目　录

基于 OpenVINO™2022.2 和蜂蛇峡谷优化并部署 YOLOv5 模型

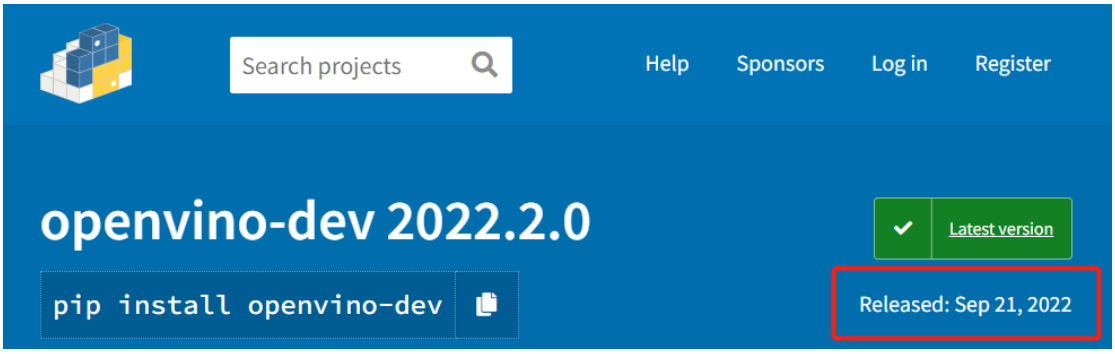
文章作者：

英特尔物联网行业创新大使 **杨雪锋 博士**

中国矿业大学机电工程学院副教授；
发表学术论文 30 余篇，获国家专利授权 20 多件（其中发明专利 8 件）

1.1 OpenVINO™ 2022.2 简介

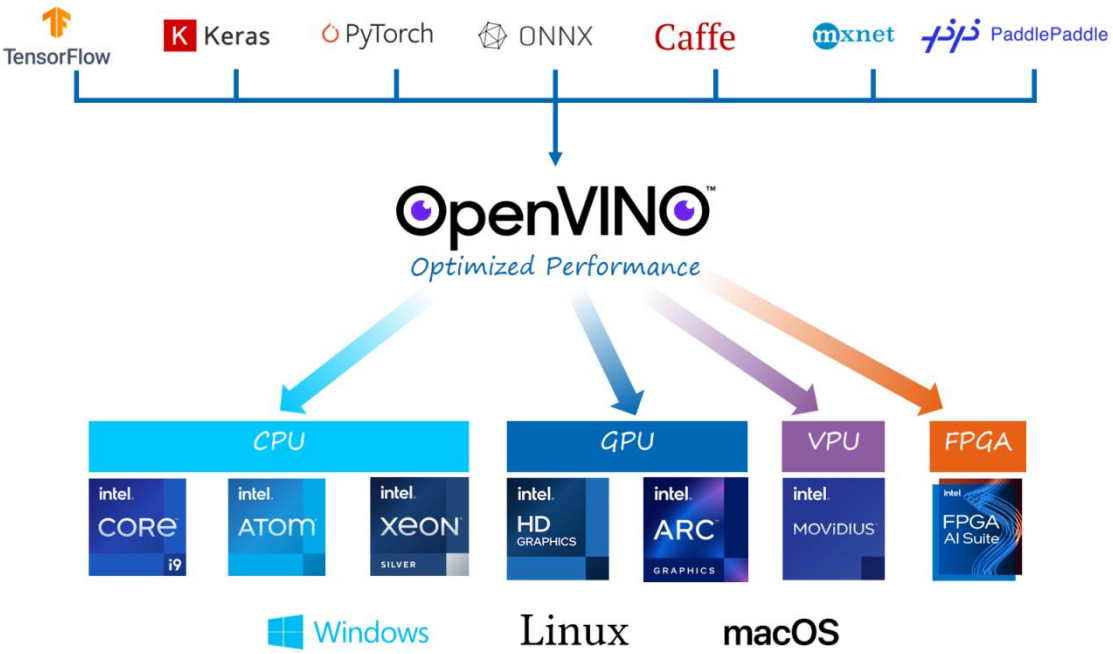
openvino-dev(OpenVINO 开发工具) 2022.2 版于 2022 年 9 月 21 日正式发布，根据官宣《支持英特尔独立显卡的 OpenVINO™ 2022.2 新版本来啦》，OpenVINO™ 2022.2 将是第一个支持英特尔独立显卡的版本。



