OpenVINO DEVCON 中国
系列工作坊 2023

基于 OpenVINO™ 加速构建 YOLOv8 部署方案

武卓
英特尔 AI 软件布道师

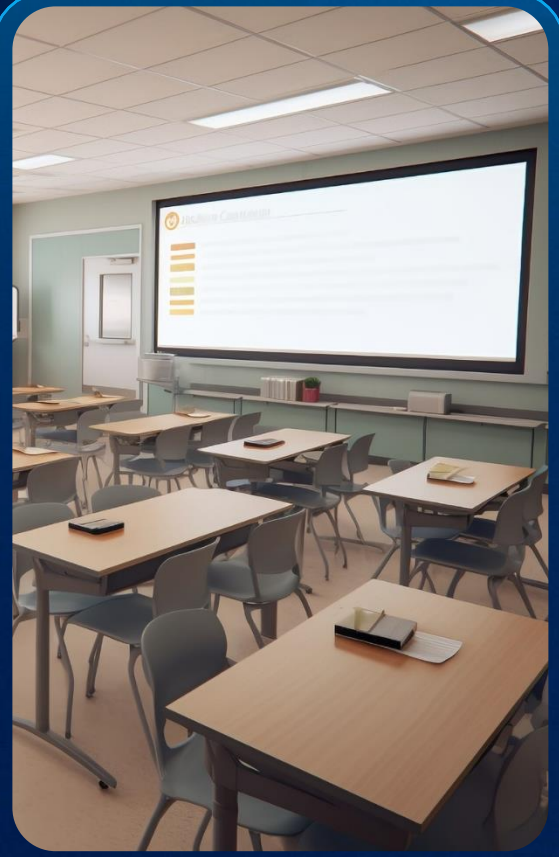
贾志刚
千瞳智能科技(苏州)有限公司 高级算法工程师

YOLOv8 介绍

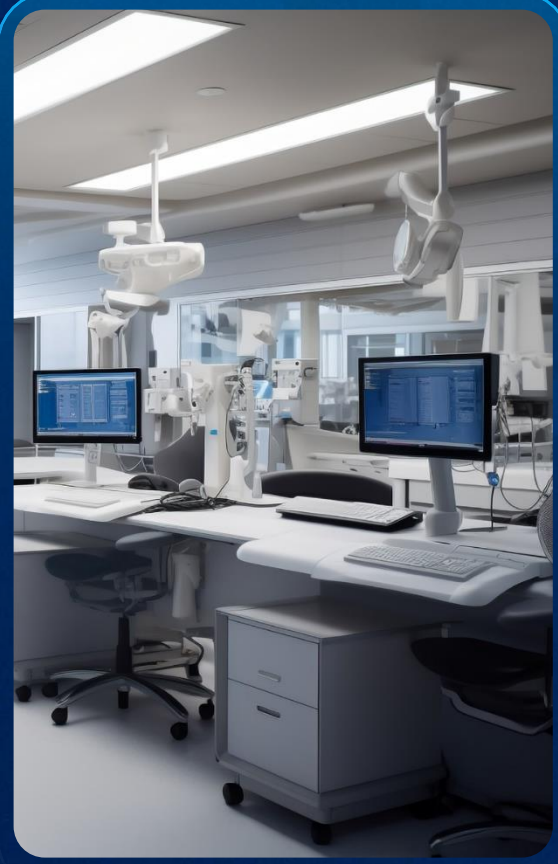
YOLO + OpenVINO™ 应用场景

工业

零售

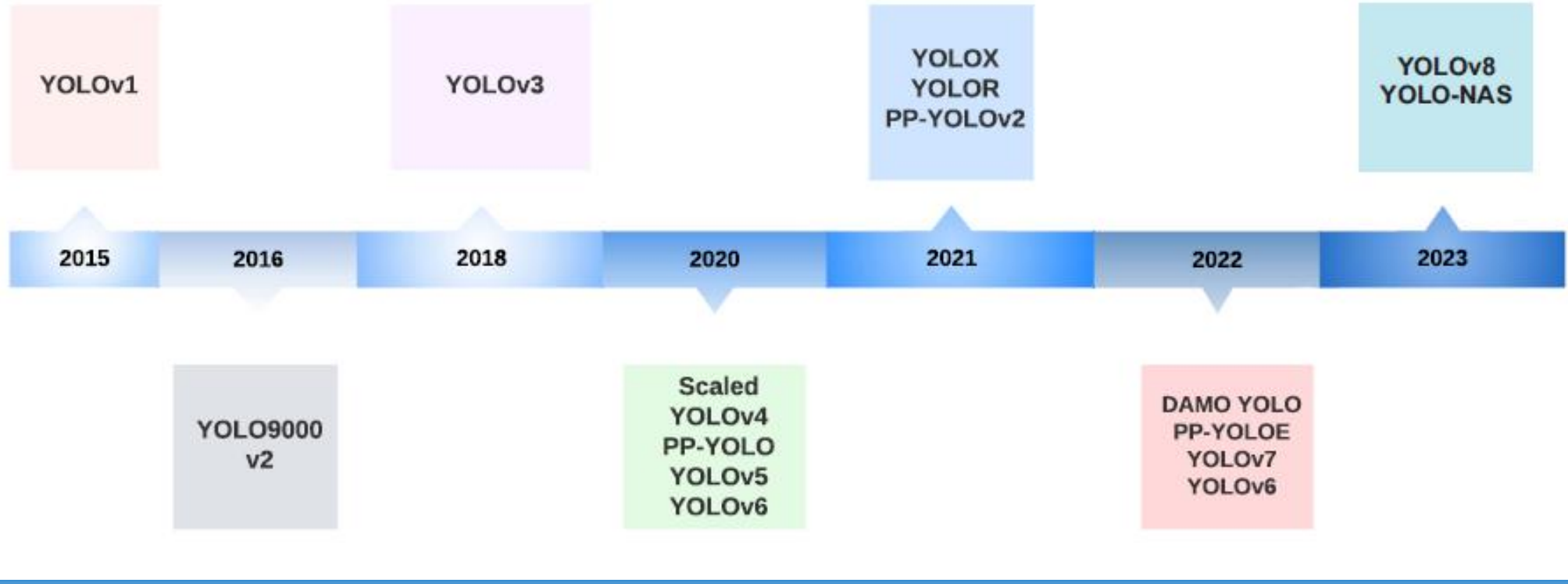


教育

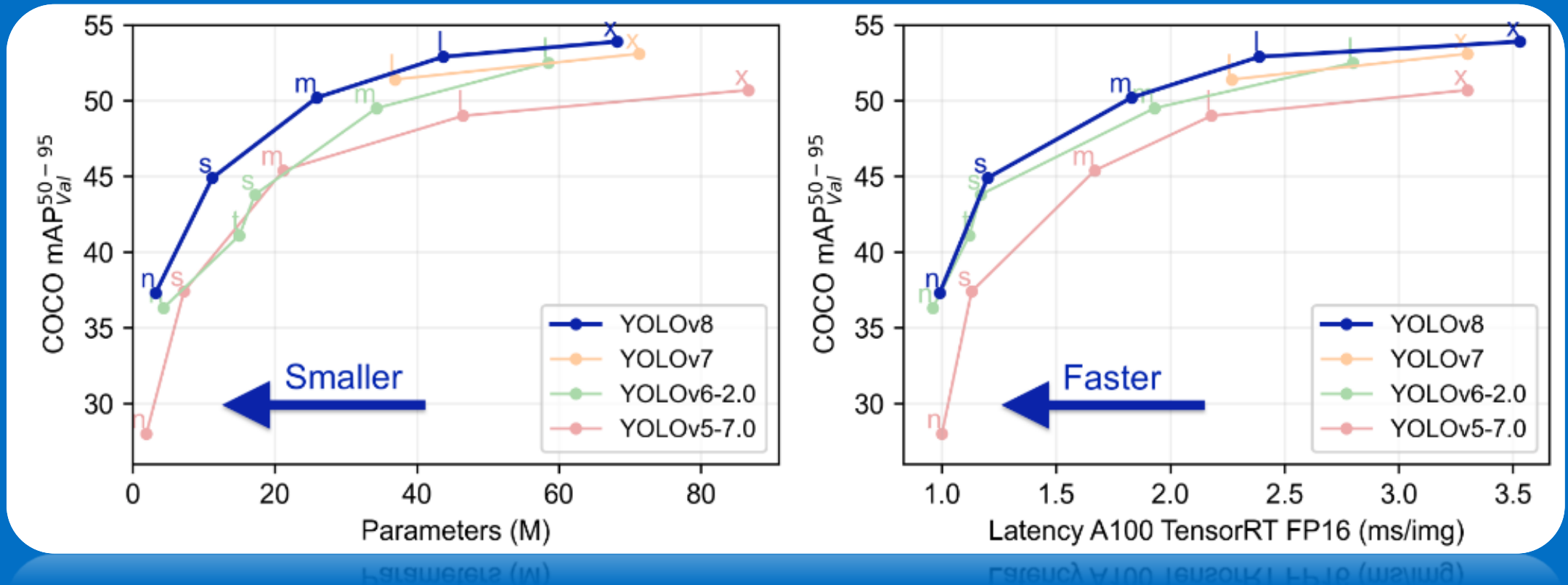


医疗

YOLO 系列网络发展历史 - OpenVINO™ 长期支持



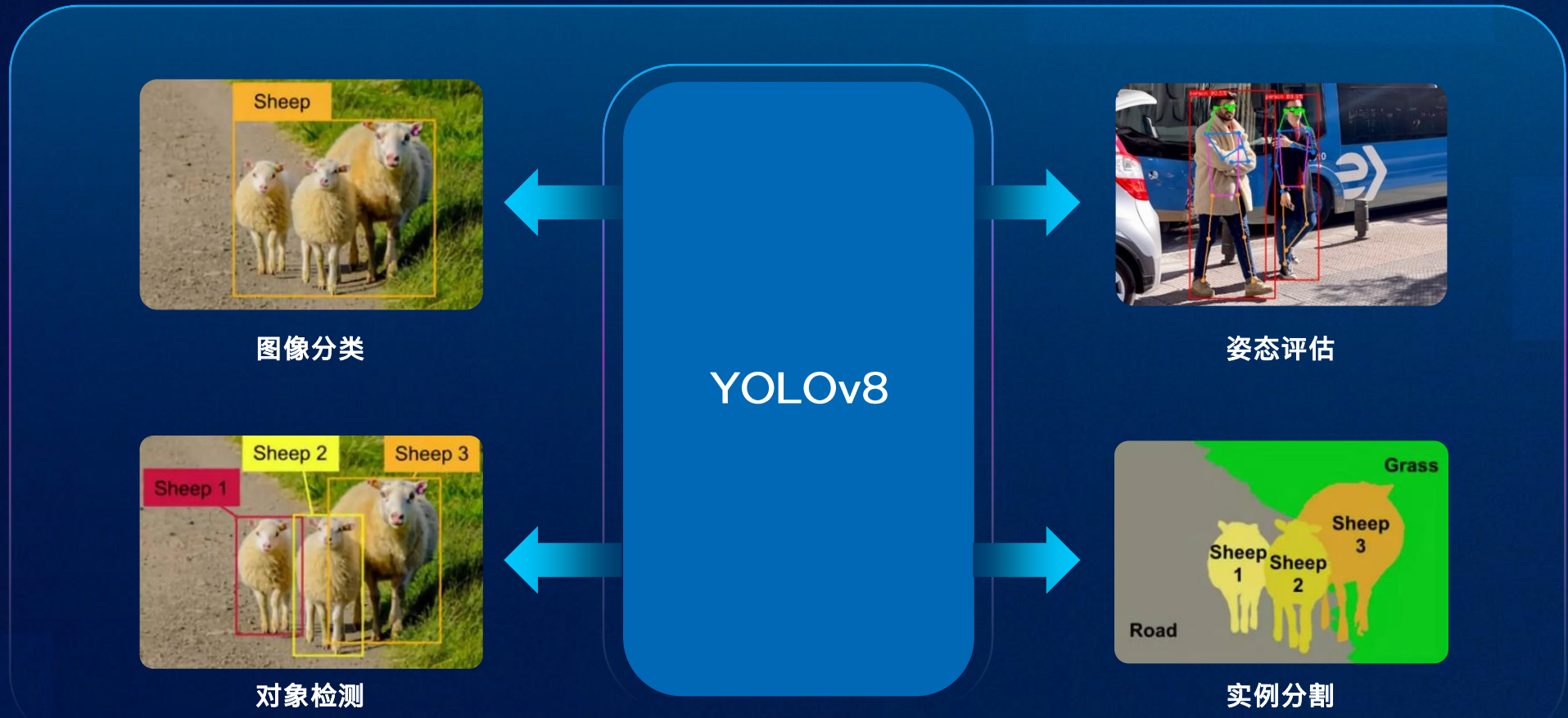
YOLOv5 ~ YOLOv8 系列模型性能对比



YOLOv8 不同大小模型性能对比

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

YOLOv8 视觉任务支持演化



YOLOv8 网络架构

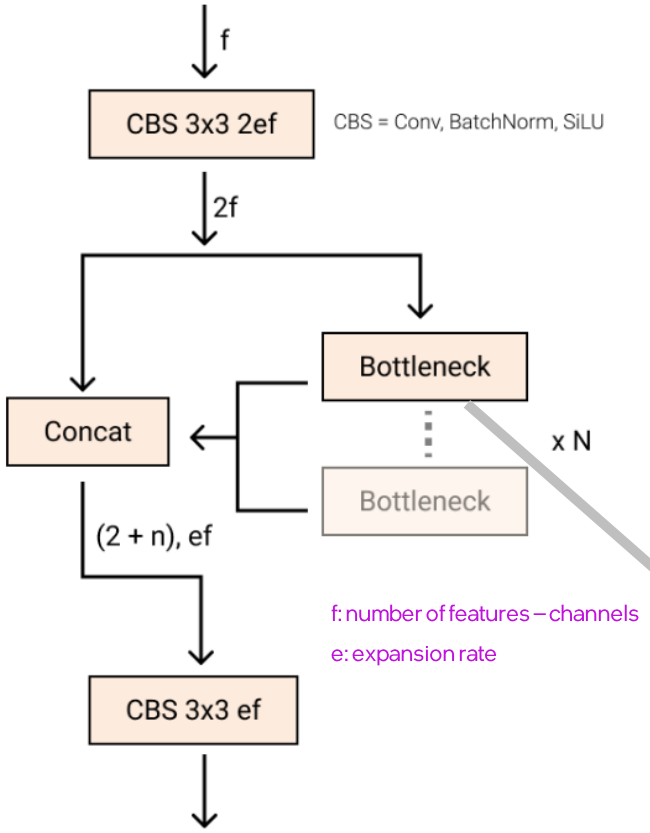
```yaml
# YOLOv8.0n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]]  # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]]  # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]]  # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]]  # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]]  # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]]  # cat backbone P4
  - [-1, 3, C2f, [512]]  # 12

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]]  # cat backbone P3
  - [-1, 3, C2f, [256]]  # 15 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]]  # cat head P4
  - [-1, 3, C2f, [512]]  # 18 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]]  # cat head P5
  - [-1, 3, C2f, [1024]]  # 21 (P5/32-large)

  - [[15, 18, 21], 1, Detect, [nc]]  # Detect(P3, P4, P5)
```

