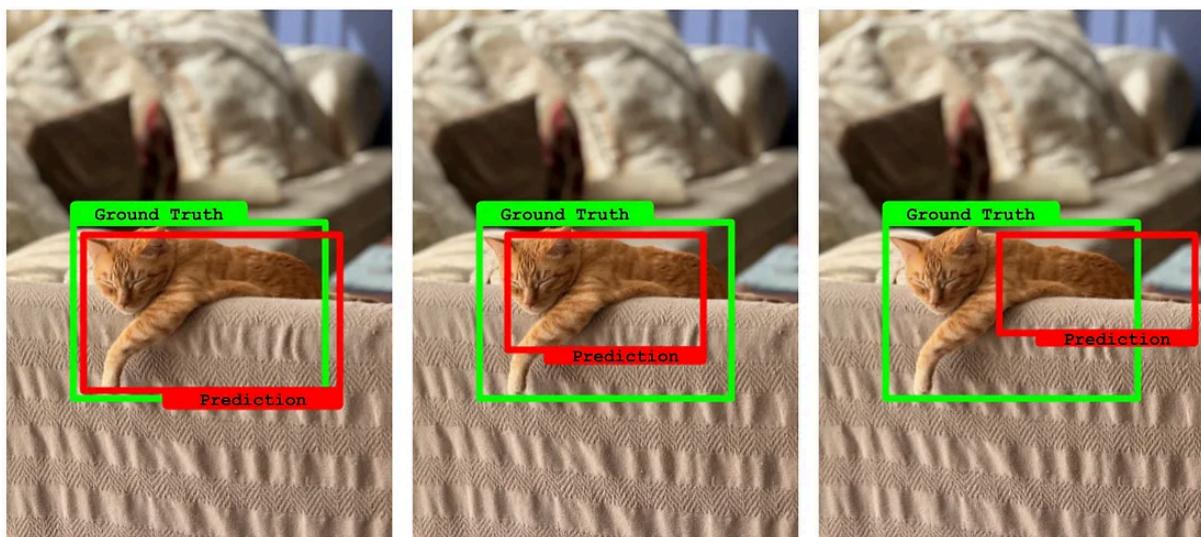
Evaluación en un Detector de Objetos

Cuando medimos la calidad de un detector de objetos, principalmente queremos evaluar dos criterios:

1. El modelo predijo la clase correcta para el objeto.
2. El cuadro delimitador predicho está lo suficientemente cerca de la verdad de base.

Como podemos ver, al combinar ambos criterios, las cosas empiezan a volverse confusas. Comparado con otras tareas de aprendizaje automático, como la clasificación, no hay una definición clara de "predicción correcta". Esto es especialmente cierto con el criterio del cuadro delimitador.

Tomemos, por ejemplo, las imágenes a continuación. ¿Cuáles dirías que son correctas y cuáles incorrectas?



Ejemplos de detecciones de objetos con diferentes calidades.

No te preocupes si te cuesta tomar una decisión, ¡es un problema difícil! Y se vuelve aún más complicado. También puede ser interesante evaluar el detector con objetos de diferentes tamaños. Un detector puede tener dificultades con objetos pequeños, pero sobresalir con los grandes. Otro detector puede no ser tan "bueno" en general, pero tener mejor rendimiento en objetos pequeños. ¿Y qué hay de la cantidad de objetos que un modelo puede detectar simultáneamente? No es tan simple como podemos ver. Eso es precisamente lo que las métricas de evaluación de COCO buscan evaluar. Estas proporcionan un estándar para medir el rendimiento de un detector de objetos bajo diferentes escenarios bien establecidos.

La evaluación de COCO va más allá de la detección de objetos y proporciona métricas para segmentación, detección de puntos clave, etc... pero por ahora nos concentraremos en métricas de detección de objetos.

IoU: Intersección sobre Unión

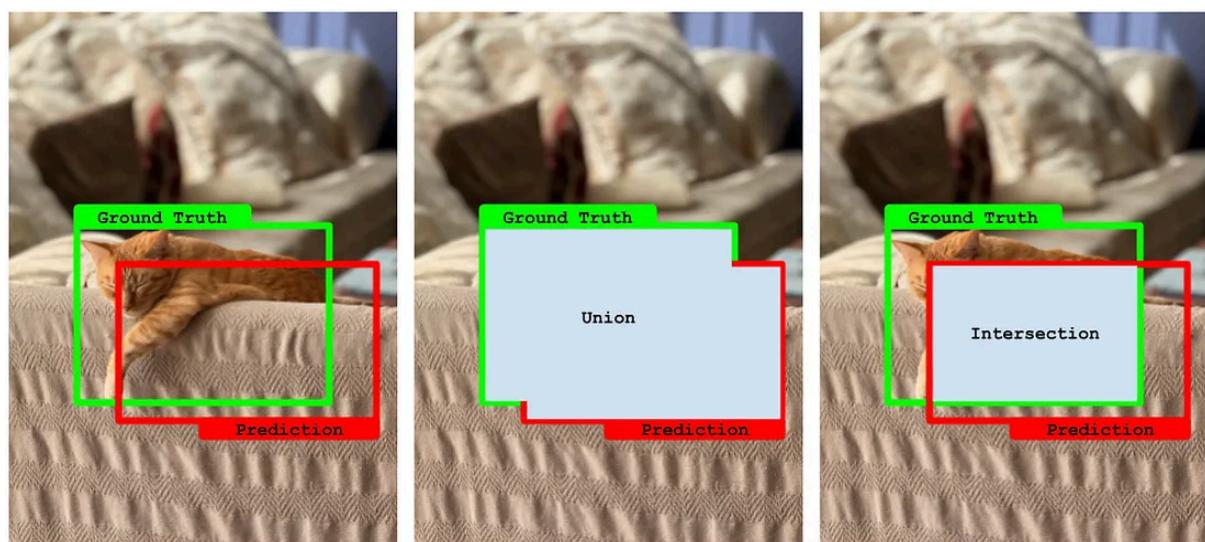
La Intersección sobre Unión es una medida de qué tan bien se alinean dos cuadros delimitadores. Se usa típicamente para medir la calidad de un cuadro predicho en comparación con la verdad de base (ground truth).

El IoU, como su nombre sugiere, se define como la intersección de los dos cuadros, dividida por la unión de ambos:

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

The diagram shows three pairs of overlapping blue rectangles. The first pair, labeled 'Poor', has a small overlap. The second pair, labeled 'Good', has a moderate overlap. The third pair, labeled 'Excellent', has a large overlap where the prediction rectangle is almost entirely contained within the ground truth rectangle.

En la práctica, podemos entender la intersección y la unión como:



Izquierda: Predicción Original. Centro: Unión. Derecha: Intersección.

Analicemos por un momento la ecuación. Hay algunos puntos que vale la pena mencionar:

- La unión siempre será mayor (o igual) que la intersección.
- Si los cuadros no se superponen, la intersección será cero, haciendo que el IoU=0.
- Si los cuadros se superponen perfectamente, la intersección coincidirá con la unión, haciendo que el IoU=1.
- El IoU siempre será un valor entre 0 y 1, inclusive.
- Cuanto mayor sea el IoU, mejor.

A veces podemos escuchar sobre el IoU mencionado con un nombre más elegante: Índice de Jaccard o Similitud de Jaccard. En el contexto de la detección de objetos, son lo mismo. El Índice de Jaccard es una forma matemática más general de comparar la similitud entre dos conjuntos finitos.

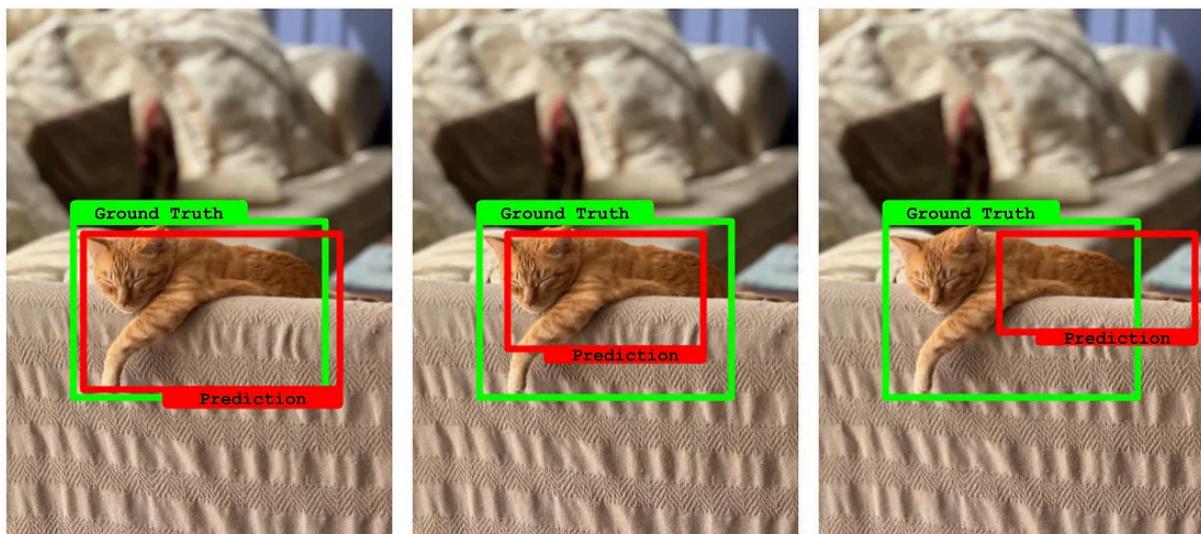
En Python, el IoU se podría calcular así:

```
def iou(bbox_a, bbox_b):  
    ax1, ay1, ax2, ay2 = bbox_a  
    bx1, by1, bx2, by2 = bbox_b  
  
    # Compute the coordinates of the intersection  
    ix1 = max(ax1, bx1)  
    iy1 = max(ay1, by1)  
    ix2 = min(ax2, bx2)  
    iy2 = min(ay2, by2)  
  
    # Compute the area of intersection rectangle  
    intersection = max(0, ix2 - ix1) * max(0, iy2 - iy1)  
  
    # Compute the area of both bounding boxes  
    box1_area = (ax2 - ax1) * (ay2 - ay1)  
    box2_area = (bx2 - bx1) * (by2 - by1)  
  
    # Finally compute the union of the areas  
    union = box1_area + box2_area - intersection  
  
    return intersection / union
```

Entonces ahora tenemos un número que describe qué tan bueno fue un cuadro delimitador predicho en comparación con la verdad de base. Pero, ¿cuánto puede empeorar el IoU antes de que descartemos la predicción?

IoU como Umbral de Detección

Observemos nuevamente los ejemplos de predicciones de gatos. Está claro que el de la derecha está mal, mientras que el de la izquierda es aceptable. ¿Cómo decidió nuestro cerebro esto tan rápidamente? ¿Y qué pasa con el del medio? Si estamos comparando el rendimiento de modelos, no podemos dejar esta decisión de manera subjetiva.



El IoU puede servir como un umbral para aceptar o no una predicción.

Verás este umbral especificado como IoU@0.5, lo que simplemente significa: "solo se consideraron correctos los cuadros delimitadores con un IoU mayor o igual a 0.5 (o 50%) con respecto a la verdad de base."

