DETECCIÓN Y TRACKING

Matías Mancini, Damián Nuñez.

Departamento de Telecomunicaciones, Facultad de Ing., Universidad Nacional de Río Cuarto, Córdoba, Argentina.

Resumen: Este trabajo se divide en tres partes, en una primera parte se presentará modelo de aprendizaje YOLO, en una segunda parte se mostrará cómo se realiza tracking, por último, se verá cómo esto puede ser llevado a la electrónica mediante el uso de una placa raspberryPI. YOLO se utilizó en sus versiones 3 y 8 para detección de objetos. Con sus modelos pre-entrenados, se realizó detección y posterior tracking de dos maneras diferentes, inferencia frame a frame, y detección sólo en el primer frame y posterior tracking mediante match template. Se finalizará mostrando otras posibles aplicaciones, además del tracking.

Palabras claves: YOLO, detección, tracking, raspberry, electrónica.

1. INTRODUCCIÓN

En este documento primero se hablará sobre el modelo de aprendizaje YOLO para reconocimiento de diferentes objetos y detección de los mismos. Después se explicará cómo funciona el tracking, y cómo la librería de OpenCV realiza un "match template", para reconocer coincidencias en una imagen.

En una tercera instancia del presente documento hablaremos sobre una placa raspberry PI, y cómo ésta puede unir la programación con la electrónica, como, por ejemplo, mover motores.

Por último, mencionaremos otras posibles aplicaciones sobre estos algoritmos capaces de detectar objetos, y sacaremos algunas conclusiones.

2. YOLO

YOLO (acrónimo de "You Only Look Once", en español "miras una sola vez") es un modelo de aprendizaje, utilizado para detección de objetos en imágenes o fotogramas de videos. Es utilizado principalmente en videos en tiempo real (stream), como cámaras de vigilancia, vehículos autónomos y robótica, debido a su gran velocidad a la hora de hacer inferencias. Como su nombre lo indica, es rápido ya que pasa la imagen por su red neuronal una sola vez, esto lo hace muy eficiente para las tareas mencionadas. Es capaz de detectar varios objetos en una sola imagen al pasarla una única vez por toda su red, creando cajas delimitadoras, o "bounding boxes", que identifican a los objetos dentro de dichas cajas.

2.1 Principio de funcionamiento.

1- Divide la imagen en una cuadrícula de S x S. En la versión 1 y 2 de YOLO hace una sola división S x S (13x13 celdas), a partir de la versión 3 en adelante hace varias capas de una misma imagen, donde cada capa realiza un grillado diferente, por ejemplo, 13x13, 26x26 y 52x52.

2- Predicción de cajas y clases. Cada celda de la cuadrícula predice:

- Coordenadas de las cajas delimitadoras.

- Confianza de que una caja contenga un objeto.

- Clase del objeto detectado.

3- Umbral de confianza. En una misma capa (división de celdas), se descartan todas las cajas pertenecientes a una misma celda que contienen el mismo objeto, dejando sólo la de mayor probabilidad.

4- NMS (Supresión de no máximos). Se eliminan objetos duplicados entre dos o más capas, cuando se considera que se trata de un mismo objeto, de acuerdo a la relación de Intersección sobre Unión (IoU).

Con este último paso lo que se logra es eliminar duplicados, o falsos positivos, como se conoce en detección.

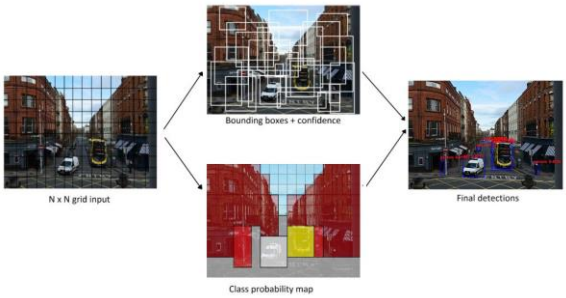

Figura 1.- Representación gráfica del funcionamiento de YOLO.

2.2 Rendimiento.

En la tabla 1 podemos visualizar una comparativa entre las diferentes versiones de YOLO. Nota: los valores de la versión 3 fueron tomados de otra fuente, por lo que es probable que no sean del todo correctos, o en todo caso, las pruebas para obtener los resultados no se realizaron en las mismas condiciones.

Tabla 1. Tabla comparativa entre las versiones más importantes de YOLO.

Modelo	mAP	Inferencia-CPU (ms)
YOLOv3n	51.5	45
YOLOv8n	37.3	80.4
YOLOv11n	39.5	56.1

3. TRACKING

Encontramos dos maneras diferentes de realizar tracking. La primera es realizar una inferencia con YOLOv11n, frame a frame.

