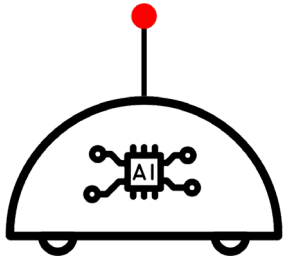


YOLO v8

Reconocimiento de objetos

Máster en Robótica y Sistemas Inteligentes

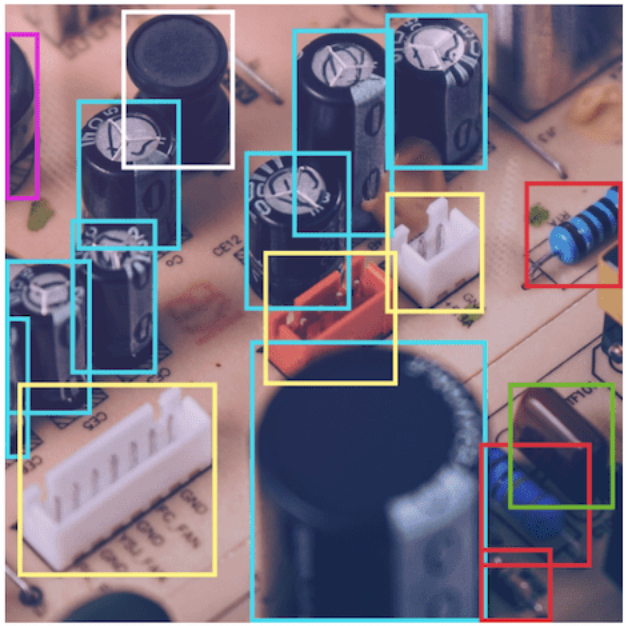
YOLO v8

Tasks

Classification

Capacitor

Object Detection



Capacitor, Resistor, Transformer, Connector, Inductor, Polyester Capacitor

Segmentation

Capacitor, Resistor, Transformer, Connector, Inductor, Polyester Capacitor

Image from https://user-images.githubusercontent.com/26833433/212094133-6bb8c21c-3d47-41df-a512-81c5931054ae.png

YOLO v8

Installation

Prerequisites: Python>=3.7 and PyTorch>=1.7

Create conda environment

```
conda create --name reob_env python=3.10
```

Conda activate environment

```
conda activate reob_env
```

Conda install Pytorch (https://pytorch.org/get-started/locally/)

```
conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia
```

Install yolov8

```
pip install ultralytics
```

Install jupyter notebook

```
pip install notebook
```

YOLO v8

Inference

In [1]:
```
from ultralytics import YOLO
import cv2
from matplotlib import pyplot as plt
import numpy as np
import random
```

```
C:\Users\virgi\miniconda3\envs\reob_env\lib\site-packages\tqdm\auto.py:22: TqdmWarning: IProgress not found. Please update jupy
ter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
# load a pretrained model yolov8x, yolov8x-seg(with segmentation), yolov8x-cls(with classification)
model = YOLO("yolov8n-seg.pt")
```

▼ Detection

See Detection Docs for usage examples with these models.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

- mAP^{val} values are for single-model single-scale on COCO val2017 dataset. Reproduce by `yolo val detect data=coco.yaml device=0`
- Speed averaged over COCO val images using an Amazon EC2 P4d instance. Reproduce by `yolo val detect data=coco128.yaml batch=1 device=0/cpu`

▼ Segmentation

See Segmentation Docs for usage examples with these models.

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

- mAP^{val} values are for single-model single-scale on COCO val2017 dataset. Reproduce by `yolo val segment data=coco.yaml device=0`
- Speed averaged over COCO val images using an Amazon EC2 P4d instance. Reproduce by `yolo val segment data=coco128-seg.yaml batch=1 device=0/cpu`

▼ Classification

See Classification Docs for usage examples with these models.

Model	size (pixels)	acc top1	acc top5	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B) at 640
YOLOv8n-cls	224	66.6	87.0	12.9	0.31	2.7	4.3
YOLOv8s-cls	224	72.3	91.1	23.4	0.35	6.4	13.5
YOLOv8m-cls	224	76.4	93.2	85.4	0.62	17.0	42.7
YOLOv8l-cls	224	78.0	94.1	163.0	0.87	37.5	99.7
YOLOv8x-cls	224	78.4	94.3	232.0	1.01	57.4	154.8

- acc values are model accuracies on the ImageNet dataset validation set. Reproduce by `yolo val classify data=path/to/ImageNet device=0`
- Speed averaged over ImageNet val images using an Amazon EC2 P4d instance. Reproduce by `yolo val classify data=path/to/ImageNet batch=1 device=0/cpu`