En la literatura es típico encontrar los siguientes umbrales de IoU:

- IoU@0.5
- IoU@0.75
- IoU@0.95
- IoU@[0.5:0.05:0.95]

Los tres primeros deberían ser claros a estas alturas. El último, aunque confuso, es fácil de entender. Se refiere a múltiples umbrales y simplemente significa: "todos

los umbrales de IoU desde 0.5 hasta 0.95, usando un paso de 0.05. Si lo expandes, verás que esta notación abarca 10 umbrales diferentes de IoU:

$\text{IoU}@[0.5:0.05:0.95] = \text{IoU}@0.5, \text{IoU}@0.55, \text{IoU}@0.6, \dots, \text{IoU}@0.85, \text{IoU}@0.9, \text{IoU}@0.95$

Esta expresión se ha vuelto muy popular gracias a COCO. A veces los autores abusan de la notación y omiten el paso, simplemente escribiendo $\text{IoU}@[0.5:0.95]$. Aunque confuso, es casi seguro que se refieren a los diez pasos de IoU descritos anteriormente. Veremos cómo aplicar múltiples umbrales más adelante.



Al evaluar un detector de objetos, si el autor no especifica un umbral de IoU, casi siempre se sobreentiende $\text{IoU}=0.5$.

Verdaderos Positivos (TP), Falsos Positivos (FP), Falsos Negativos (FN) y Verdaderos Negativos (TN)

Los siguientes conceptos a entender son TP (True Positive), FP (False Positive), TN (True Negative) y FN (False Negative). Estos términos se toman prestados de las tareas de clasificación binaria. La siguiente tabla, para un **clasificador hipotético de manzanas**, los resume:

Objeto	Predicción	Categoría	Explicación
🍎	✓	TP (True Positive)	El objeto fue clasificado correctamente como una manzana.
🍌	✓	FP (False Positive)	El objeto fue clasificado incorrectamente como una manzana.
🍎	✗	FN (False Negative)	El objeto fue clasificado incorrectamente "no es una manzana"
🍌	✗	TN (True Negative)	El objeto fue clasificado correctamente "no es una manzana"



Un buen clasificador tiene muchas predicciones Verdaderas Positivas y Verdaderas Negativas (TP/TN), mientras minimiza las Falsas Positivas y Falsas Negativas (FP/FN).

Clasificadores Multiclas

El concepto presentado anteriormente no encaja del todo si tenemos un clasificador multiclase. El clasificador binario responde "¿es este objeto de esta clase?", mientras que el clasificador multiclase responde "¿a cuál de estas clases pertenece este objeto?". Afortunadamente, podemos extrapolarlo usando un enfoque de Uno-contra-Todos (One-vs-All) o Uno-contra-el-Resto (One-vs-Rest). La idea es simple: evaluamos cada clase individualmente y la tratamos como un clasificador binario. Luego, se cumple lo siguiente:

- Positivo: la clase en cuestión fue predicha correctamente para el objeto.
- Negativo: cualquier otra clase fue predicha para el objeto.

Aquí está la misma tabla resumen vista desde la perspectiva de la clase .

Objeto	Predicción	Categoría	Explicación
		TP	El objeto fue clasificado correctamente como una manzana.
		FP	El objeto fue clasificado incorrectamente como una manzana.
		FN	El objeto fue clasificado incorrectamente como una banana (no manzana).
		FN	El objeto fue clasificado incorrectamente como un mango (no manzana).

No es común usar el calificativo de Verdaderos Negativos (TN) cuando se habla de modelos multiclase, pero sería algo como: "las muestras que fueron correctamente identificadas como pertenecientes a otras clases".

Ahora hagamos un ejercicio similar desde la perspectiva de banana.

Objeto	Predicción	Categoría	Explicación
		TP	El objeto fue clasificado correctamente como una banana.
		FN	El objeto fue clasificado incorrectamente como una manzana (no banana).
		FP	El objeto fue clasificado incorrectamente como una banana.

¿Notaste lo que pasó? La combinación (, ) es un FP desde la perspectiva de la clase manzana, pero un FN desde la perspectiva de la clase banana. Una situación similar ocurre con la combinación (, ); es un FN desde la perspectiva de la

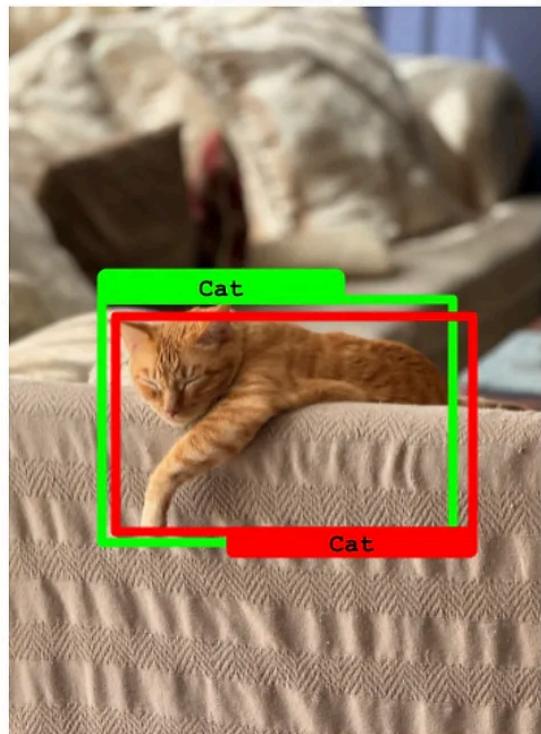
clase manzana, pero un FP desde la perspectiva de la clase banana. Este solapamiento es esperado y típico de un modelo multiclas.

Métricas en la detección de objetos

La pregunta restante es: ¿Cómo aplico los conceptos anteriores a un detector de objetos? Los detectores son típicamente clasificadores multiclas, pero también tienen un factor de localización de objetos. Afortunadamente, ya tenemos la métrica IoU para eso. Supongamos IoU=50%:

Verdadero Positivo

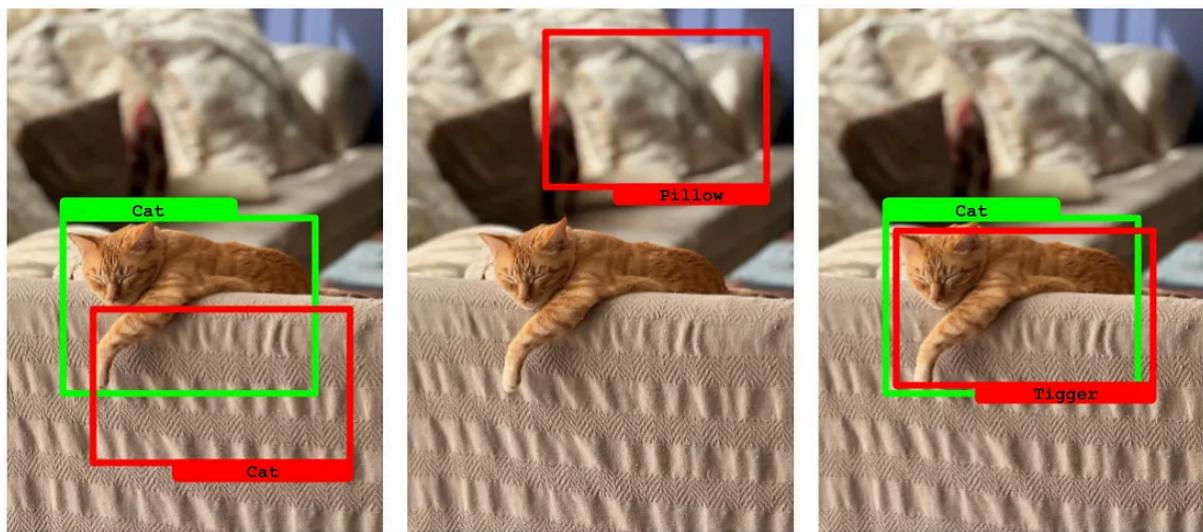
- El cuadro delimitador predicho tiene un IoU por encima del 50% con respecto a la verdad de base, y
- La clase predicha coincide con la verdad de base.



Ejemplo de un verdadero positivo en la detección de objetos.

Falso Positivo

- El cuadro delimitador predicho tiene un IoU por debajo del 50% con respecto a la verdad de base o
- Detectamos un objeto y no hay una verdad de base (ground truth) asociada o,
- La clase predicha no coincide con la verdad de base (ground truth).



Ejemplos de falsos positivos en la detección de objetos. Izquierda: IoU por debajo del 50%.
Centro: Falta de verdad de base. Derecha: Clase incorrecta (*).



En nuestro caso, de detección de objetos con múltiples clases, si tenemos un "ground truth" de un gato y el modelo predice un tigre con un IoU (Intersección sobre Unión) mayor a 0.5, así es como se computa para la matriz de confusión:

1. **Falso Positivo (FP)**: Aunque la predicción coincide espacialmente con el objeto "ground truth" (el gato), la clase es incorrecta (predijo un tigre en lugar de un gato). Esto se considera un error en la clasificación de la clase. Por tanto, cada instancia incorrecta detectada donde no debería haber un tigre (pero el modelo dijo que había uno) se cuenta como un "Falso Positivo".
2. **Falso Negativo (FN)**: También necesitamos contar la instancia de gato (la clase real) que no fue correctamente identificada como tal. Dado que el modelo no identificó correctamente la clase gato (aunque sí detectó un objeto tigre en la posición correcta), esto cuenta como un "Falso Negativo" para la clase gato.

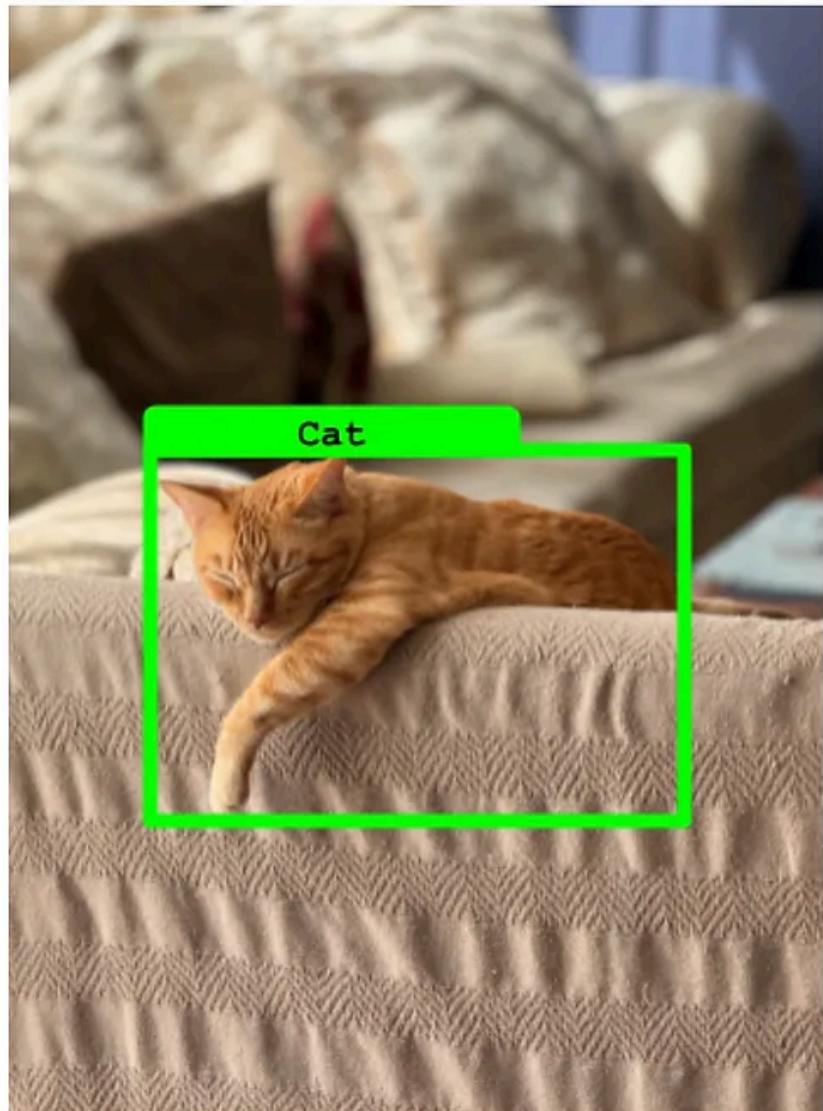
En resumen, para este caso particular:

- Tenemos un FP porque el modelo identificó incorrectamente un tigre donde no lo hay.
- Tenemos un FN porque el modelo no identificó un gato donde sí lo hay.