图片来源：https://pypi.org/project/openvino-dev/

从开发者的角度来看，对于提升开发效率或运行效率有用的特性有：

- **支持英特尔独立显卡**。开发者只需要编写一次 OpenVINO 程序，即可以将 AI 模型通过指定推理设备的方式，部署到英特尔的 CPU、集成显卡、独立显卡、VPU 或 FPGA 上，大大降低了学习投入，提高了开发效率，如下图所示。

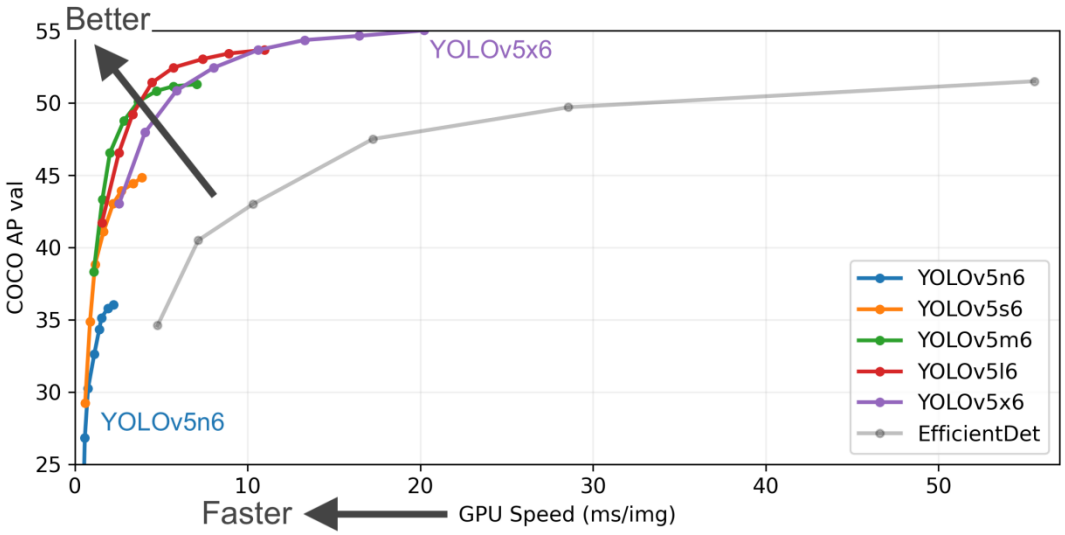


- **为 AUTO 设备插件新增“Cumulative throughput”性能倾向选择。**这意味着开发者无需手动新增推理计算请求，便可在多个 AI 推理计算设备（例如：多个 GPU）上进行并发推理。
- **AUTO 和 MULTI 设备插件支持用“-CPU”将 CPU 移除推理设备列表。**举个例子：“AUTO:-CPU”意味着不使用 CPU 作为 AI 推理计算设备，从而将 CPU 从 AI 推理计算中释放出来，聚焦于任务调度和其它非 AI 推理计算(例如：从摄像头采集图像、运行传统图像处理算法...)。

1.2 YOLOv5 简介

Ultralytics 公司贡献的 YOLOv5 PyTorch (https://github.com/ultralytics/yolov5)实现版，由于其工程化和文档做的特别好，深受广大 AI 开发者的喜爱，GitHub 上的星标超过了 31.1K，而且被 PyTorch 官方收录于 PyTorch 的官方模型仓。

由于 YOLOv5 精度高速度快，且工程化做的非常好，使得产业实践中，即便 YOLOv6 和 YOLOv7 已发布，但大多数人仍然选用 YOLOv5 做目标检测——参考 OpenCV 学堂的测评文章《YOLOv5，YOLOv6，YOLOv7 在 TensorRT 推理速度比较》。