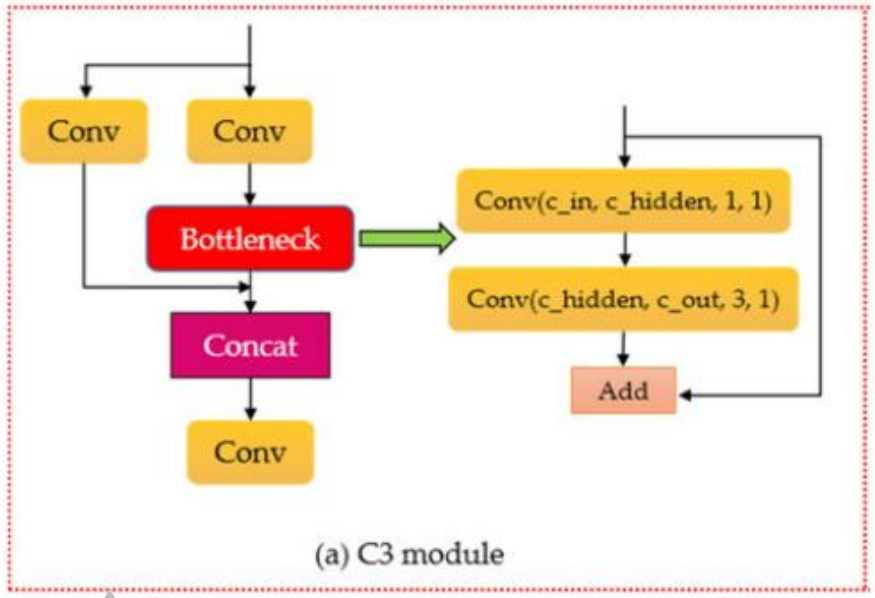
模型结构： backbone+neck+head

关键改进：

- 从 C3 到 C2F 模块
- CIOU 与 DFL 损失
- 解耦头与 anchor-free 的预测方式

YOLOv8 网络架构 – C3 与 C2f



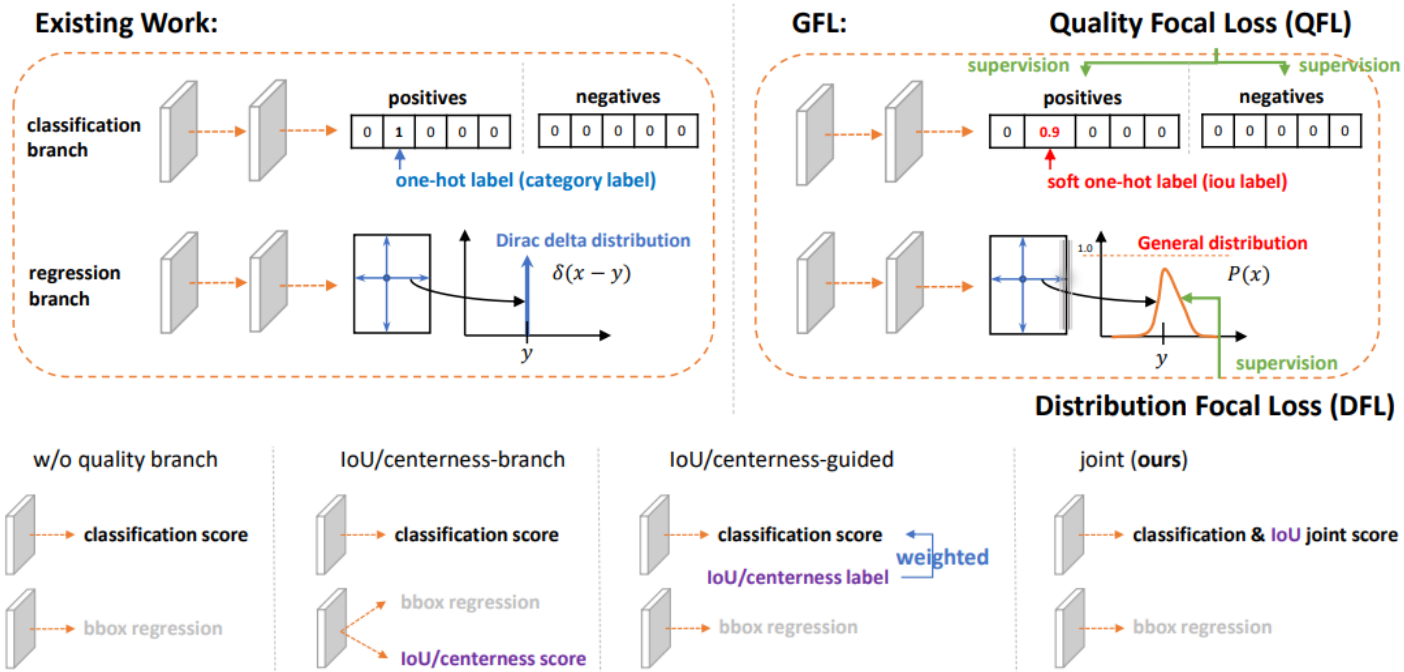


Bottleneck

CIOU 与 DFL

$$\mathbf{DFL}(\mathcal{S}_i, \mathcal{S}_{i+1}) = -\big((y_{i+1} - y)\log(\mathcal{S}_i) + (y - y_i)\log(\mathcal{S}_{i+1})\big).$$

$$\hat{y} = \sum_{j=0}^{n} P(y_j)y_j = \mathcal{S}_i y_i + \mathcal{S}_{i+1} y_{i+1} = \frac{y_{i+1} - y}{y_{i+1} - y_i} y_i + \frac{y - y_i}{y_{i+1} - y_i} y_{i+1}$$