Ambos, FP y FN, deberían ser registrados en la matriz de confusión en sus respectivas posiciones: FP bajo la clase "tigre" y FN bajo la clase "gato". Esto ayuda a reflejar tanto los errores de localización como los de clasificación de clases.

Falso Negativo

Cada verdad de base que no tiene una predicción coincidente.

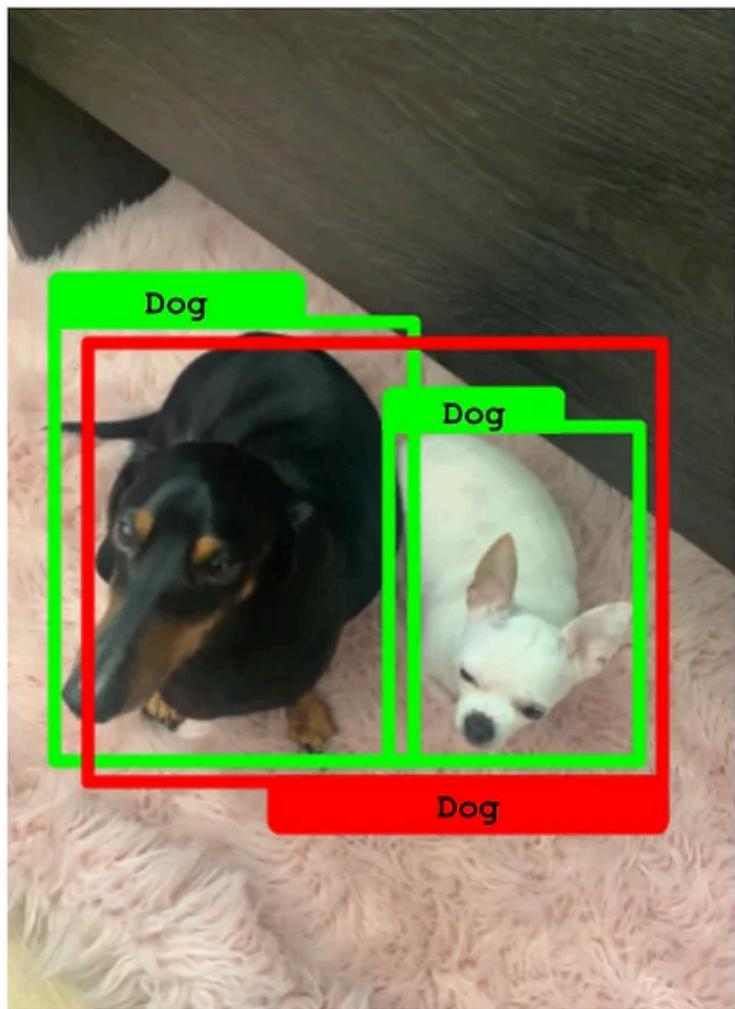


Ejemplo de un falso negativo en la detección de objetos (no hay predicción para una verdad de base dada).

De nuevo, los Verdaderos Positivos no son un caso interesante en este escenario.

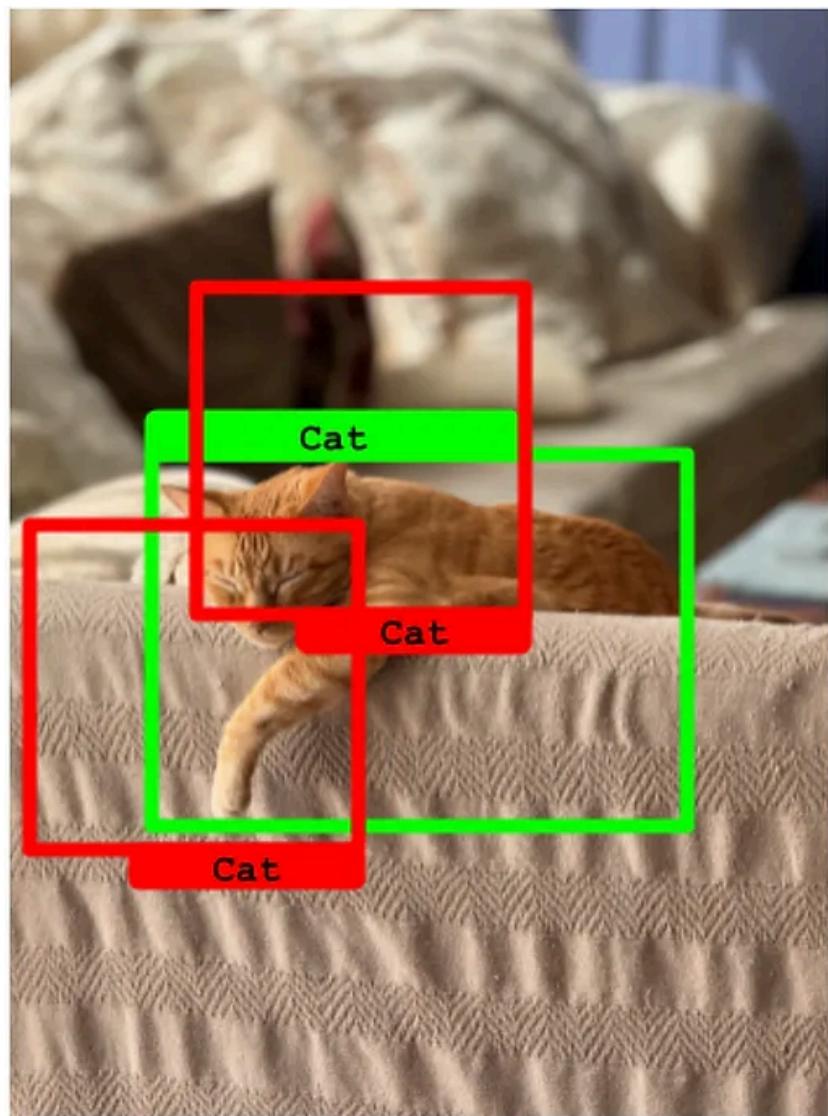
Ejemplos Ambiguos

En la siguiente imagen hay dos verdades de base, pero una sola predicción. En este caso, la predicción solo puede atribuirse a una de las verdades de base (el IoU está por encima del 50%). La otra verdad de base, desafortunadamente, se convierte en un Falso Negativo.



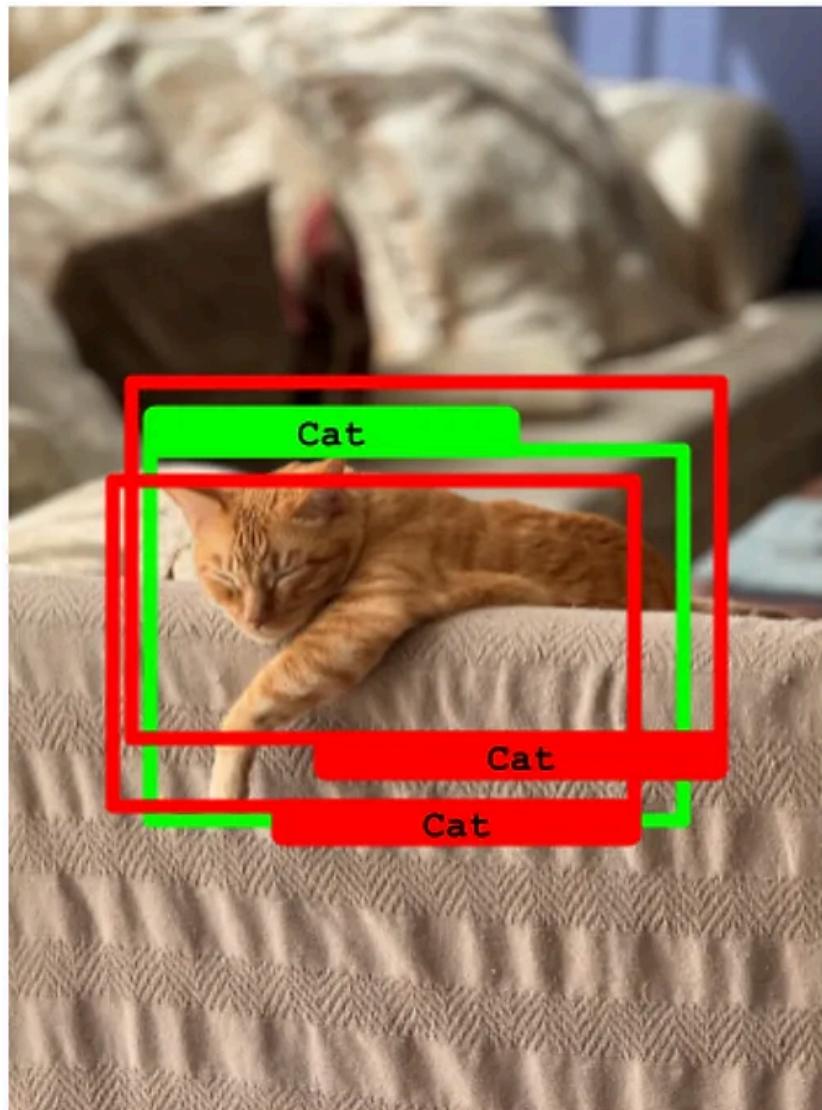
La predicción solo puede asignarse a una verdad de base. La otra verdad de base se convierte en un falso negativo.

En este próximo ejemplo, se hicieron dos predicciones para la misma verdad de base. Sin embargo, ambas tienen un IoU por debajo del 50%. En este caso, las dos predicciones se cuentan como Falsos Positivos y la verdad de base huérfana es un Falso Negativo.



Ambas predicciones tienen un IoU por debajo del 50%. Se convierten en falsos positivos y la verdad de base se convierte en un falso negativo.

En este último ejemplo, tenemos un escenario similar, excepto que ambas predicciones tienen un buen IoU (por encima del 50%). En este caso, la que tiene el mejor IoU se considera un Verdadero Positivo, mientras que la otra se convierte en un Falso Positivo.



Ambas predicciones tienen IoUs por encima del 50%. Una se convierte en un verdadero positivo, la otra en un falso positivo.

Precision, Recall y Puntuación F1

Ahora que sabemos cómo aplicar TP, FP, TN y FN a los detectores de objetos, podemos tomar otras métricas de los clasificadores. Estas son la Precisión, el Recall y la Puntuación F1. Nuevamente, estas métricas se miden por clase. La siguiente tabla las resume:

Precision, Recall and F1-Score

Metric	Formula	Concept
Precision	$P = \frac{TP}{TP + FP}$	For all the predictions made for a class, percentage that were actually correct.
Recall	$R = \frac{TP}{TP + FN}$	From all the objects of a class, percentage that were correctly predicted.
F1	$F1 = 2 \cdot \frac{P \cdot R}{P + R}$	The geometric mean between the precision and recall.

Precisión

Para una clase dada, la precisión nos indica qué porcentaje de las predicciones de la clase realmente pertenecían a esa clase.