YOLO v8

Inference

In [3]:
```
1 img = cv2.imread('bus.jpg') #"https://ultralytics.com/images/bus.jpg"
2 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [4]:
```
1 plt.imshow(img)
```

Out[4]:
```
<matplotlib.figure.AxesImage at 0x1b7f7d66d10>
```

In [5]:
```
1 r = model.predict(img)
2 # r = model(img)
```

```
Ultralytics YOLOv8.0.39  Python-3.10.9 torch-1.13.1 CUDA:0 (NVIDIA GeForce RTX 2060, 6144MiB)
YOLOv8n-seg summary (fused): 195 layers, 3404320 parameters, 0 gradients, 12.6 GFLOPs

0: 640x480 4 persons, 1 bus, 1 skateboard, 24.1ms
```

YOLO v8

Inference

In [6]:

```python
names = model.names
colors = [tuple([random.randint(0, 255) for _ in range(3)]) for _ in range(len(names))]
roi = np.zeros(img.shape).astype(np.int32)
for box, mask in zip(r[0].boxes, r[0].masks):
    # Bounding box & Category
    bbox = box.xyxy.cpu().numpy()[0].astype(np.int32)
    category = int(box.cls.cpu().numpy()[0])
    img = cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (colors[category]), 5)
    label = names[category] +' '+str(box.conf.cpu().numpy()[0])
    (w, h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
    # Prints the text.
    img = cv2.rectangle(img, (bbox[0], bbox[1] - 20), (bbox[0] + w, bbox[1]), colors[category], -1)
    img = cv2.putText(img, label, (bbox[0], bbox[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,0), 1)
    # Mask
    m = mask.data.cpu().numpy()
    resized = cv2.resize(m, (img.shape[1], img.shape[0]), interpolation = cv2.INTER_AREA)
    contours, _ = cv2.findContours(np.array(resized, np.uint8), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnt = []
    area = 0
    for c in contours:
        a = cv2.contourArea(c)
        if a > area:
            cnt = c
            area = a
    roi = cv2.drawContours(roi, [cnt], -1, colors[category], -1)
blended = ((0.6 * img) + (0.4 * roi)).astype("uint8")
plt.imshow(blended)
plt.show()
```

Prediction contains a list of bounding boxes and masks
REMEMBER: it is a segmentation model

More info in https://docs.ultralytics.com/modes/predict/#boxes

YOLO v8

Inference

In [6]:

```python
names = model.names
colors = [tuple([random.randint(0, 255) for _ in range(3)]) for _ in range(len(names))]
roi = np.zeros(img.shape).astype(np.int32)
for box, mask in zip(r[0].boxes, r[0].masks):
    # Bounding box & Category
    bbox = box.xyxy.cpu().numpy()[0].astype(np.int32)
    category = int(box.cls.cpu().numpy()[0])
    img = cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (colors[category]), 5)
    label = names[category] +' '+str(box.conf.cpu().numpy()[0])
    (w, h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
    # Prints the text.
    img = cv2.rectangle(img, (bbox[0], bbox[1] - 20), (bbox[0] + w, bbox[1]), colors[category], -1)
    img = cv2.putText(img, label, (bbox[0], bbox[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,0), 1)
    # Mask
    m = mask.data.cpu().numpy()
    resized = cv2.resize(m, (img.shape[1], img.shape[0]), interpolation = cv2.INTER_AREA)
    contours, _ = cv2.findContours(np.array(resized, np.uint8), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnt = []
    area = 0
    for c in contours:
        a = cv2.contourArea(c)
        if a > area:
            cnt = c
            area = a
    roi = cv2.drawContours(roi, [cnt], -1, colors[category], -1)
blended = ((0.6 * img) + (0.4 * roi)).astype("uint8")
plt.imshow(blended)
plt.show()
```

Bounding box attributes (box)
xyxy: returns two extremes of the bounding box
cls: class of the prediction
conf: probability of the prediction

YOLO v8

Inference

In [6]:

```python
names = model.names
colors = [tuple([random.randint(0, 255) for _ in range(3)]) for _ in range(len(names))]
roi = np.zeros(img.shape).astype(np.int32)
for box, mask in zip(r[0].boxes, r[0].masks):
    # Bounding box & Category
    bbox = box.xyxy.cpu().numpy()[0].astype(np.int32)
    category = int(box.cls.cpu().numpy()[0])
    img = cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (colors[category]), 5)
    label = names[category] +' '+str(box.conf.cpu().numpy()[0])
    (w, h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
    # Prints the text.
    img = cv2.rectangle(img, (bbox[0], bbox[1] - 20), (bbox[0] + w, bbox[1]), colors[category], -1)
    img = cv2.putText(img, label, (bbox[0], bbox[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,0), 1)
    # Mask
    m = mask.data.cpu().numpy()
    resized = cv2.resize(m, (img.shape[1], img.shape[0]), interpolat
    contours, _ = cv2.findContours(np.array(resized, np.uint8), cv2.
    cnt = []
    area = 0
    for c in contours:
        a = cv2.contourArea(c)
        if a > area:
            cnt = c
            area = a
    roi = cv2.drawContours(roi, [cnt], -1, colors[category], -1)
blended = ((0.6 * img) + (0.4 * roi)).astype("uint8")
plt.imshow(blended)
plt.show()
```

Bounding box attributes (box)
xyxy: returns two extremes of the bounding box
cls: class of the prediction
conf: probability of the prediction

Bounding box attributes (box)

```
box.xyxy  # box with xyxy format, (N, 4)
box.xywh  # box with xywh format, (N, 4)
box.xyxyn  # box with xyxy format but normalized, (N, 4)
box.xywhn  # box with xywh format but normalized, (N, 4)
box.conf  # confidence score, (N, 1)
box.cls  # cls, (N, 1)
box.data  # raw bboxes tensor, (N, 6) or boxes.boxes .
```

YOLO v8

Inference

In [6]:

```python
names = model.names
colors = [tuple([random.randint(0, 255) for _ in range(3)]) for _ in range(len(names))]
roi = np.zeros(img.shape).astype(np.int32)
for box, mask in zip(r[0].boxes, r[0].masks):
    # Bounding box & Category
    bbox = box.xyxy.cpu().numpy()[0].astype(np.int32)
    category = int(box.cls.cpu().numpy()[0])
    img = cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (colors[category]), 5)
    label = names[category] +' '+str(box.conf.cpu().numpy()[0])
    (w, h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
    # Prints the text.
    img = cv2.rectangle(img, (bbox[0], bbox[1] - 20), (bbox[0] + w, bbox[1]), colors[category], -1)
    img = cv2.putText(img, label, (bbox[0], bbox[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,0), 1)
    # Mask
    m = mask.data.cpu().numpy()
    resized = cv2.resize(m, (img.shape[1], img.shape[0]), interpolation = cv2.INTER_AREA)
    contours, _ = cv2.findContours(np.array(resized, np.uint8), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnt = []
    area = 0
    for c in contours:
        a = cv2.contourArea(c)
        if a > area:
            cnt = c
            area = a
    roi = cv2.drawContours(roi, [cnt], -1, colors[category], -1)
blended = ((0.6 * img) + (0.4 * roi)).astype("uint8")
plt.imshow(blended)
plt.show()
```

Mask attributes (mask)
data: mask of the object

YOLO v8

Train

In [1]:
```
1 from ultralytics import YOLO
```

In [2]:
```
1 model = YOLO("yolov8n-seg.pt")
```

In [*]:
```
1 model.train(data="open_data/dataset.yaml", epochs=10, imgsz=640, name='test1') #image size to be divisible by 32
```

```
Ultralytics YOLOv8.0.39  Python-3.10.9 torch-1.13.1 CUDA:0 (NVIDIA GeForce RTX 2060, 6144MiB)
yolo\engine\trainer: task=segment, mode=train, model=yolov8n-seg.pt, data=open_data/dataset.yaml, epochs=10, patience=50, bat
                                                                                                roject=None, name=test1, exist_ok=False, p
                                                                                                s=False, image_weights=False, rect=False,
                                                                                                ask_ratio=4, dropout=False, val=True, spli
                                                                                                alse, dnn=False, plots=True, source=ultral
                                                                                                abels=False, hide_conf=False, vid_stride=
                                                                                                ne, retina_masks=False, boxes=True, format
                                                                                                e, opset=None, workspace=4, nms=False, lr0
                                                                                                ntum=0.8, warmup_bias_lr=0.1, box=7.5, cls
                                                                                                sv_v=0.4, degrees=0.0, translate=0.1, scal
                                                                                                opy_paste=0.0, cfg=None, v5loader=False, t