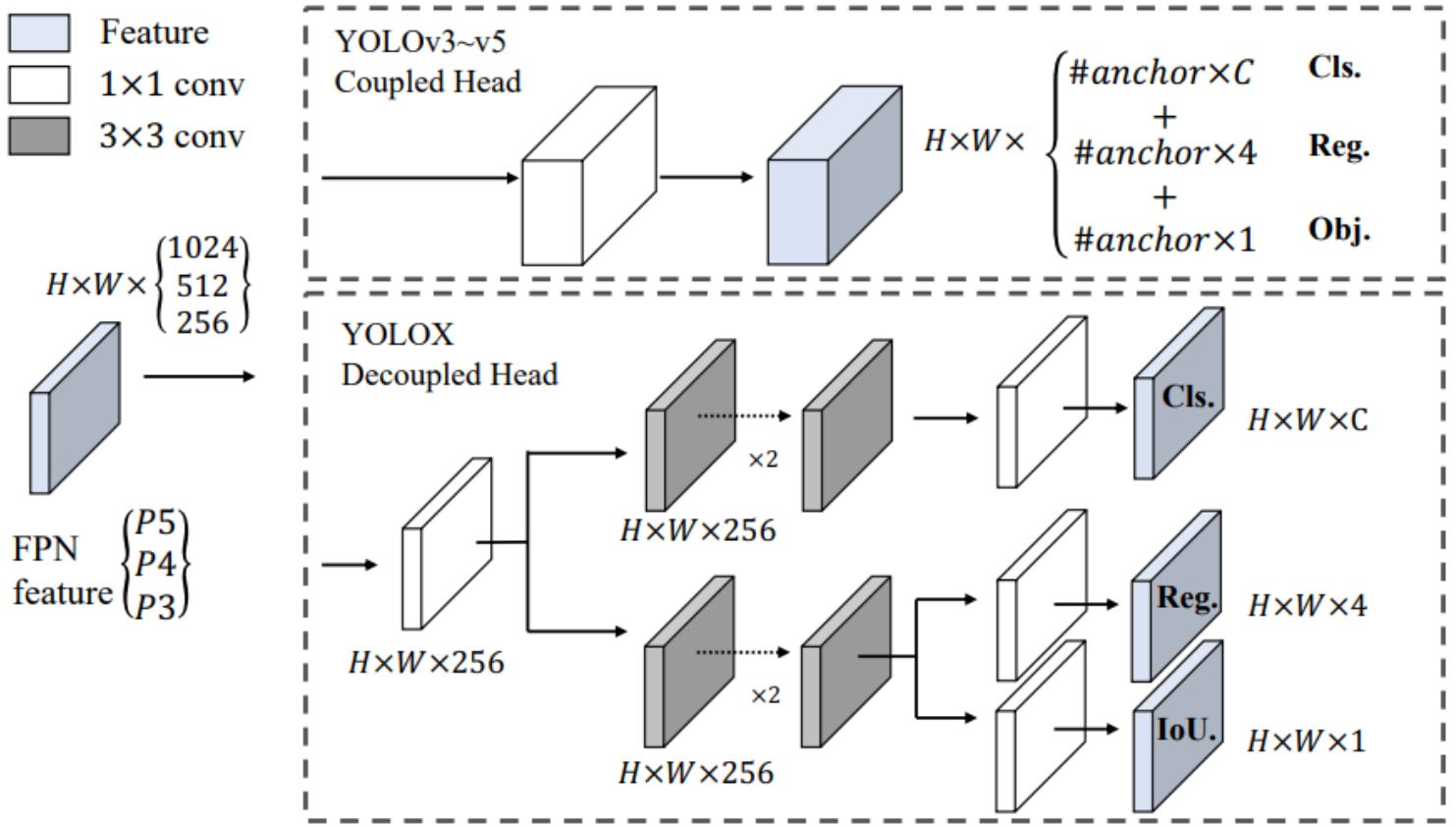


YOLOv5 损失：BCE + FL

YOLOv8 位置预测损失：Box IOU + DFL

好处：分布式感知+并交比得分损失更加合理，训练效果更好。

解耦头与 anchor-free 的预测方式



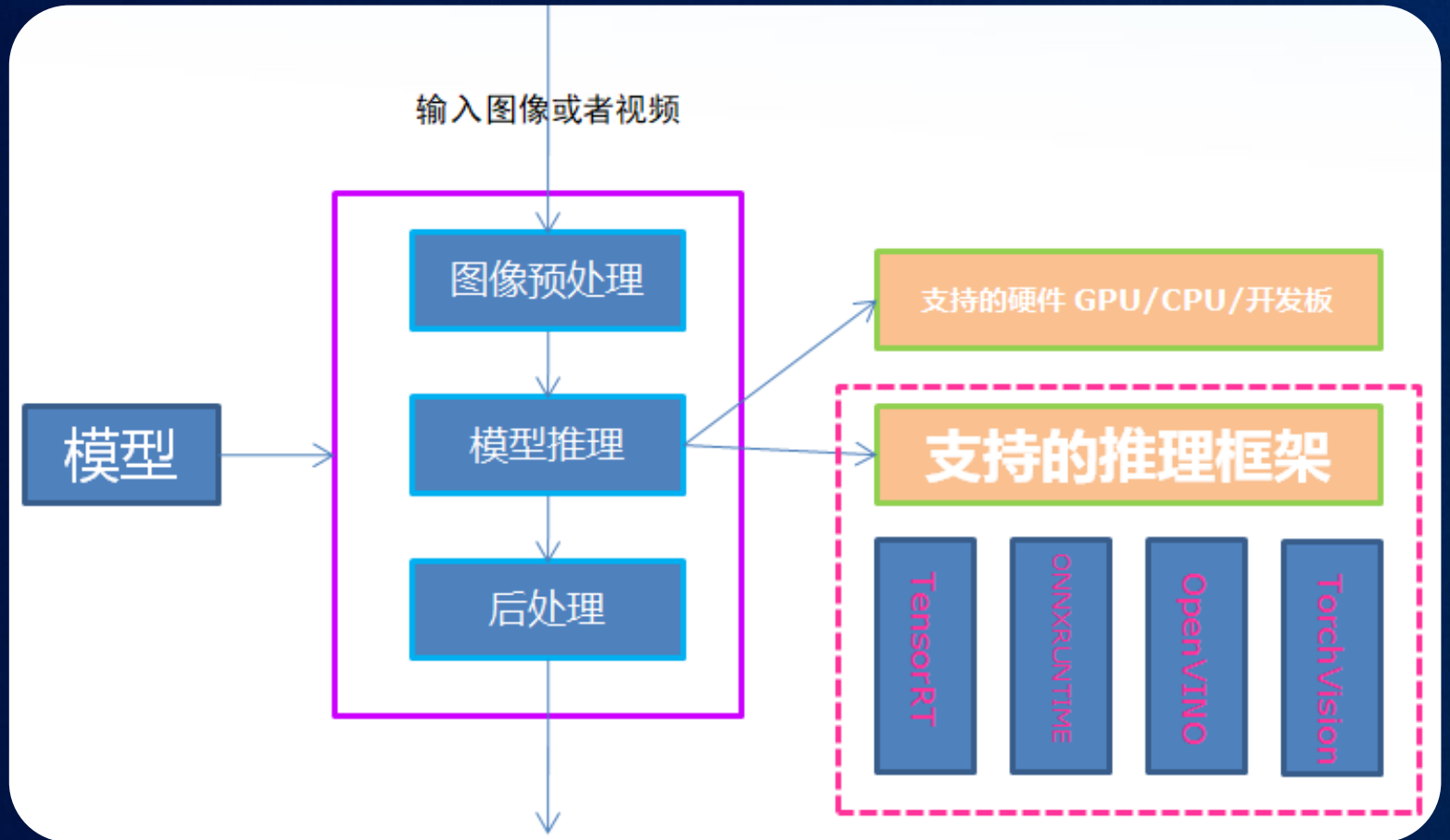
YOLOv5 之前的预测方式 – 把分类与位置回归合并在一起预测

YOLOv8 改进 分开预测， 学习 YOLOX 的

好处：采用直接的位置回归之后预测，精度明显提升，YOLOv8 去掉 conf 预测，直接预测四个位置。

YOLOv8 部署

YOLOv8 模型部署

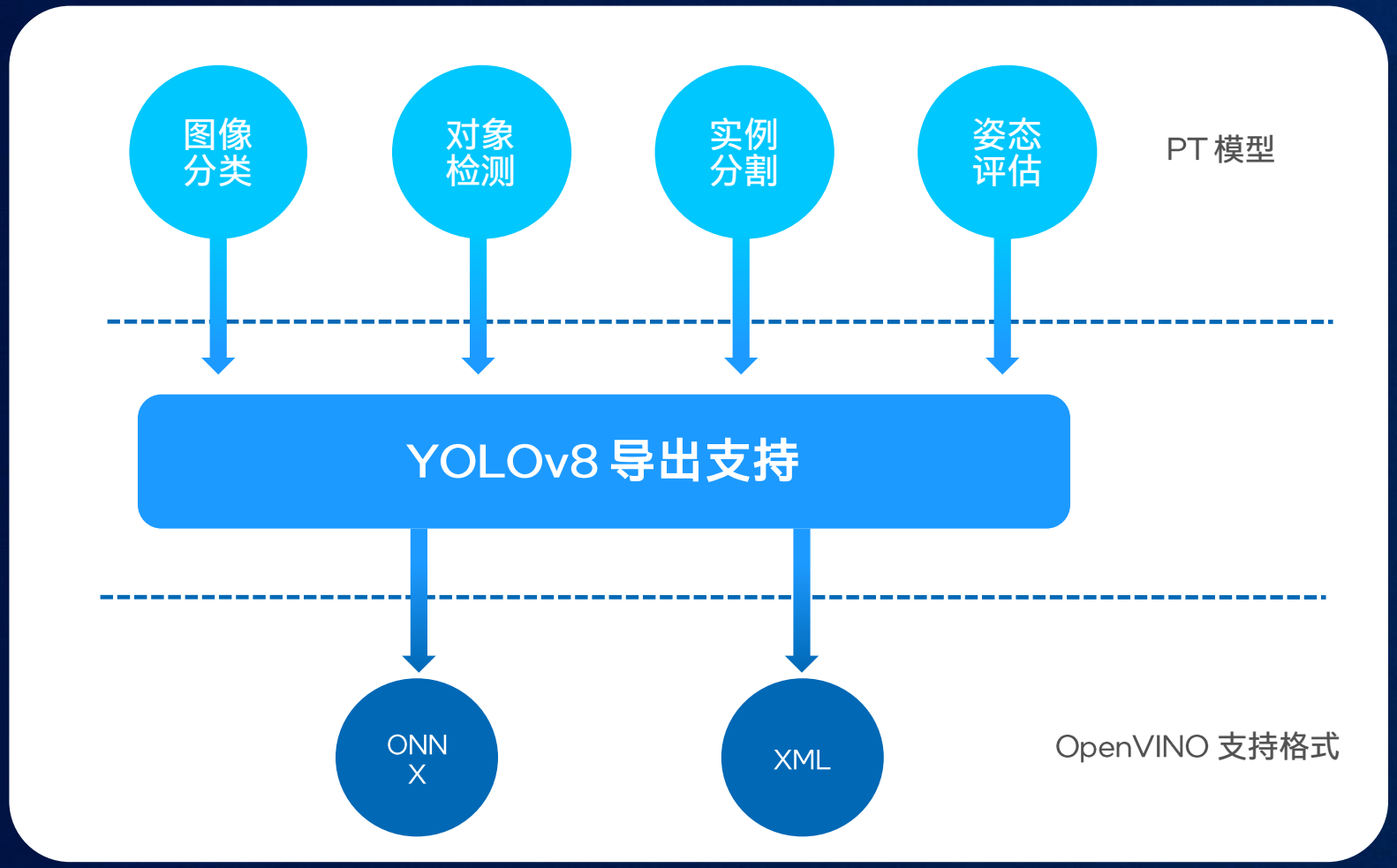


YOLOv8 支持导出各种主流格式的模型文件包括:

ONNX、XML、engine、ts等，根据支持的 CPU、GPU、ARM硬件不同可以选择不同的导出模型格式

好处：基于 YOLOv8 框架的命令行支持直接调用导出的模型文件运行推理，快速部署验证。