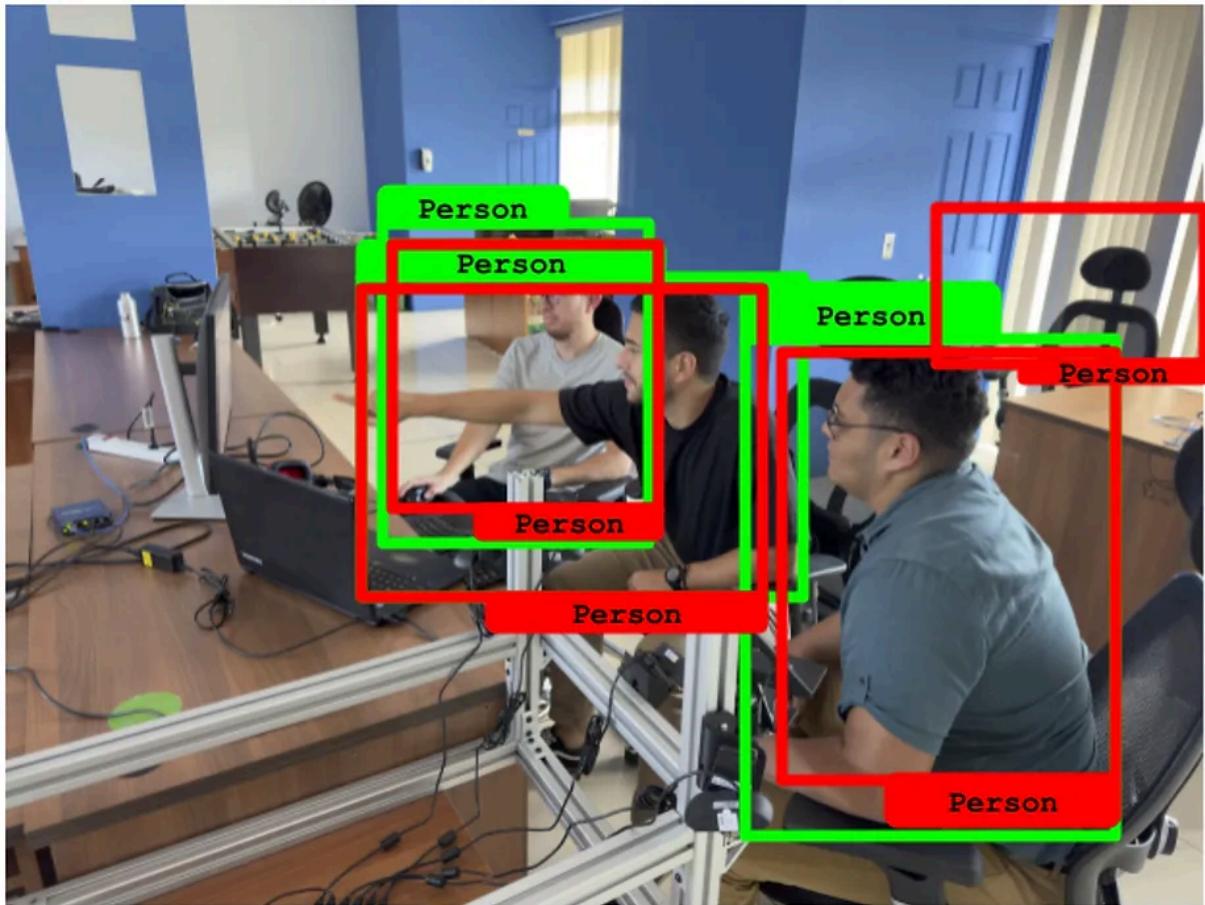


En un modelo de detección de objetos multi-clase, la precisión representa la proporción de verdaderos positivos (detecciones correctas) entre todas las detecciones realizadas por el modelo para una clase particular. Es una medida de cuán precisas son las predicciones del modelo para esa clase específica.

$$\text{Precisión} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Positivos (FP)}}$$

La siguiente imagen muestra el resultado de un detector, donde hubo 3 Verdaderos Positivos y 1 Falso Positivo, resultando en una precisión de:

$$P = \frac{3}{3 + 1} = 0.75 = 75\%$$

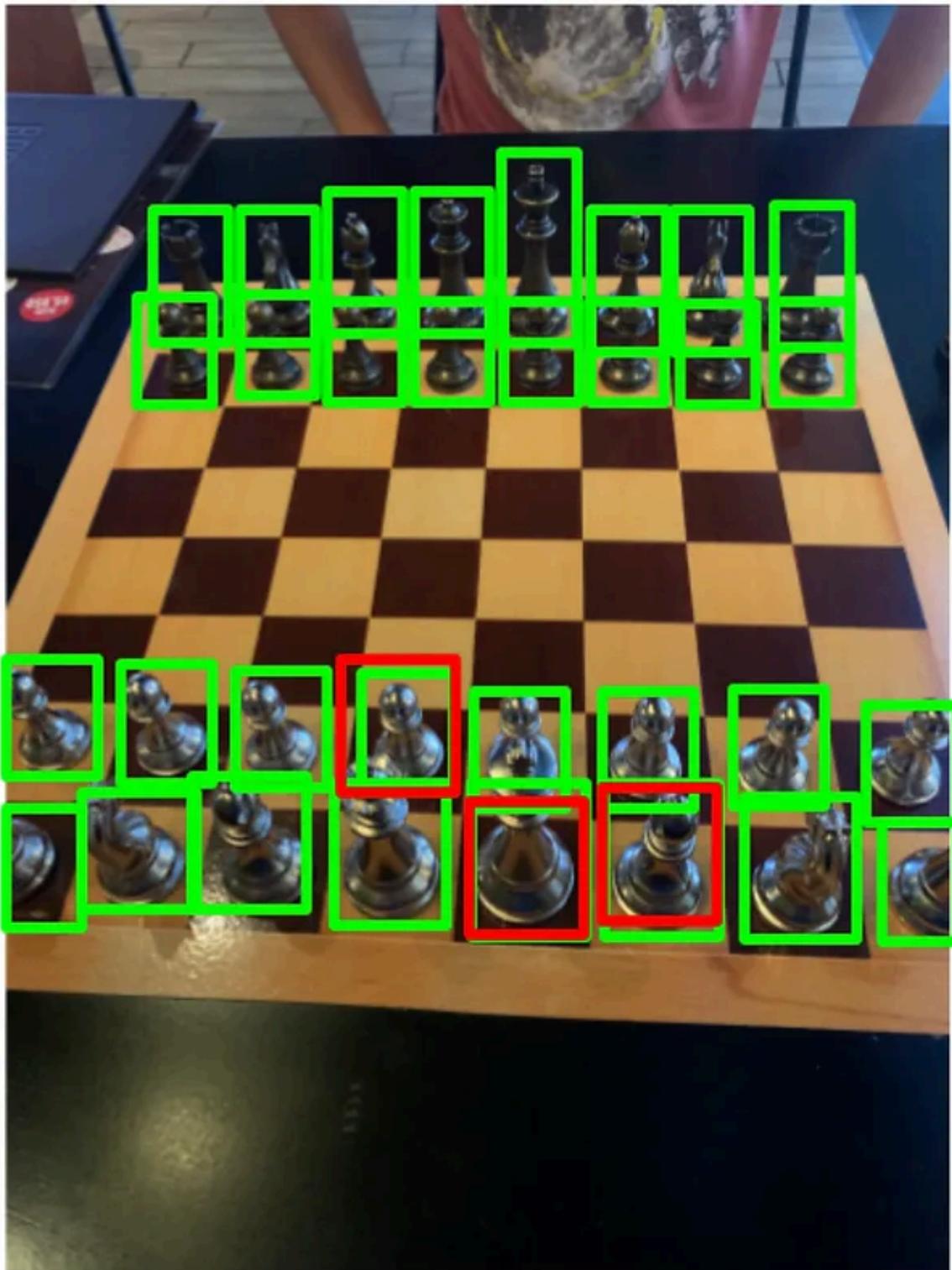


Si bien hay tres personas, se hicieron cuatro detecciones de personas.

Debemos prestar atención a la precisión, si nos importa que nuestro modelo no genere Falsos Positivos. Por ejemplo: en el mercado de valores, es más importante evitar predecir incorrectamente un aumento (falso positivo) porque actuar sobre una falsa creencia de que el mercado va a subir puede llevar a decisiones de inversión equivocadas y pérdidas. Por lo tanto, es preferible que el modelo tenga una alta precisión, aunque eso signifique perder algunas oportunidades de aumento real (falsos negativos), para evitar el riesgo de tomar decisiones basadas en información incorrecta.

El lector astuto podría haber notado que la fórmula de precisión no toma en cuenta los Falsos Negativos. Prestar atención solo a la precisión puede ser muy engañoso. Como muestra el siguiente ejemplo:

$$P = \frac{3}{3 + 0} = 1 = 100\%$$



Aquí se están haciendo 3 detecciones, y son de la clase correcta, por lo que la precisión es del 100%. Sin embargo hay muchas detecciones que no se hicieron.

Como podemos ver, la métrica de precisión resultó en un 100%, pero el modelo está funcionando mal, porque tiene muchos falsos negativos.



No debemos medir la precisión solamente, ya que no se tienen en cuenta los Falsos Negativos.

Recall

Para una clase dada, el recall (o sensibilidad) nos indica qué porcentaje de las instancias reales de la clase fueron predichas correctamente.

