Clase 134 — YOLOv11 práctico: detección, segmentación, pose, tracking

Parte: 2 — Deep Learning · Fuente: Ultralytics YOLOv11 docs. Duración estimada: 85 min.

Clase 134 — YOLOv11 práctico: detección, segmentación, pose, tracking

Parte: 2 — Deep Learning · Fuente: Ultralytics YOLOv11 docs. Duración estimada: 85 min.

Objetivo

Dominar YOLOv11 (Ultralytics, 2024) — el detector default industrial 2026: inference real-time, fine-tuning sencillo, export a ONNX/TensorRT/CoreML/TFLite con un kwarg. Cubrir las 4 tareas: detection, segmentation, pose estimation, oriented bounding boxes (OBB).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Inferir con un modelo COCO-pretrained: YOLO('yolo11n.pt')(img).
- Fine-tunear con dataset propio en formato YOLO (txt por imagen).
- Aplicar las 4 tareas: detection (yolo11n.pt), segmentation (yolo11n-seg.pt), pose (yolo11n-pose.pt), OBB (yolo11n-obb.pt).
- Tracking de objetos en video con ByteTrack / BoT-SORT integrado.
- Exportar a ONNX/TensorRT para deploy.

Temas

- Modelos: n (nano), s, m, l, x. Trade-off speed vs accuracy.
- Formato YOLO de anotación: class cx cy w h (normalizado).
- dataset.yaml: paths + nombres de clases.
- Fine-tuning: model.train(data='dataset.yaml', epochs=100, imgsz=640).
- Modes: predict, val, train, export, benchmark.
- Tracking integrado.

Definiciones y características

- YOLO: clase Ultralytics que carga cualquier variante.
- results[0]: objeto con boxes, masks (seg), keypoints (pose), probs.
- OBB (Oriented Bounding Box): caja rotada — útil para aerial, documents.
- model.export(format='onnx'): 1 línea para ONNX/TensorRT/CoreML/TFLite.
- mAP@0.5: métrica standard, threshold IoU 0.5.

Dataset / recursos

- COCO pre-trained para inference.
- Roboflow Universe para dataset propio.
- Librerías: ultralytics (pip install ultralytics).

Ejercicios

1. Inference: model = YOLO('yolo11n.pt'); results = model('zidane.jpg'); results[0].show().
2. Segmentation: YOLO('yolo11n-seg.pt'). Visualizar máscaras.
3. Pose: YOLO('yolo11n-pose.pt'). Detectar keypoints en una foto.
4. Fine-tune: dataset propio (50-100 imágenes anotadas). model.train(data='ds.yaml', epochs=50, imgsz=640).
5. Tracking: model.track('video.mp4', tracker='botsort.yaml').

Homework verificable

Detector custom de 2 clases:

1. Etiquetar ~100 imágenes con Roboflow / LabelImg → formato YOLO.
2. dataset.yaml correcto.
3. Train YOLOv11n por 100 épocas.
4. Reportar mAP@0.5; visualizar 3 inferencias.
5. Exportar a ONNX y verificar que predice igual.

Criterio de aceptación: mAP@0.5 ≥ 0.7; ONNX produce predicciones consistentes.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
dataset.yaml paths mal	Errores de path. Fix: usar paths absolutos
mAP bajo con dataset chico	Pocos datos. Fix: ≥ 100 ejemplos por clase
Inference lento en CPU	YOLO es para GPU. Fix: device='cuda' o exp
Class names mismatched	Orden en yaml debe coincidir con anotacion
Anotaciones con coords absolutas	YOLO espera normalizadas [0,1]. Fix: conve

Preguntas frecuentes

YOLOv11 vs versiones anteriores?

v11 mejor accuracy/latency. v8 sigue siendo válido y muy usado. v5 legacy pero funciona.

Tareas además de detección?

Segmentation, pose estimation (COCO keypoints), classification, OBB. Una sola API.

Producción cómo deploy?

Export → ONNX → ONNX Runtime / TensorRT. O TF Lite para mobile. Soportado nativo.

Licencia?

AGPL para uso open-source. Para uso comercial cerrado, license enterprise.

Augmentation automática?

Sí, mosaic/mixup/etc. Built-in. Tunear con model.train(augment=True).

Referencias

- Ultralytics YOLOv11.
- Redmon et al. (2016), You Only Look Once — paper original.
- Roboflow Universe — datasets pre-anotados.

Siguiente clase

Clase 135 — RNNs: neuronas recurrentes, BPTT

Apéndice: notebook (primer bloque)

ultralytics descarga modelos ~6 MB (yolo11n) — funciona en CPU pero lento. Fallback: sliding-window + HOG features + SVM sklearn.

```
USE_YOLO = False
try:
    from ultralytics import YOLO
    USE_YOLO = True
    print('ultralytics disponible')
except Exception as e:
    print('YOLO no disponible. Fallback sliding-window HOG+SVM. Motivo:', type(e).__name__)

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
np.random.seed(42)
```

Archivos complementarios

- notebook.ipynb