图片来源：https://github.com/ultralytics/yolov5/blob/master/.github/README_cn.md

1.3 蝰蛇峡谷简介

蝰蛇峡谷(Serpent Canyon) 是一款性能强劲，并且体积小巧的高性能迷你主机，搭载全新一代混合架构的第 12 代智能英特尔® 酷睿™ 处理器，并且内置了英特尔锐炫™ A770M 显卡。强悍内芯搭配全新独显的蝰蛇峡谷体积仅约 2.5 升，节省桌面空间的同时提供了丰富的接口，作为生产力工具，从内到外都是高标准要求，能够为用户带来优质的工作体验。

英特尔锐炫™ A770M 显卡基于 Xᵉ-HPG 微架构，Xᵉ HPG GPU 中的每个 Xᵉ 内核都配置了一组 256 位矢量引擎，旨在加速传统图形和计算工作负载，以及新的 1024 位矩阵引擎或 Xᵉ 矩阵扩展，旨在加速人工智能工作负载。

蝰蛇峡谷上有两块 GPU：一块是英特尔®锐炬®集成显卡，一块是英特尔®锐炫® A770M 独立显卡。本文在后面章节将使用 OpenVINO 2022.2 的“**Cumulative throughput**”性能倾向选择新特性，同时使用这两块显卡进行 AI 推理计算。

1.4 准备 YOLOv5 的 OpenVINO 推理程序开发环境

要完成 YOLOv5 的 OpenVINO 2022.2 推理程序开发，需要安装：

- YOLOv5
- OpenVINO 2022.2

由于 YOLOv5 的工程化做的实在太好，在 Windows 中安装上述环境，只需要两条命令：

```
git clone https://github.com/ultralytics/yolov5  # clonecd yolov5
cd yolov5
pip install -r requirements.txt && pip install openvino-dev[onnx] #
install
```

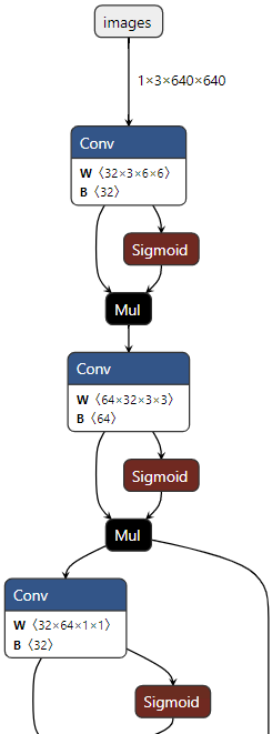
1.5 导出 yolov5s.onnx 模型

在 yolov5 文件夹下，使用命令：python export.py --weights yolov5s.pt --include onnx，完成 yolov5s.onnx 模型导出，如下图所示。

```
(ppov) C:\Users\NUC\Desktop\yolov5>python export.py --weights yolov5s.pt --include onnx
export: data=C:\Users\NUC\Desktop\yolov5\data\coco128.yaml, weights=['yolov5s.pt'], imgsz=[640, 640], ba
tch_size=1, device=cpu, half=False, inplace=False, keras=False, optimize=False, int8=False, dynamic=Fals
e, simplify=False, opset=12, verbose=False, workspace=4, nms=False, agnostic_nms=False, topk_per_class=1
00, topk_all=100, iou_thres=0.45, conf_thres=0.25, include=['onnx']
YOLOv5  v6.2-167-g9006b41 Python-3.8.13 torch-1.12.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients

PyTorch: starting from yolov5s.pt with output shape (1, 25200, 85) (14.1 MB)

ONNX: starting export with onnx 1.11.0...
ONNX: export success  1.6s, saved as yolov5s.onnx (28.0 MB)

Export complete (2.0s)
Results saved to C:\Users\NUC\Desktop\yolov5
Detect:          python detect.py --weights yolov5s.onnx
Validate:        python val.py --weights yolov5s.onnx
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'yolov5s.onnx')
Visualize:       https://netron.app
```