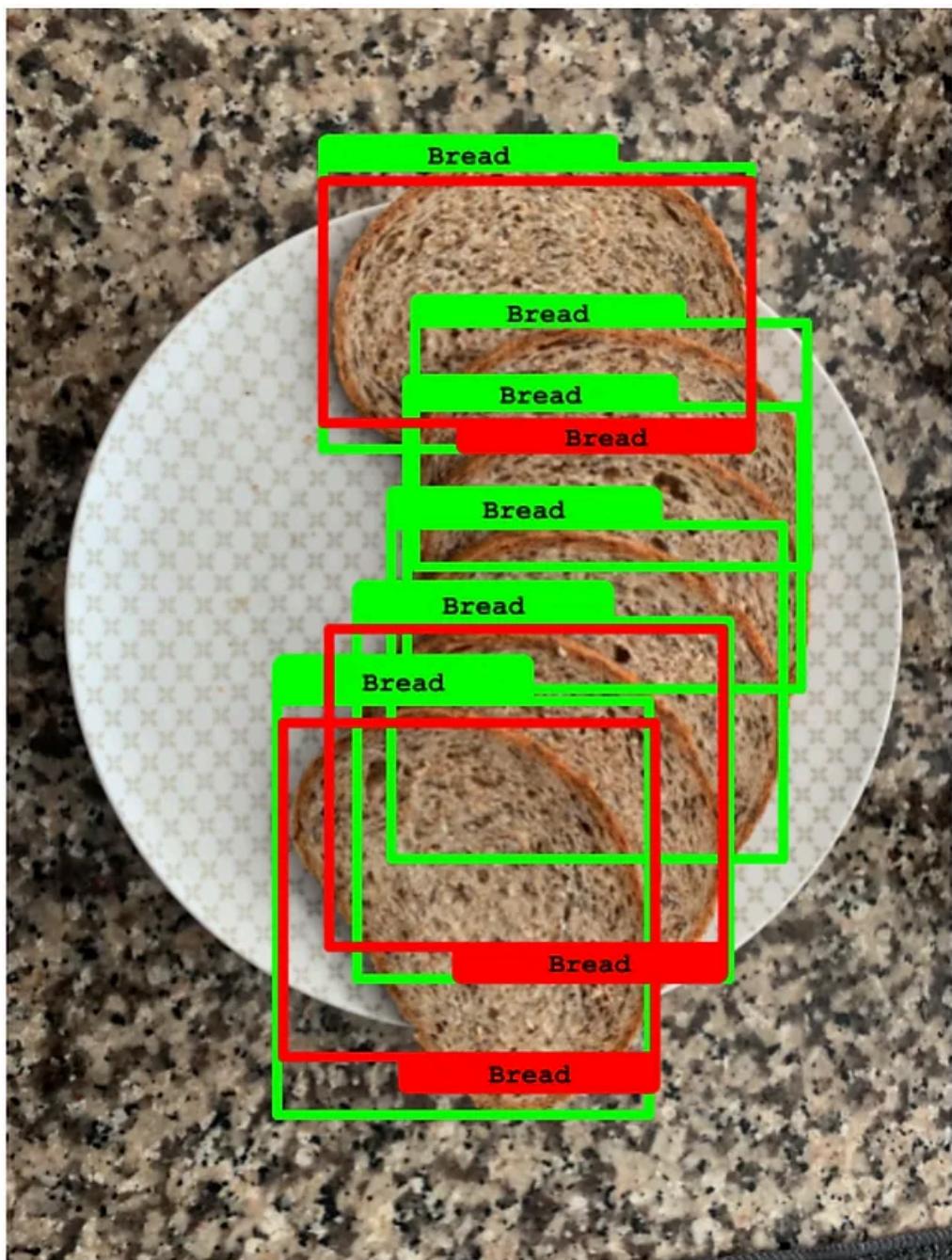


En un modelo de detección de objetos multi-clase, el **recall** (o tasa de verdaderos positivos) representa la proporción de verdaderos positivos (objetos correctamente detectados) entre todos los objetos que realmente están presentes en la imagen para una clase particular. En otras palabras, el recall mide la capacidad del modelo para identificar todos los objetos de una clase específica que realmente existen en la imagen:

$$\text{Recall} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Negativos (FN)}}$$

En la siguiente imagen, el detector predijo correctamente 3 instancias de la clase (Verdaderos Positivos) pero 3 instancias no fueron predichas en absoluto (Falsos Negativos):

$$R = \frac{3}{3+3} = 0.5 = 50\%$$

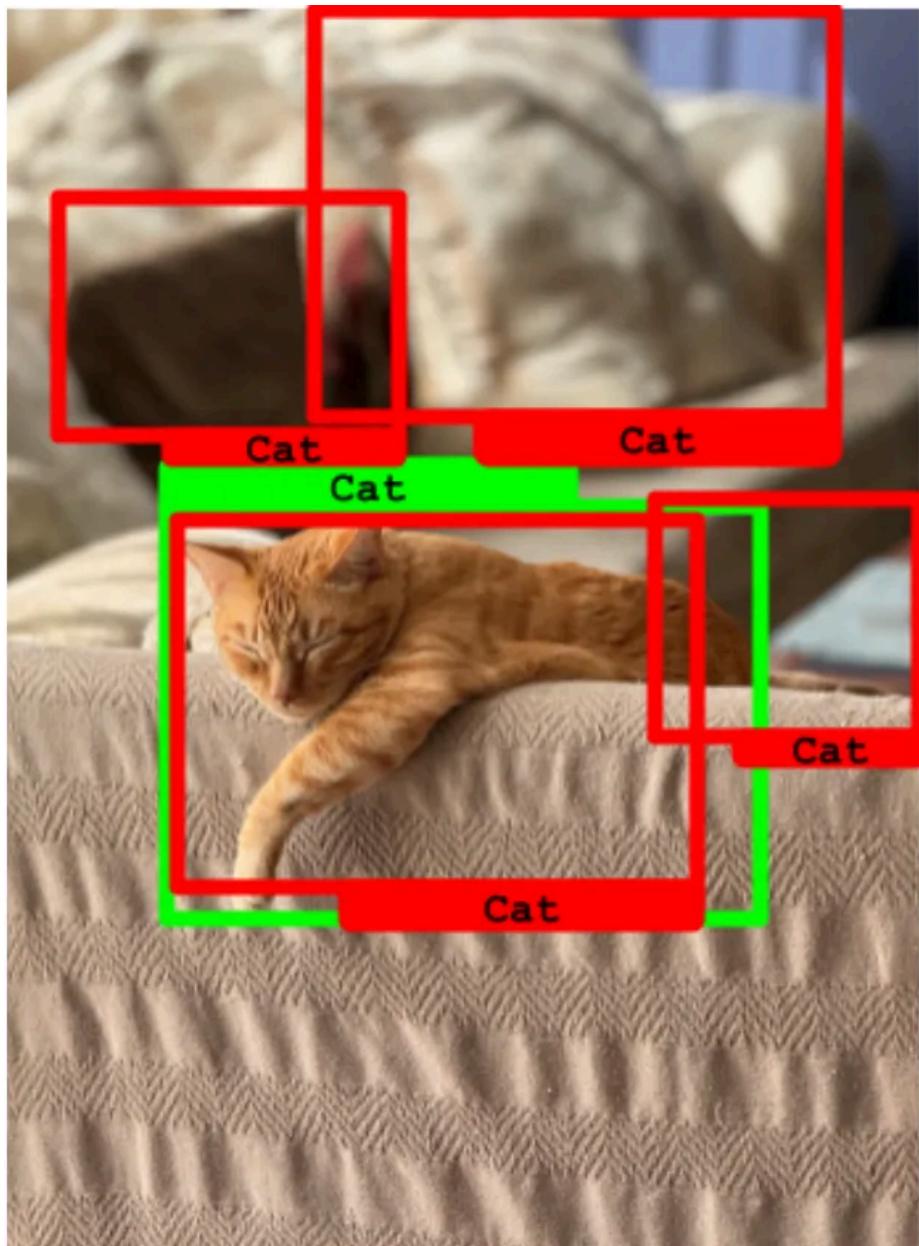


Solo se detectaron 3 rodajas de pan de las 6 existentes.

Prestemos atención al recall, si queremos evitar los Falsos Negativos. Por ejemplo:
es preferible diagnosticar erróneamente una enfermedad terminal, que predecir erróneamente que el paciente está sano.

De nuevo, se puede notar que el recall no tiene en cuenta los Falsos Positivos, por lo que puede ser engañoso si se mide de manera aislada:

$$R = \frac{3}{3+0} = 1 = 100\%$$



El único gato en la imagen fue detectado con éxito, pero hay 3 falsos positivos más.

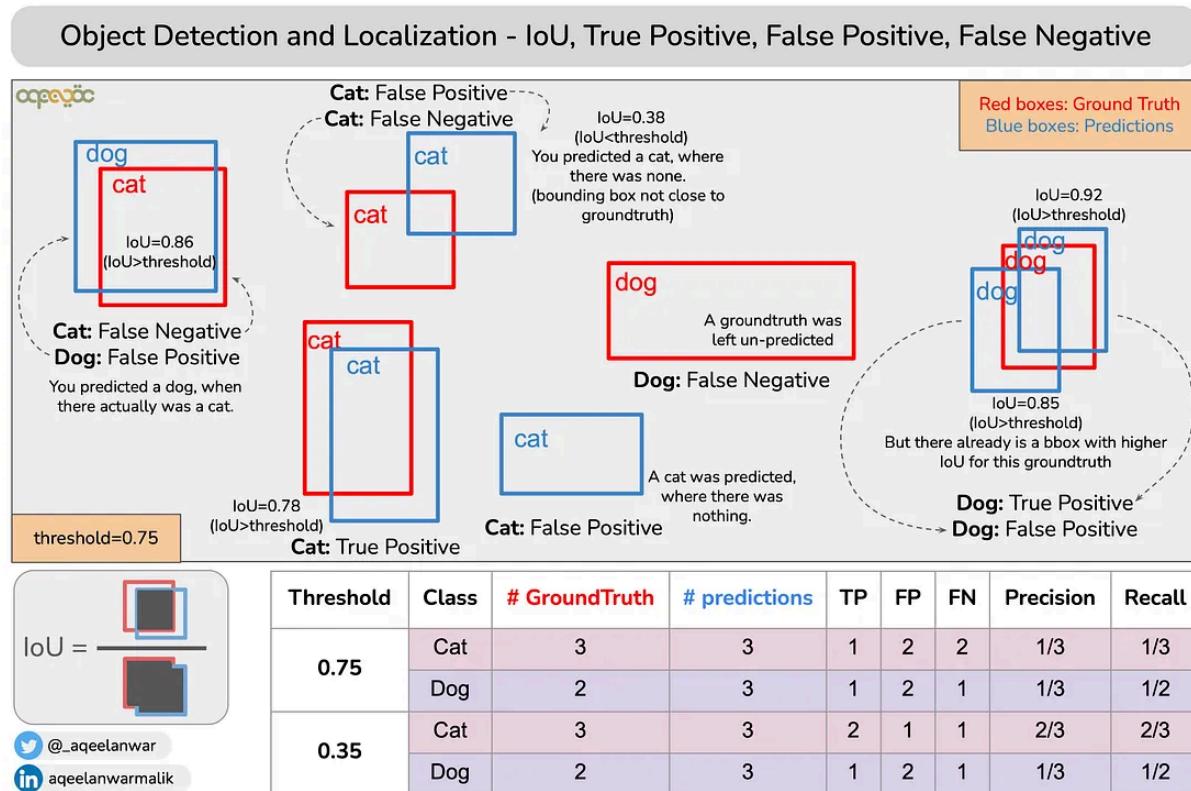
Como podemos ver, la métrica de recall resultó en un 100%, pero el modelo está funcionando mal, porque tiene muchos falsos positivos.



No midas el Recall por sí solo, ya que los Falsos Positivos no se tienen en cuenta.

Resumen de Precision y Recall

Aquí vemos un ejemplo de los cálculos de precision y recall:



Fuente: <https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b>

Puntuación F1

La puntuación F o puntuación F1 es una métrica que combina tanto la precisión como el recall, y nos proporciona un buen equilibrio entre ellos.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Entonces, usando la misma imagen anterior, el modelo devolvería una puntuación F1 de:

$$F1 = 2 \cdot \frac{0.5 \cdot 0.75}{0.5 + 0.75} = 0.6 = 60\%$$



La puntuación F1 mide un equilibrio entre precision y recall.

Curva de Precisión-Recall

Hasta ahora, hemos visto que la precisión y el recall caracterizan diferentes aspectos de un modelo. En algunos escenarios puede ser más conveniente tener una mayor precisión, y en otros más conveniente tener un mayor recall. Podemos ajustar estas métricas afinando el umbral de confianza (confidence threshold) del detector de objetos.

En un modelo de detección de objetos, el **confidence score** (puntuación de confianza) es una medida de cuán seguro está el modelo de que una detección particular es correcta. Esta puntuación es un valor numérico entre 0 y 1, donde un valor cercano a 1 indica una alta confianza del modelo en que la detección es correcta, y un valor cercano a 0 indica baja confianza.



Durante la inferencia, se establece un umbral de confianza. Las detecciones con una puntuación de confianza por encima de este umbral se consideran válidas, mientras que las detecciones con puntuaciones por debajo se descartan. Ajustar este umbral afecta el balance entre precisión y recall.