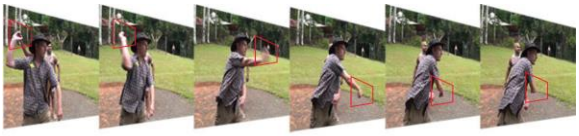

Figura 2. Inferencia en cada frame.

La segunda es utilizar YOLOv3 para hacer una detección sólo en el primer frame, y posterior a tener las bounding box, utilizarlas como plantillas, en conjunto con la herramienta "match template" de OpenCV, para buscar coincidencias entre el frame y la plantilla (obtenida de la caja delimitadora otorgada por YOLO).

Figura 3.- Herramienta Match Template de OpenCV.

4. RASPBERRY PI

Una raspberry pi es una computadora compacta, de bajo costo y bajo consumo energético. Posee memoria RAM, CPU y GPU, y puertos periféricos como HDMI, USB y Ethernet. También tiene su propio sistema operativo basado en Linux.

En su comienzo fue creada con el fin de que las escuelas tuvieran la posibilidad de acceder a computadoras. Con el tiempo se fue incrementando cada vez más sus capacidades de hardware, lo que permite exigirlas un poco más. Al día de hoy, una raspberry con 8 Gb de memoria RAM ya puede soportar programas más complejos, como el que nosotros implementamos en este documento. Dando lugar lo que se conoce como procesamiento en el borde. Esto disminuye mucho la latencia y da la posibilidad de hacer incluso un tracking en directo de un objeto o varios objetos, como por ejemplo un vehículo con conducción autónoma.

Esta placa nos permitió enviar imágenes desde un vehículo a control remoto, para hacerles un procesamiento y detección de objetos desde una PC, ambos dentro de la misma red WiFi. Se utilizó un router configurado con direcciones ip fijas, tanto para la PC como para la raspberry, y de esta manera facilitar la conexión.

CONCLUSIONES

Para finalizar este documento, mostraremos otras posibles aplicaciones de una red neuronal como YOLO.

Además del seguimiento de objetos, YOLO puede ser utilizado para contar, ya sea tomando una fotografía con un dron a un campo y contar cuantos animales hay de cada clase. Hasta contar cuantas cajas pasan por una cinta transportadora de una fábrica, esto se logra con un video en vivo dónde se cuentan cuantos

bounding box pasan de un lado a otro de una línea predefinida. Esto podría mejorarse aún más si se realiza un entrenamiento de YOLO con nuestras propias imágenes, para detectar no sólo cajas sino también clasificarlas según etiquetas, por ejemplo.

Para concluir, podemos decir que hay una gran cantidad de aplicaciones diferentes, y que, en nuestro caso, necesitamos indispensablemente realizar el procesamiento y tracking directamente en el borde, caso contrario, la gran latencia producida hace inviable el proyecto.

REFERENCIAS

Apuntes de la cátedra.

Geron Aurelien (2017), Hands-On machine learning with Scikit-Learn & TensorFlow: concepts, tools and techniques to build intelligent systems. O'Reilly Media – Sbastopol.

Jan Erik Solem (2012), Programming Computer Vision with Python, O'Reilly.

J. C. Russ (1994), The image processing Handbook – 2do ed., CRC Press.

Kike Arnaiz (2000), Fotografía desde cero, Independently published.

Mellado, Jose Maria (2017),Fotografía de alta calidad: los fundamentos de la fotografía – Adobe CC.

Open Source Computer Vision (s.f.), Template Matchingh, https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html

Raschka, Sebastian (2015),Python machine learning: unlock deeper insights into machine learning with this vital guide to culting-edge predictive analytics, Packt Publishing – Birmingham.

R. C. González – R. E. Woods (2008),Digital image processing, 3° Ed. Prentice Hall.

R. C. González – R. E. Woods (2018),Digital image processing, 4° Ed. Pearson Mx.

R. C. González – R. E. Woods (1996),Tratamiento digital de imágenes, 1°Ed. Addison Wesley.

Sharma, Aditya – Shirmall, Vishwesh Ravi – Beyeler, Michael (2019),Machine learning for OpenCV4 – 2nd ed., Packt Publishing – Birmingham.

Ultralytics (Nov 12 2023), YOLOv3, YOLOv3-Ultralytics y YOLOv3u, https:// docs.ultralytics.com/es/models/yolov3/

Ultralytics (Nov 12 2023), Ultralytics YOLOv8, https://docs.ultralytics.com/es/models/yolov8/#can-i-benchmark-yolov8-models-for-performance

Ultralytics (Sep 30 2024), Ultralytics YOLO11, https://docs.ultralytics.com/es/models/yolo11/