YOLOv8 支持 OpenVINO™ 部署



OpenVINO™ 支持全系列的 YOLO 模型部署。从 YOLOv1~YOLOv8

好处：可以在 CPU 端获得 YOLO 系列模型推理的最佳性能

YOLOv8 命令行使用

对比之前的 YOLO 系列版本，YOLOv8 在工程化与易用性方面大大提升，支持 CLI 命令行。

推理：
yolo predict model=yolov8n.pt source='https://ultralytics.com/images/bus.jpg'

训练：
yolo train data=your_dataset.yaml model=yolov8n.pt epochs=10 lr0=0.01

导出部署：
yolo export model=yolov8n.pt format=onnx imgsz=640

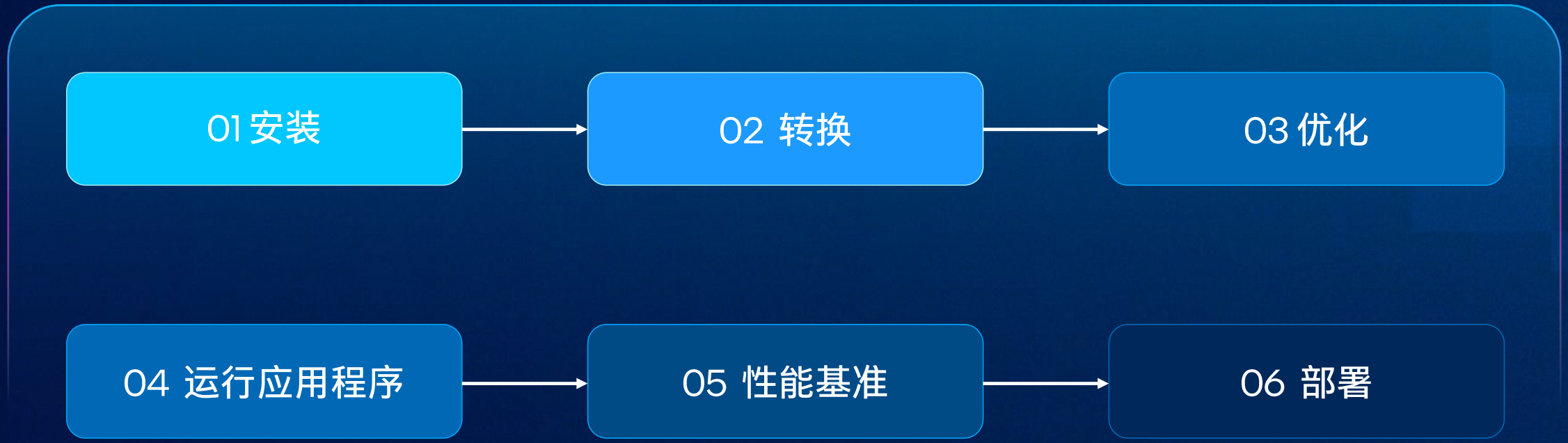
设置：
yolo settings
yolo settings runs_dir='/path/to/runs' tensorboard=False