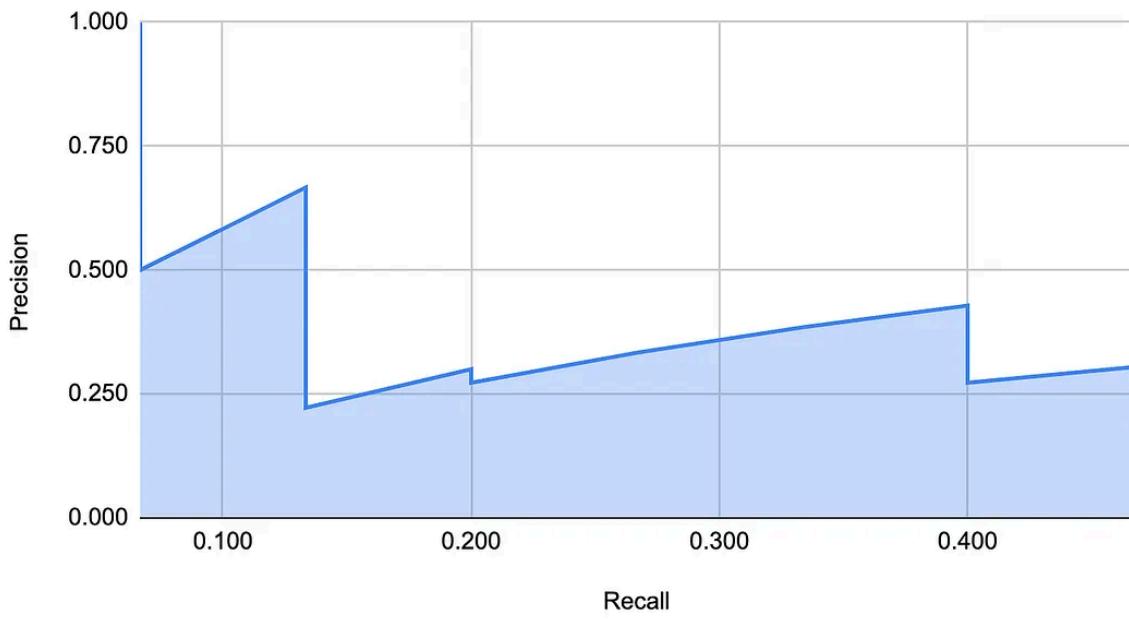


Un umbral más alto generalmente aumenta la precisión (menos falsos positivos) pero puede reducir el recall (más falsos negativos). Un umbral más bajo generalmente aumenta el recall pero puede reducir la precisión.

- **Umbral de confianza:** Es el valor a partir del cual el modelo considera que una predicción es válida. Si la probabilidad predicha por el modelo es mayor o igual a este umbral, se considera que el objeto está presente.
- **Efectos de aumentar el umbral:**
 - **Reducir el riesgo de sobre-detección:** Un umbral más alto significa que el modelo es más estricto, y solo considerará las predicciones con alta confianza. Esto reduce la probabilidad de detectar objetos que no están realmente presentes, es decir, reduce los falsos positivos.
 - **Incrementar el riesgo de detecciones perdidas:** Al ser más estricto, el modelo puede pasar por alto objetos que están presentes pero que no cumplen con el alto umbral de confianza, aumentando los falsos negativos.
- **Ejemplo con umbral = 1.0:**
 - Si el umbral es 1.0, el modelo solo aceptará predicciones con una confianza del 100%. Dado que es casi imposible que el modelo tenga tanta confianza, no se detectará ningún objeto.
 - **Precisión:** Será 1.0 porque no hay falsos positivos (no hay predicciones incorrectas).
 - **Recall:** Será 0.0 porque no se detecta ningún objeto real (todos los objetos verdaderos son falsos negativos).
- **Ejemplo con umbral = 0.0:**
 - Si el umbral es 0.0, el modelo considerará cualquier predicción, sin importar cuán baja sea su confianza.
 - **Precisión:** Será 0.0 porque el modelo detectará muchos objetos incorrectamente (muchos falsos positivos).
 - **Recall:** Será 1.0 porque el modelo detectará todos los objetos reales (no hay falsos negativos).

Resulta que hay una forma muy conveniente de visualizar la respuesta del modelo a una clase específica en diferentes umbrales de clasificación. Esta es la curva de precisión-recall, y se muestra en la siguiente figura:

PR Curve

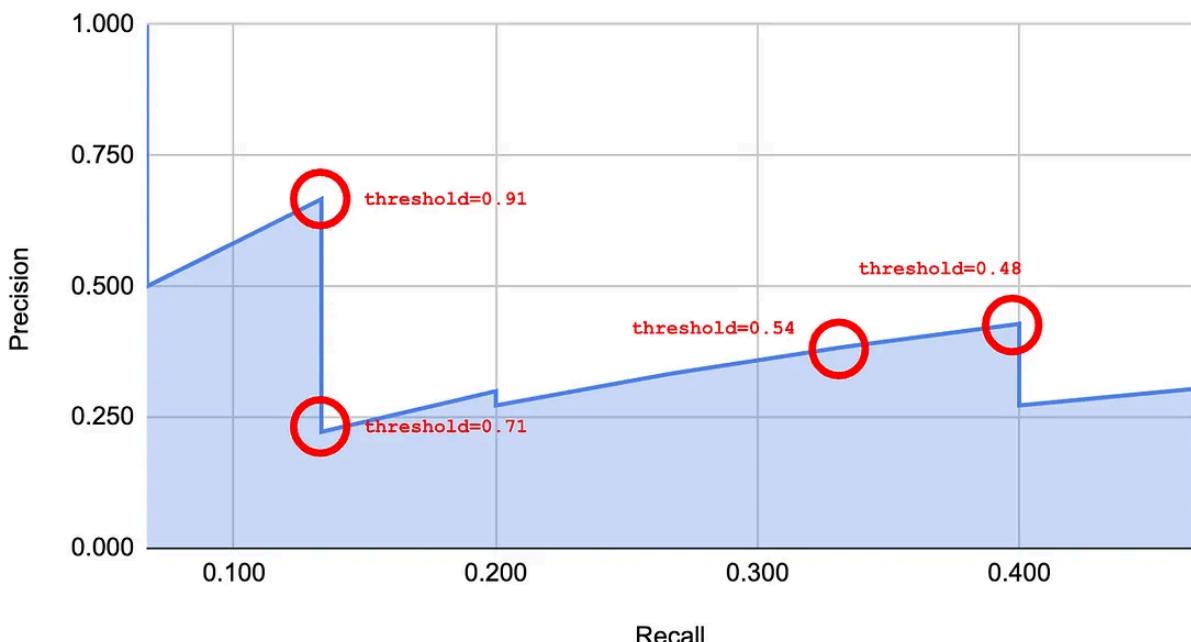


Ejemplo de una curva PR (Precision - Recall).

El proceso de creación de una curva PR es:

1. Comenzamos configurando el umbral de confidencia en 1 (la precisión será 1). El recall será 0 (si hacemos los cálculos). Marcamos este punto en la curva.
2. Luego disminuimos la confidencia hasta obtener la primera detección. Calculamos la precisión y el recall. La precisión será nuevamente 1 (porque no tienes falsos positivos). Marcamos este punto en la curva.
3. Continuamos disminuyendo el umbral hasta que ocurra una nueva detección. Marcamos ese punto en la curva.
4. Repetimos hasta que el umbral sea lo suficientemente bajo para que el recall sea 1. En ese punto, la precisión probablemente será alrededor de 0.5.
5. La siguiente imagen muestra el mismo gráfico con el umbral evaluado en algunos puntos.

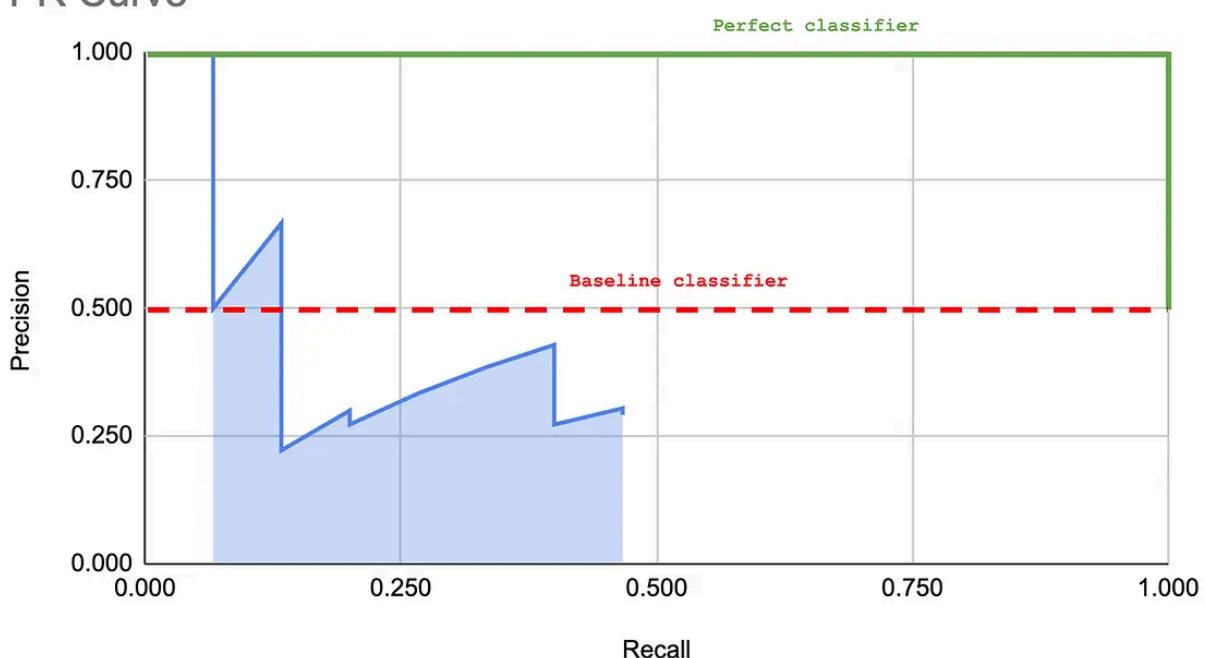
PR Curve



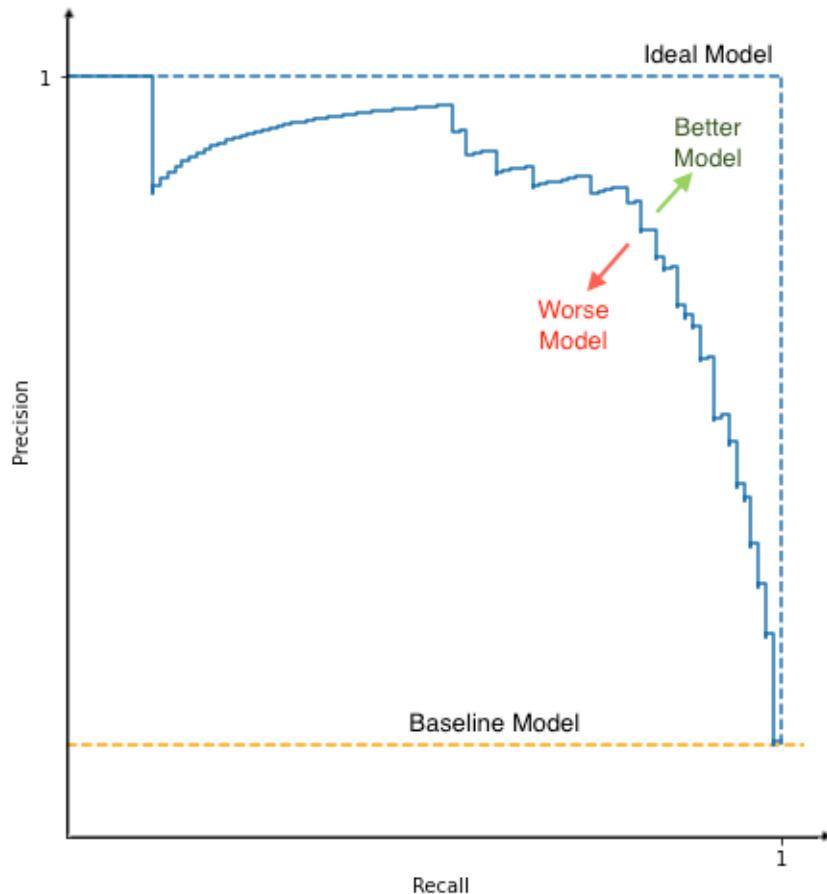
Los valores del umbral para algunos puntos en la curva PR

Se puede observar que la forma de esta curva puede usarse para describir el rendimiento del modelo. La siguiente figura muestra un clasificador "línea de base" y un clasificador "perfecto". Cuanto más cerca esté el clasificador de la curva "perfecta", mejor.