1.6 用 Netron 工具查看 yolov5s.onnx 模型的输入和输出

使用 Netron(https://netron.app/)，查看 yolov5s.onnx 模型的输入和输出



从图中可以看出：YOLOv5 模型的输出叫：“output0”，每张图片的推理结果有 25200 行，每行 85 个数值，前面 5 个数值分别是：

cx, cy, w, h, score, 后面 80 个参数是 MSCOCO 的分类得分。

1.7 使用模型优化器将 yolov5s.onnx 转换为 FP16 精度的 IR 模型

模型优化器(Model Optimizer)是 OpenVINO 自带的跨平台的命令行模型优化工具，当执行完“pip install openvino-dev”安装命令后，模型优化器已经随同 OpenVINO 一并安装好了。参考链接：https://pypi.org/project/openvino-dev/

In addition, the openvino-dev package installs the following components by default:

Component	Console Script	Description
Model Optimizer	mo	**Model Optimizer** imports, converts, and optimizes models that were trained in popular frameworks to a format usable by OpenVINO components. Supported frameworks include Caffe*, TensorFlow*, MXNet*, PaddlePaddle*, and ONNX*.

模型优化器主要通过进行静态模型分析，执行与硬件无关的网络优化(例如：网络层与算子融合、删除死节点等)，更改模型精度，添加归一化参数(mean & scale)等等，最终输出 IR(Intermediate Representation)模型，从而进一步提升 AI 推理计算速度。参考链接：https://docs.openvino.ai/latest/openvino_docs_model_optimization_guide.html

使用模型优化器优化并转换模型时，**模型精度通常选 FP16**，因为它是在 GPU 上性能最好，且所有推理设备都支持的模型精度，参考链接：https://docs.openvino.ai/latest/openvino_docs_OV_UG_supported_plugins_Supported_Devices.html

Supported Model Formats

Plugin	FP32	FP16	I8
CPU plugin	Supported and preferred	Supported	Supported
GPU plugin	Supported	Supported and preferred	Supported
VPU plugins	Not supported	Supported	Not supported
GNA plugin	Supported	Supported	Not supported
Arm® CPU plugin	Supported and preferred	Supported	Supported (partially)

For Multi-Device and Heterogeneous executions the supported models formats depends on the actual underlying devices. *Generally, FP16 is preferable as it is most ubiquitous and performant.*

综上，使用命令：mo --input_model yolov5s.onnx --data_type FP16，进一步优化 yolov5s 模型，并将模型格式转换为 IR 格式，模型精度转换为 FP16，运行结果如下图所示。

```
!mo --input_model yolov5s.onnx --data_type FP16
```

[1] ✓ 2.4s

```
Output exceeds the size limit. Open the full output data in a text editor
Model Optimizer arguments:
Common parameters:
        - Path to the Input Model:      c:\Users\NUC\Desktop\yolov5\yolov5s.onnx
        - Path for generated IR:        c:\Users\NUC\Desktop\yolov5\.
        - IR output name:       yolov5s
        - Log level:    ERROR
        - Batch:        Not specified, inherited from the model
        - Input layers:         Not specified, inherited from the model
        - Output layers:        Not specified, inherited from the model
        - Input shapes:         Not specified, inherited from the model
        - Source layout:        Not specified
        - Target layout:        Not specified
        - Layout:       Not specified
        - Mean values:  Not specified
        - Scale values:         Not specified
        - Scale factor:         Not specified
        - Precision of IR:      FP16
        - Enable fusing:        True
        - User transformations:         Not specified
        - Reverse input channels:       False
        - Enable IR generation for fixed input shape:   False
        - Use the transformations config file:  None
```