好处：零代码实现 YOLOv8 模型的训练、导出、推理，全部一条命令行搞定。

解决方案

如何开始

逐步学习和构建



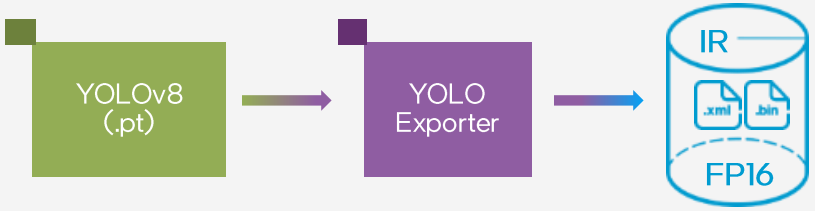
01 安装

```
# Step 1: Clone the Repository
$ git clone -b recipes https://github.com/openvinotoolkit/openvino_notebooks.git
$ cd openvino_notebooks/recipes/intelligent_queue_management

# Step 2: Create the Environment
$ python3 -m venv venv
$ source venv/bin/activate

# Step 3: Install Requirements
$ python -m pip install --upgrade pip
$ pip install -r requirements.txt
```

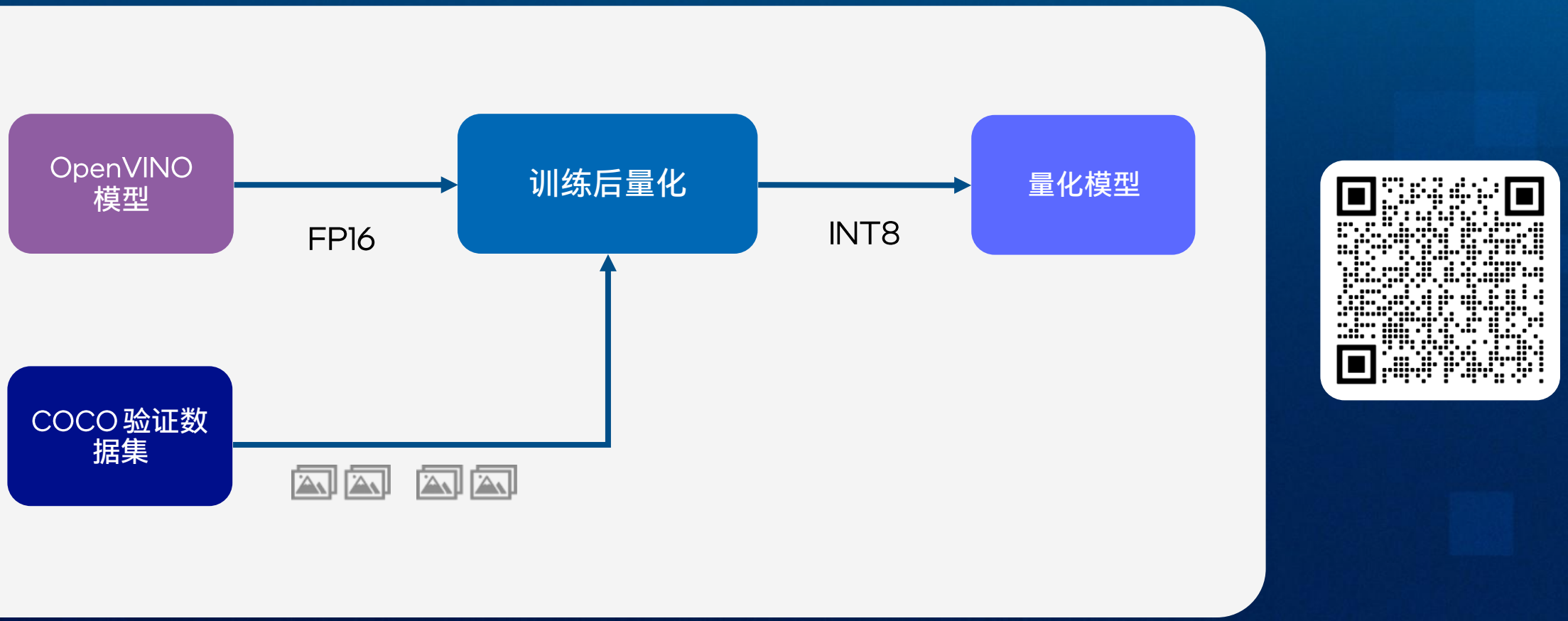
02 将 YOLOv8 转换为 OpenVINO™

02 Convert YOLOv8 to OpenVINO™

```python
from ultralytics import YOLO

# the name of the model we want to use
DET_MODEL_NAME = "yolov8m"

# create a model
det_model = YOLO(f"model/{DET_MODEL_NAME}.pt")

# export model to OpenVINO format
out_dir = det_model.export(format="openvino", dynamic=False, half=True)
```



03 优化 YOLOv8（量化）



03 优化 YOLOv8（量化）

Data Preparation

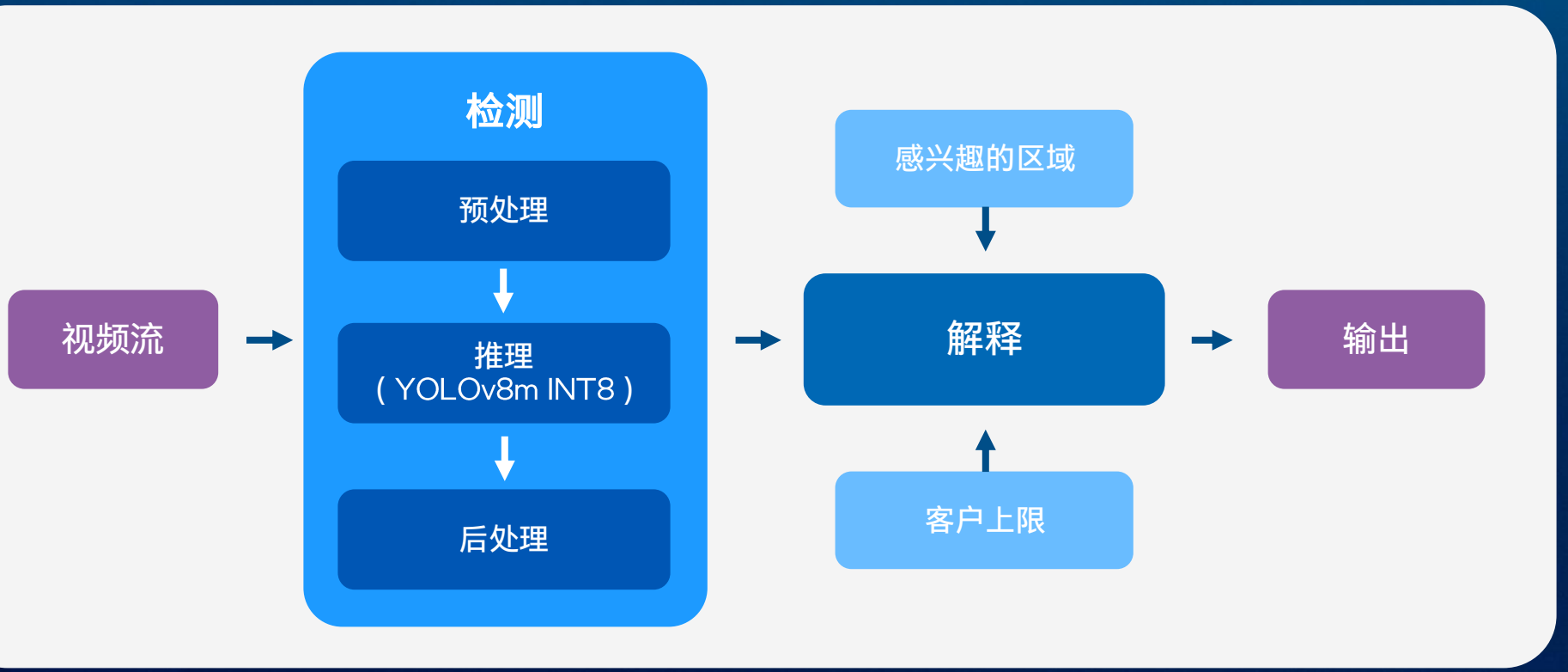
```
import numpy as np
import nncf
from torch.utils.data import DataLoader
from torchvision import datasets
from ultralytics.yolo.data import augment

# create the COCO validation dataset
coco_dataset = datasets.CocoDetection("coco/images/val2017",
                                      annFile="coco/annotations/instances_val2017.json", ...)

# get the loader with batch size 1
det_data_loader = DataLoader(coco_dataset, batch_size=1, shuffle=True)

# transform input function
def transform_fn(data_item):
  # skip the label
  return data_item[0]

quantization_dataset = nncf.Dataset(det_data_loader, transform_fn)
```