PR Curve



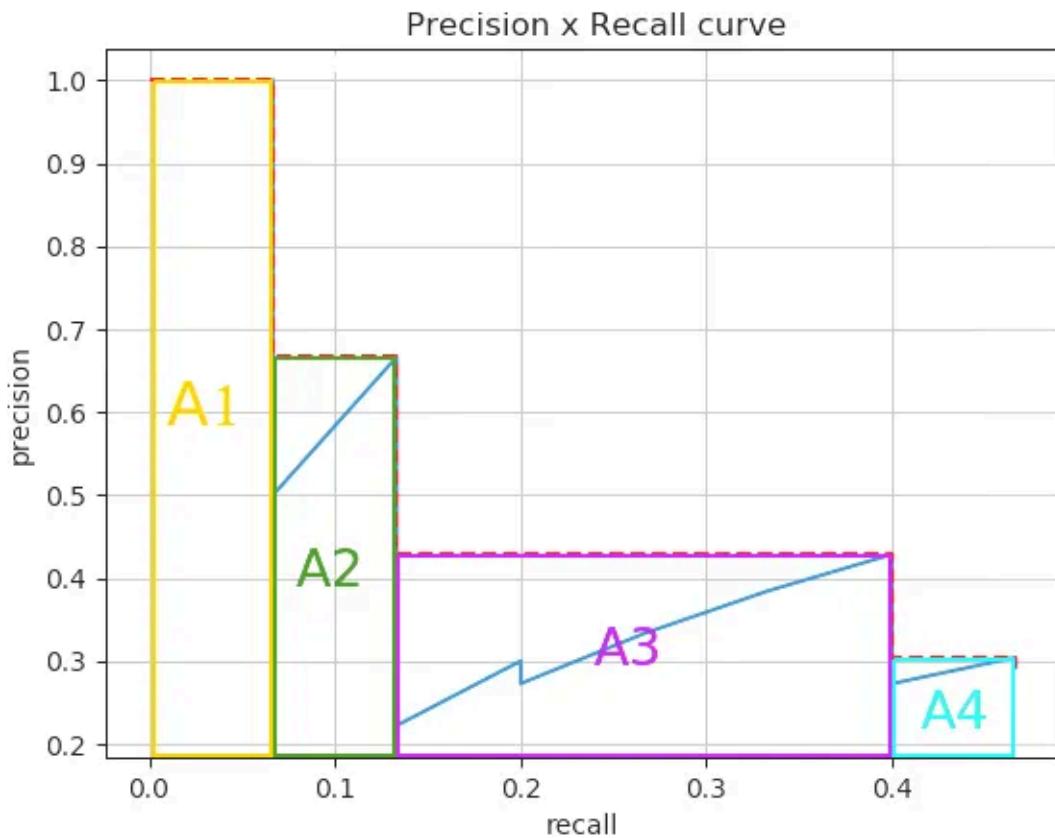
En el caso de un buen modelo de aprendizaje automático, no ocurrirá sobre-detección incluso si se reduce el umbral (se aumenta el Recall), y la Precisión se mantendrá alta. Por lo tanto, cuanto más alta esté la curva hacia la derecha en la gráfica, mejor será el modelo de aprendizaje automático.



AP (Average Precision)

Trazar la Curva de Precisión-Recall es un proceso que consume tiempo, y como la curva a menudo zigzaguea, el juicio sobre si el modelo es bueno o no es subjetivo.

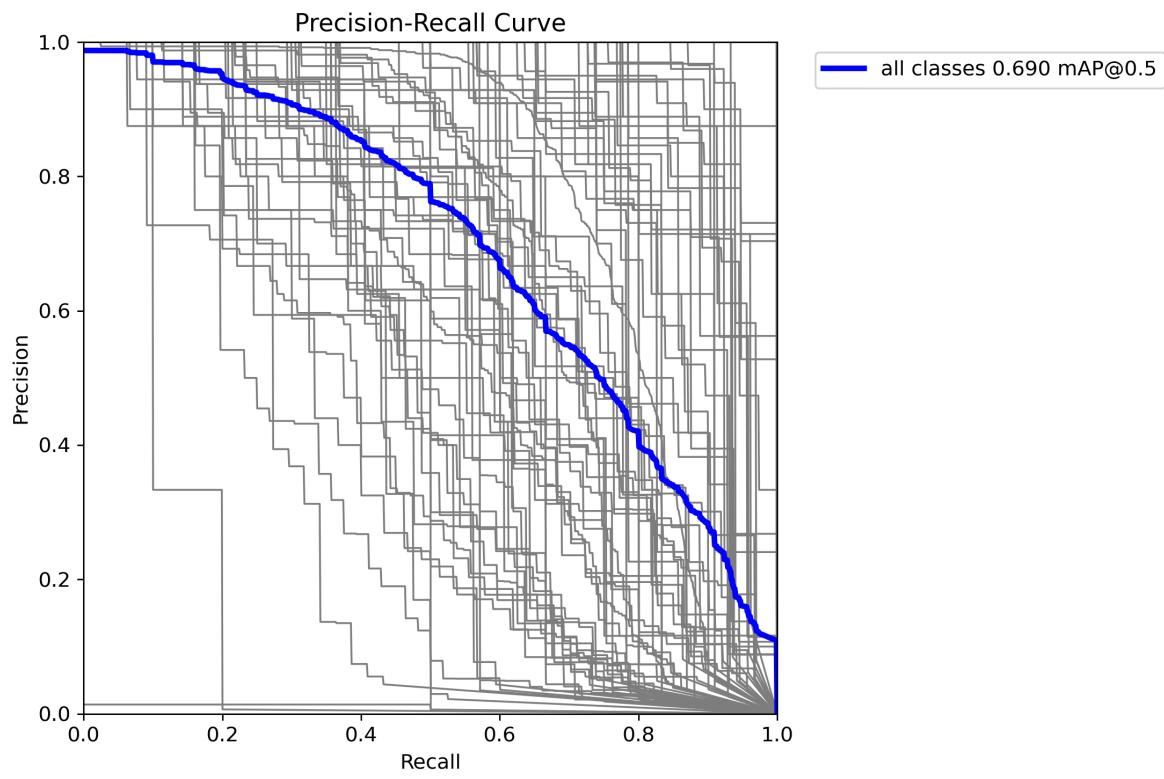
Una forma más intuitiva de evaluar los modelos es el AP (Precisión Promedio), que representa el área bajo la curva (AUC) de la Curva de Precisión-Recall. Cuanto más alta esté la curva en la esquina superior derecha, mayor será el área, por lo que cuanto mayor sea el AP, mejor será el modelo de aprendizaje automático.



El AP (Average Precision) es el área bajo la curva (AUC) de la curva PR.

mAP (Precisión Promedio Media)

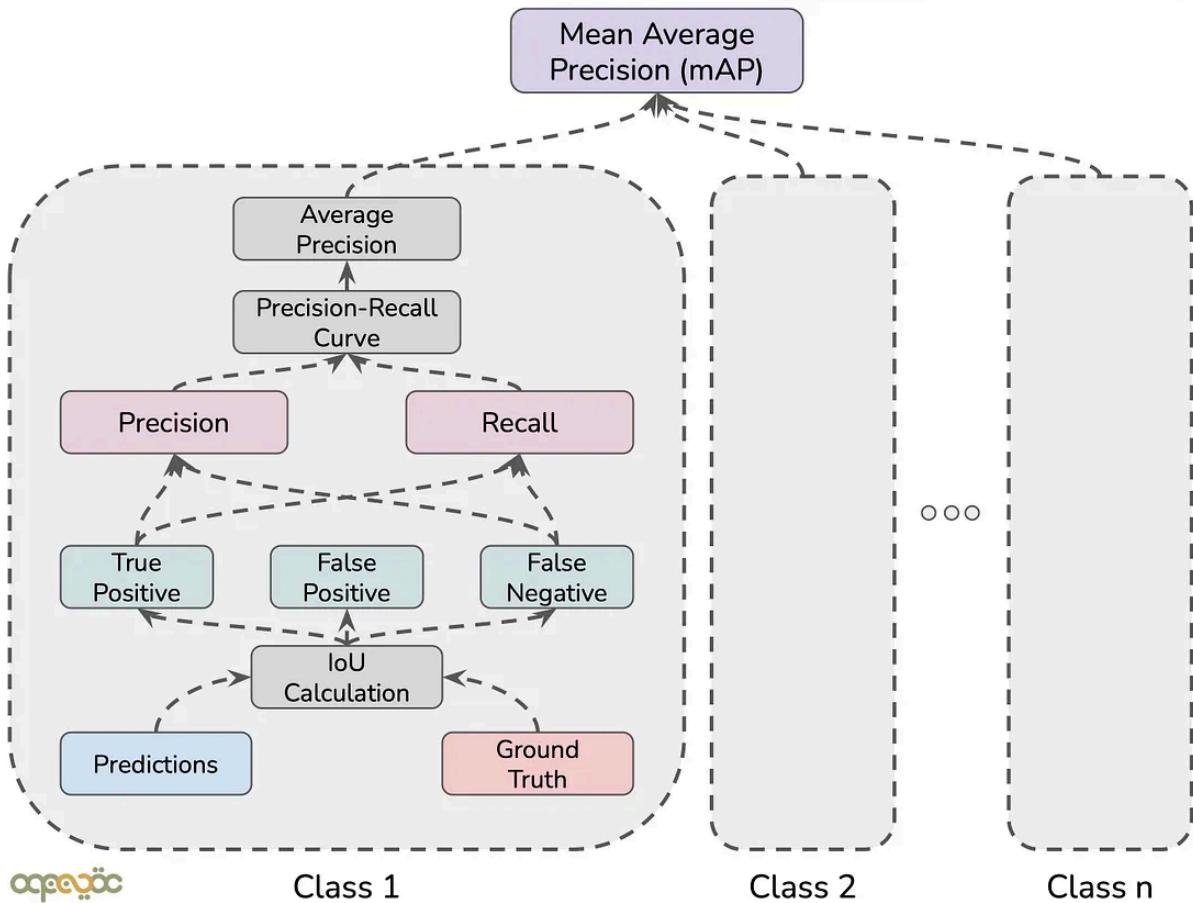
El mAP extiende el concepto de AP calculando el promedio de los valores de AP a través de múltiples clases de objetos. Esto es útil en escenarios de detección de objetos multiclase para proporcionar una evaluación integral del rendimiento del modelo.



En un modelo multi-clase, promediamos los AP de cada clase para obtener una métrica media de la performance del modelo con todas las clases.