                                                                                                rguments
                                                                                                [3, 16, 3, 2]
                                                                                                [16, 32, 3, 2]
                                                                                                [32, 32, 1, True]
```

Key	Value	Description
model	None	path to model file, i.e. yolov8n.pt, yolov8n.yaml
data	None	path to data file, i.e. coco128.yaml
epochs	100	number of epochs to train for
batch	16	number of images per batch (-1 for AutoBatch)
imgsz	640	size of input images as integer or w,h
save	True	save train checkpoints and predict results
save_period	-1	Save checkpoint every x epochs (disabled if < 1)
cache	False	True/ram, disk or False. Use cache for data loading
device	None	device to run on, i.e. cuda device=0 or device=0,1,2,3 or device=cpu
workers	8	number of worker threads for data loading (per RANK if DDP)
name	None	experiment name
pretrained	False	whether to use a pretrained model
optimizer	'SGD'	optimizer to use, choices=['SGD', 'Adam', 'AdamW', 'RMSProp']
verbose	False	whether to print verbose output
lr0	0.01	initial learning rate (i.e. SGD=1E-2, Adam=1E-3)
lrf	0.01	final learning rate (lr0 * lrf)

YOLO v8

Train

```
In [1]: 1 from ultralytics import YOLO
```

```
In [2]: 1 model = YOLO("yolov8n-seg.pt")
```

```
In [*]: 1 model.train(data="open_data/dataset.yaml", epochs=10, imgsz=640, name='test1') #image size to be divisible by 32
```

```
Ultralytics YOLOv8.0.39  Python-3.10.9 torch-1.13.1 CUDA:0 (NVIDIA GeForce RTX 2060, 6144MiB)
yolo\engine\trainer: task=segment, mode=train, model=yolov8n-seg.pt, data=open_data/dataset.yaml, epochs=10, patience=50, bat
ch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=test1, exist_ok=False, p
retrained=False, optimizer=SGD, verbose=True, seed=0, deterministic=True, single_cls=False, image_weights=False, rect=False,
ue, mask_ratio=4, dropout=False, val=True, spli
alf=False, dnn=False, plots=True, source=ultral
ide_labels=False, hide_conf=False, vid_stride=
es=None, retina_masks=False, boxes=True, format
=False, opset=None, workspace=4, nms=False, lr0
momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls
.7, hsv_v=0.4, degrees=0.0, translate=0.1, scal
.0, copy_paste=0.0, cfg=None, v5loader=False, t
```

Key	Value	Description
momentum	0.937	SGD momentum/Adam beta1
weight_decay	0.0005	optimizer weight decay 5e-4
warmup_epochs	3.0	warmup epochs (fractions ok)
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	7.5	box loss gain
cls	0.5	cls loss gain (scale with pixels)
dfl	1.5	dfl loss gain
fl_gamma	0.0	focal loss gamma (efficientDet default gamma=1.5)
label_smoothing	0.0	label smoothing (fraction)
nbs	64	nominal batch size
overlap_mask	True	masks should overlap during training (segment train only)
mask_ratio	4	mask downsample ratio (segment train only)
dropout	0.0	use dropout regularization (classify train only)
val	True	validate/test during training

```
arguments
[3, 16, 3, 2]
[16, 32, 3, 2]
[32, 32, 1, True]
```

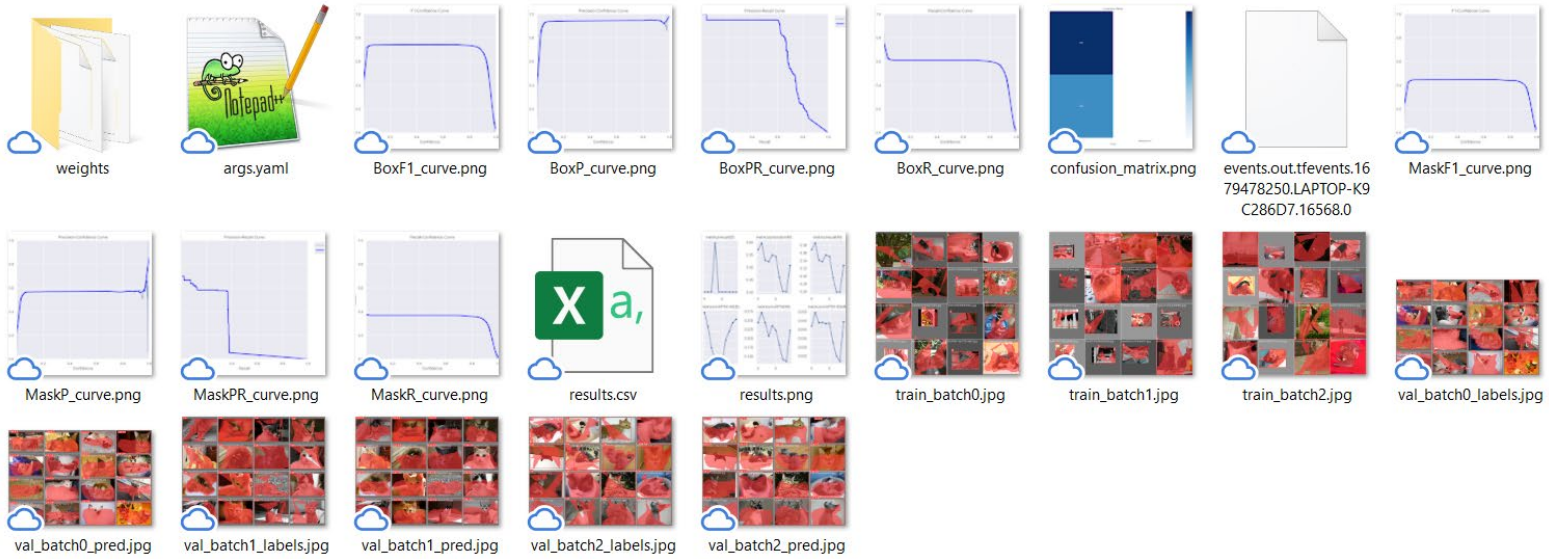
YOLO v8

Train

runs

segment

test1 *(name)*



YOLO v8

Export