03 优化 YOLOv8（量化）

Quantization

```
from openvino import runtime as ov

# load model
core = ov.Core()
ov_model = core.read_model(ov_model_path)

ignored_scope = nncf.IgnoredScope(types=["Multiply", "Subtract", "Sigmoid"],
                                  names=["/model.22/dfl/conv/Conv", "/model.22/Add", ...])
# quantize
quantized_model = nncf.quantize(ov_model, quantization_dataset,
                                preset=nncf.QuantizationPreset.MIXED,
                                ignored_scope=ignored_scope)
# save to disk
ov.serialize(quantized_model, str(int8_model_path))
```

04 运行应用程序



04 运行应用程序

```
# a: Run the Scripts
$ python convert_and_optimize.py --model_name yolov8m --model_dir model --data_dir data –quantize
$ python app.py --stream sample_video.mp4 --model_path model/yolov8m_openvino_int8_model/yolov8m.xml
                --zones_config_file zones.json --customers_limit 3

# b: or Launch the Notebooks
$ jupyter lab docs
```

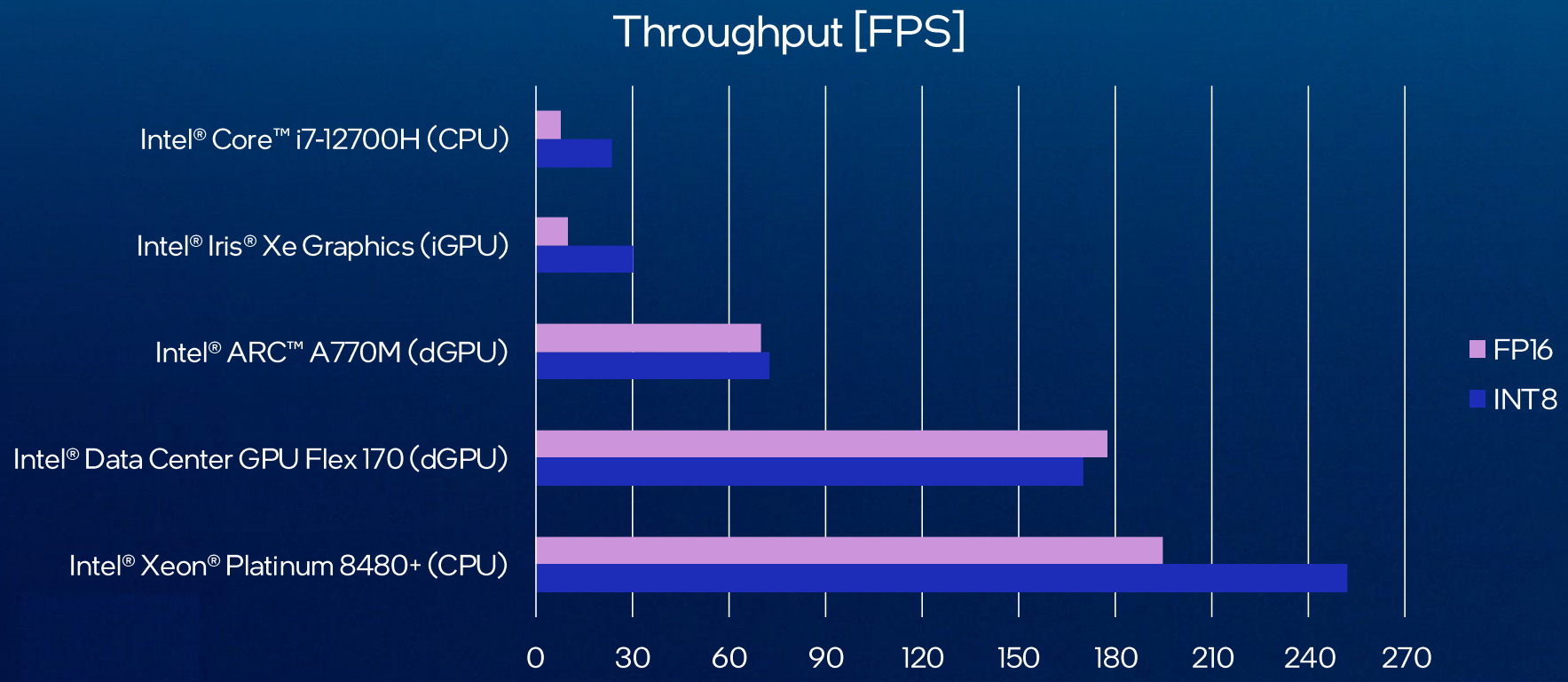
解决方案

Live Demo

解决方案

Live Demo

05 性能基准




```
$ benchmark_app -m $model_path -d $device -hint latency -t 30
```