Aquí vemos un resumen de todos los pasos necesarios hasta llegar al mAP:

How to calculate mean average precision (mAP)



Resumen de métricas

Métrica	Explicación	Comportamiento (para obtener un mejor modelo)
Intersección sobre Unión (IoU)	Mide la superposición entre un cuadro delimitador predicho y uno de verdad. Evalúa la precisión de la localización de objetos.	Un buen modelo debe tener altos valores de IoU, indicando una localización precisa de los objetos predichos respecto a la verdad de base.
Precisión Promedio (AP)	Calcula el área bajo la curva precisión-recall, proporcionando un valor que resume el rendimiento de precisión y recall del modelo.	En un buen modelo, el AP debe ser alto, mostrando que el modelo tiene tanto una alta precisión como un buen recall en sus predicciones.
Precisión Promedio Media (mAP)	Extiende el concepto de AP promediando los valores de AP a través de múltiples clases de	Un buen modelo tendrá un mAP alto, lo que indica un rendimiento

Métrica	Explicación	Comportamiento (para obtener un mejor modelo)
	objetos. Útil para una evaluación integral en escenarios multiclas.	consistente y preciso en todas las clases de objetos.
Precisión y Recall	La precisión mide la proporción de verdaderos positivos entre todas las predicciones positivas; el recall mide la proporción de verdaderos positivos entre todos los positivos reales.	En un buen modelo, la precisión y el recall deben ser altos, indicando que el modelo tiene pocos falsos positivos y una alta tasa de detección de las instancias correctas.
Puntuación F1	Media armónica de precisión y recall, proporcionando una evaluación equilibrada del rendimiento del modelo considerando falsos positivos y negativos.	Un buen modelo tendrá una alta puntuación F1, lo que refleja un equilibrio óptimo entre precisión y recall, con bajas tasas de falsos positivos y negativos.

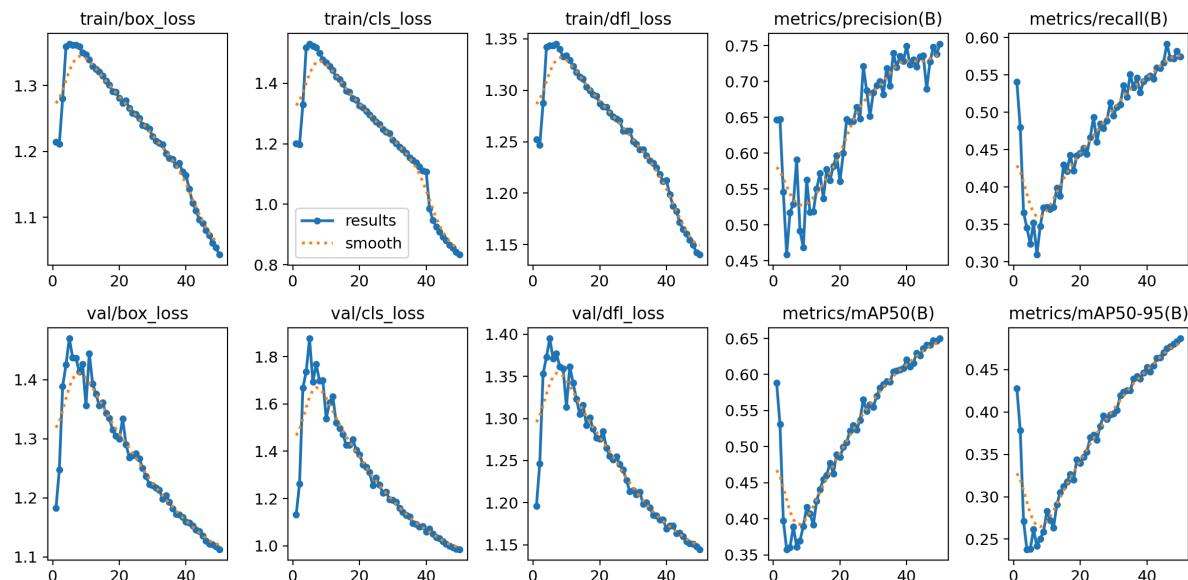
Recomendaciones para mejorar un modelo entrenado

Cuando las métricas de nuestro entrenamiento no son como lo esperamos, podemos tomar ciertas acciones para revertirlas o mejorar nuestro modelo. Los cambios pueden implicar tanto operaciones con el dataset, como modificaciones en la arquitectura e hiperparámetros. A continuación algunas recomendaciones:

Métrica	Bajo Rendimiento	Mejoras en el Entrenamiento
Bajo mAP	Indica que el modelo puede necesitar refinamientos generales.	Realizar ajustes generales en la arquitectura del modelo y en el proceso de entrenamiento.
Bajo IoU	El modelo puede tener dificultades para localizar objetos con precisión.	Probar diferentes métodos de cuadros delimitadores y ajustar el algoritmo de localización.
Baja Precisión	El modelo puede estar detectando demasiados objetos inexistentes.	Ajustar los umbrales de confianza para reducir las detecciones falsas.
Bajo Recall	El modelo podría estar omitiendo objetos reales.	Mejorar la extracción de características o usar más datos de entrenamiento.
Puntuación F1 Desbalanceada	Hay una disparidad entre la precisión y el recall.	Equilibrar las métricas de precisión y recall ajustando el proceso de entrenamiento y la arquitectura del modelo.
AP Específico por Clase	Puntuaciones bajas pueden destacar clases	Enfocarse en mejorar el rendimiento del modelo en las clases problemáticas ajustando el entrenamiento específico y

Métrica	Bajo Rendimiento	Mejoras en el Entrenamiento
	con las que el modelo tiene dificultades.	asegurando una mejor representación de esas clases en el dataset.

Monitoreo del entrenamiento



Métricas durante el entrenamiento de un modelo YoloV8. El eje horizontal representa las épocas.

Estas gráficas representan diferentes métricas durante el entrenamiento y validación de un modelo YOLOv8 usando la librería [ultralytics](#). Aquí está la explicación de cada una:

Fila Superior

1. **train/box_loss**: Pérdida de los cuadros delimitadores durante el entrenamiento. Muestra cómo la pérdida disminuye a medida que el modelo mejora en ajustar los cuadros delimitadores a los objetos reales.
2. **train/cls_loss**: Pérdida de clasificación durante el entrenamiento. Indica cómo el modelo mejora en predecir las clases correctas.
3. **train/dfl_loss**: Pérdida durante el entrenamiento para la función de distribución. Representa la mejora en la localización precisa de los objetos. La función de pérdida de enfoque distribuido (DFL), está diseñada para mejorar la precisión en la detección de objetos. A diferencia de las funciones de pérdida estándar, DFL se concentra en los ejemplos difíciles de detectar. Ayuda al modelo asignando más peso a las instancias desafiantes, facilitando así el aprendizaje a partir de ellas.

4. **metrics/precision(B)**: Precisión del modelo durante el entrenamiento. Aumenta a medida que el modelo hace menos predicciones falsas.
5. **metrics/recall(B)**: Recall durante el entrenamiento. Aumenta a medida que el modelo captura más objetos correctos.

Fila Inferior

1. **val/box_loss**: Pérdida de los cuadros delimitadores durante la validación. Similar a la métrica de entrenamiento, pero en el conjunto de datos de validación.
2. **val/cls_loss**: Pérdida de clasificación durante la validación. Similar a la métrica de entrenamiento, pero en el conjunto de datos de validación.
3. **val/dfl_loss**: Pérdida durante la validación para la función de distribución. Similar a la métrica de entrenamiento, pero en el conjunto de datos de validación.
4. **metrics/mAP50(B)**: Precisión media a un IoU de 50% durante el entrenamiento. Indica la precisión global del modelo.
5. **metrics/mAP50-95(B)**: Precisión media a IoU entre 50% y 95% durante el entrenamiento. Proporciona una visión más detallada de la precisión del modelo a diferentes niveles de IoU.

Interpretación General

- **Pérdidas (Losses)**: Las pérdidas para los bounding boxes, la clasificación y la función de distribución deben disminuir a lo largo del tiempo, lo que indica que el modelo está mejorando. Una función de pérdida calcula el "error" o "costo" asociado a una predicción incorrecta. El objetivo principal del entrenamiento de un modelo es minimizar esta función de pérdida. Al minimizar la pérdida, el modelo está efectivamente reduciendo la diferencia entre lo que predice y lo que son los valores verdaderos. Esto implica que el modelo está "aprendiendo" de los errores y mejorando sus predicciones. Al observar tanto la pérdida de entrenamiento como la de validación, se puede detectar si el modelo se está sobreajustando. El sobreajuste ocurre cuando un modelo aprende patrones que son demasiado específicos para los datos de entrenamiento y no generaliza bien a nuevos datos. Si la pérdida de entrenamiento continúa disminuyendo mientras que la pérdida de validación aumenta o se estanca, esto podría ser una señal de sobreajuste.
- **Métricas (Metrics)**: Las métricas de precision, recall y mAP deben aumentar, lo que indica que el modelo está haciendo predicciones más precisas y recuperando más objetos correctamente.

Entrenamiento y log de eventos

El armado y entrenamiento de muchos modelos, se basan en frameworks de Deep Learning (como Tensorflow, Pytorch, etc.), que ofrecen una gran cantidad de utilidades y objetos para la manipulación de los mismos y facilitar la utilización de múltiples GPU en el entrenamiento.

Al utilizar la librería `ultralytics`, y como en otros casos, vemos un log que nos permite monitorear el progreso del entrenamiento de un modelo. Dado que este proceso puede durar desde minutos hasta varios días, es importante poder enterarnos de algún modo si el entrenamiento progresá positivamente, sobre todo, cuando consumimos recursos costosos de hardware.

Aquí vemos un extracto de un log de entrenamiento de un modelo YoloV8 usando `ultralytics`:

```
train: Scanning /content/yolo_dataset/train/data.cache... 12829 images, 116 backgrounds
...
val: Scanning /content/yolo_dataset/val/data.cache... 496 images, 4 backgrounds
...
Starting training for 100 epochs...

Epoch  GPU_mem  box_loss  cls_loss  dfl_loss Instances     Size
1/100   12.5G    1.367    1.544    1.348      16       640: 100% 203/203 [02
          Class  Images Instances Box(P)      R mAP50 mAP50-95): 100
          all    500    4056   0.559    0.433    0.457   0.322
...
          Epoch  GPU_mem  box_loss  cls_loss  dfl_loss Instances     Size
100/100  10.7G    1.026    0.8351   1.13       2       640: 100% 203/203 [02
          Class  Images Instances Box(P)      R mAP50 mAP50-95): 100
          all    500    4056   0.77     0.57    0.663   0.486
```

Algunos puntos a destacar de este log:

- Entrenamos el modelo con 100 épocas.
- El set de entrenamiento es de 12.945 imágenes. El tamaño del batch es de 64. La cantidad de lotes de entrenamiento es de $12.945 / 64 = 202,26 \Rightarrow 203$ iteraciones por época.

- El mAP inició en 0.457 y terminó 0.663, y el mAP50-95 inició en 0.322 y terminó 0.486. Estos valores indican que nuestro modelo ha mejorado en su capacidad de detectar objetos con mayor precisión, según el proceso de validación (contra el dataset de validación de 500 imágenes). Estas métricas se calculan teniendo en cuenta todas las clases.
- Los valores de pérdida **box_loss**, **cls_loss** y **dfl_loss** han disminuido, con indicadores que nuestro modelo ha aprendido desde su estado inicial.

Más información

<https://www.youtube.com/watch?v=q7LwPoM7tSQ>