```
1 model.export(format='onnx')
```

```
Ultralytics YOLOv8.0.39  Python-3.10.9 torch-1.13.1 CPU
YOLOv8n-seg summary (fused): 195 layers, 3258259 parameters, 0 gradients, 12.0 GFLOPs

PyTorch: starting from runs\segment\test1\weights\best.pt with input shape (1, 3, 640, 640) BCHW and output shape(s) ((1, 37, 8
400), (1, 32, 160, 160)) (6.4 MB)
requirements: YOLOv8 requirement "onnx>=1.12.0" not found, attempting AutoUpdate...
Requirement already satisfied: onnx>=1.12.0 in c:\users\virgi\miniconda3\envs\reob_env\lib\site-packages (1.13.1)
Requirement already satisfied: protobuf<4,>=3.20.2 in c:\users\virgi\miniconda3\envs\reob_env\lib\site-packages (from onnx>=1.1
2.0) (3.20.3)
Requirement already satisfied: typing-extensions>=3.6.2.1 in c:\users\virgi\miniconda3\envs\reob_env\lib\site-packages (from on
nx>=1.12.0) (4.4.0)
Requirement already satisfied: numpy>=1.16.6 in c:\users\virgi\miniconda3\envs\reob_env\lib\site-packages (from onnx>=1.12.0)
(1.23.5)

requirements: 1 package updated per ['onnx>=1.1
requirements:  Restart runtime or rerun command


ONNX: starting export with onnx 1.13.1...
ONNX: export success  4.3s, saved as runs\segme

Export complete (4.8s)
Results saved to G:\Mi unidad\DOCENCIA\MARSI\re
Predict:         yolo task=segment mode=predict
Validate:        yolo task=segment mode=val mod
Visualize:       https://netron.app
```

```
'runs\\segment\\test1\\weights\\best.onnx'
```

Key	Value	Description
format	'torchscript'	format to export to
imgsz	640	image size as scalar or (h, w) list, i.e. (640, 480)
keras	False	use Keras for TF SavedModel export
optimize	False	TorchScript: optimize for mobile
half	False	FP16 quantization
int8	False	INT8 quantization
dynamic	False	ONNX/TF/TensorRT: dynamic axes
simplify	False	ONNX: simplify model
opset	None	ONNX: opset version (optional, defaults to latest)
workspace	4	TensorRT: workspace size (GB)
nms	False	CoreML: add NMS

Export information in https://docs.ultralytics.com/modes/export/

YOLO v8

Export

Format	`format` Argument	Model	Metadata
PyTorch	-	`yolov8n.pt`	✅
TorchScript	`torchscript`	`yolov8n.torchscript`	✅
ONNX	`onnx`	`yolov8n.onnx`	✅
OpenVINO	`openvino`	`yolov8n_openvino_model/`	✅
TensorRT	`engine`	`yolov8n.engine`	✅
CoreML	`coreml`	`yolov8n.mlmodel`	✅
TF SavedModel	`saved_model`	`yolov8n_saved_model/`	✅
TF GraphDef	`pb`	`yolov8n.pb`	❌
TF Lite	`tflite`	`yolov8n.tflite`	✅
TF Edge TPU	`edgetpu`	`yolov8n_edgetpu.tflite`	✅
TF.js	`tfjs`	`yolov8n_web_model/`	✅
PaddlePaddle	`paddle`	`yolov8n_paddle_model/`	✅

Export information in https://docs.ultralytics.com/modes/export/