06 部署

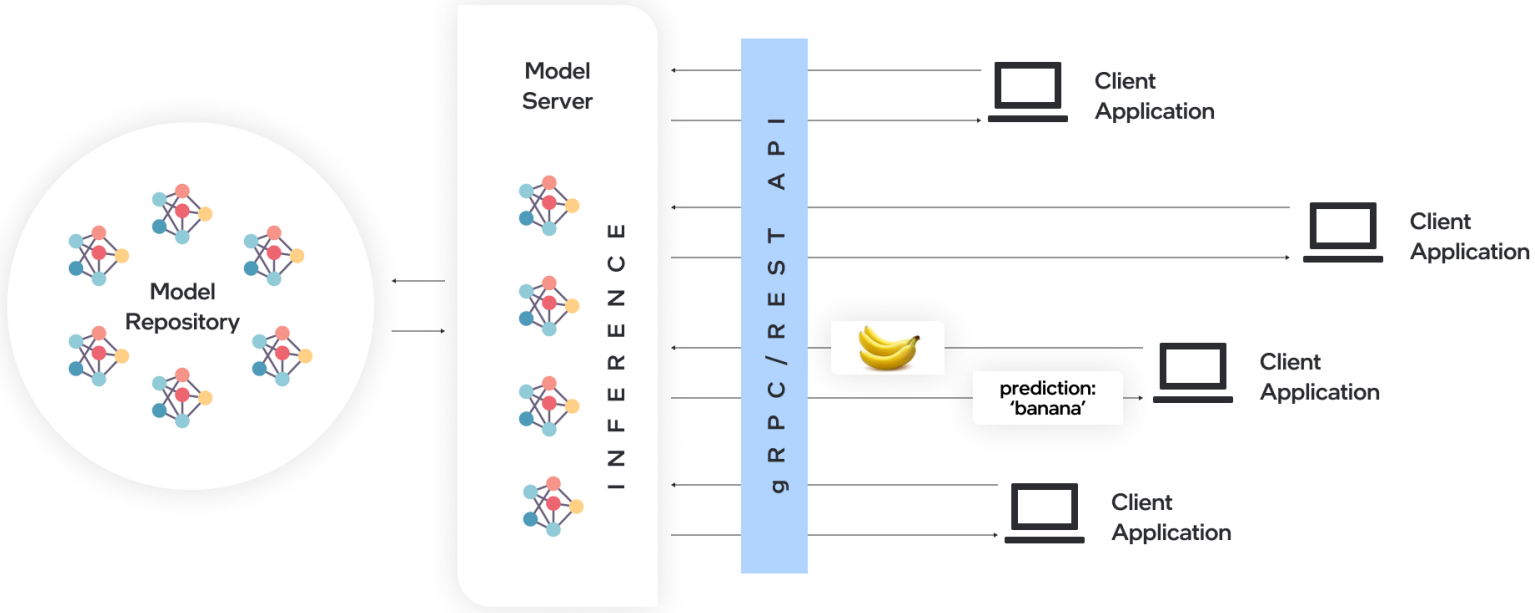

app.py


convert_and_optimize.py


main.py

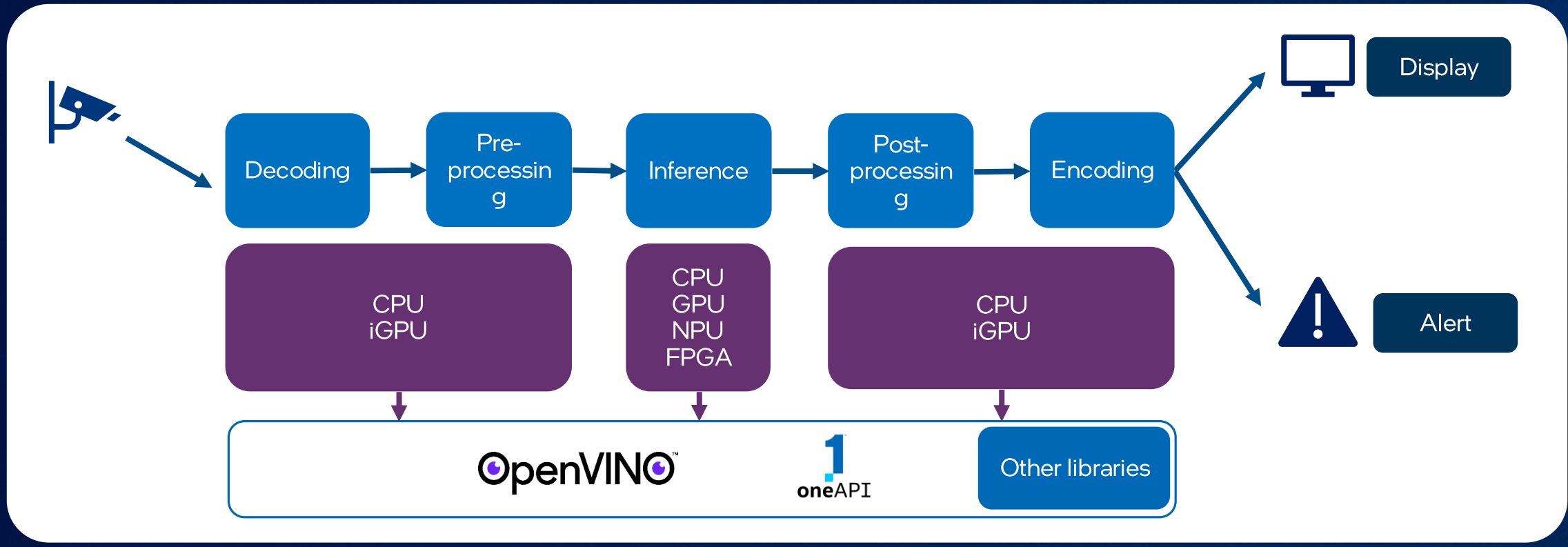
06 部署

OpenVINO™ 模型服务器 (OVMS)



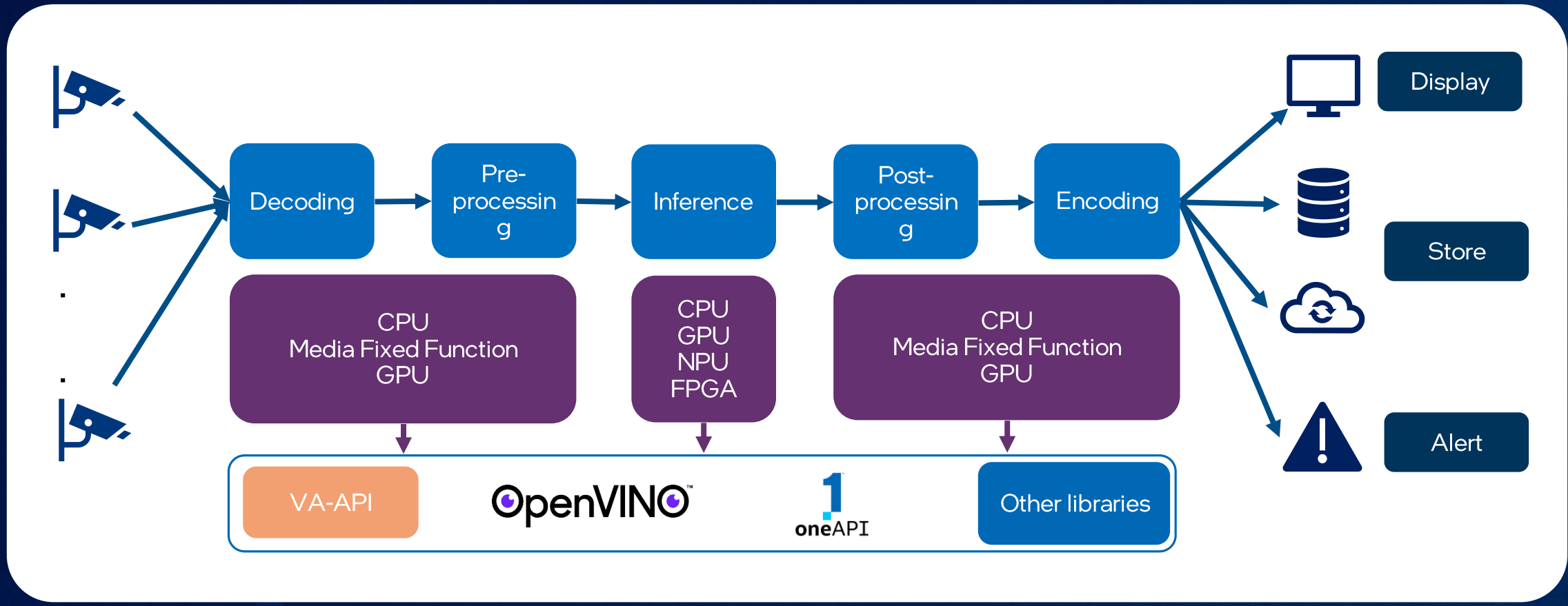
Deploy

Single-stream video analytics pipeline with Intel HW + SW



Deploy

Multi-stream video analytics pipeline with Intel HW + SW



Placeholder for demo video

Notices and disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.
Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel® products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

Intel® technologies may require enabled hardware, software, or service activation. No product or component can be absolutely secure. Your costs and results may vary.