1.8 使用 benchmark_app 获得 yolov5s.xml 模型的性能数据

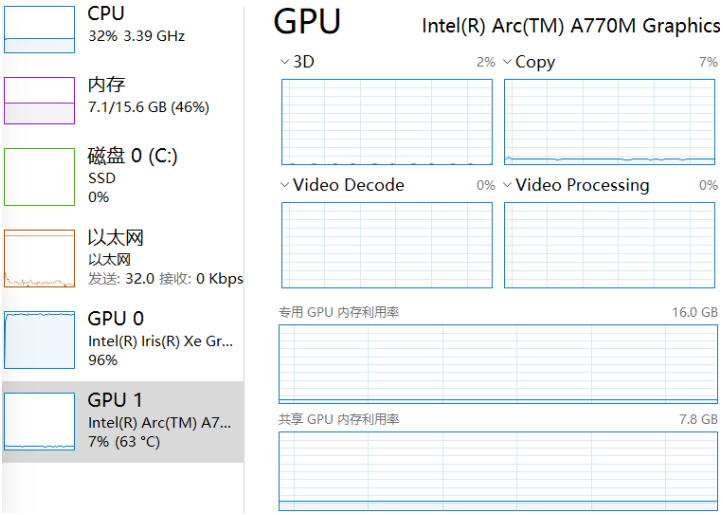
benchmark_app 也是 OpenVINO 自带的跨平台的命令行工具，通过该工具，可以快速获得模型的性能数据。

如前所述，直接使用“**AUTO:-CPU**”设备插件，并指定“**Cumulative throughput**”性能倾向选择，即可同时使用蝰蛇峡谷上的两块显卡做 AI 推理计算。

使用命令获得 yolov5s.xml 模型在两块显卡上同时做 AI 推理计算的性能数据：

```
benchmark_app -m yolov5s.xml -d AUTO:-CPU -hint cumulative_throughput
```

从任务管理器中可以看出，通过“**AUTO:-CPU**”设备插件释放了 CPU；通过“**Cumulative throughput**”性能倾向选择，使得两块显卡都在做 AI 推理计算。



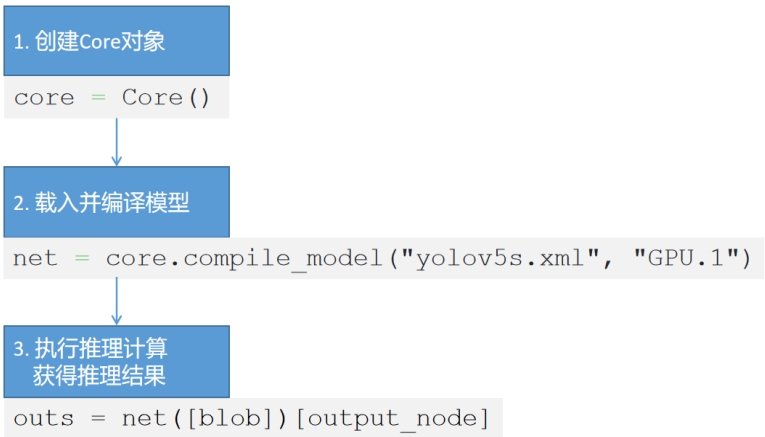
benchmark_app 运行结果，如下图所示。

```
[ INFO ] Read model took 32.37 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1
[Step 6/11] Configuring input of the model
[ INFO ] Model input 'images' precision u8, dimensions ([N,C,H,W]): 1 3 640 640
[ INFO ] Model output 'output0' precision f32, dimensions ([...]): 1 25200 85
[Step 7/11] Loading the model to the device
[ INFO ] Compile model took 6068.09 ms
[Step 8/11] Querying optimal runtime parameters
[ INFO ] DEVICE: AUTO
[ INFO ]   FULL_DEVICE_NAME  , AUTO
[ INFO ]   OPTIMIZATION_CAPABILITIES  , ['BIN', 'FP16', 'FP32', 'GPU_HW_MATMUL', 'INT8']
[ INFO ]   PERF_COUNT  , NO
[ INFO ]   PERFORMANCE_HINT  , CUMULATIVE_THROUGHPUT
[Step 9/11] Creating infer requests and preparing input data
[ INFO ] Create 8 infer requests took 0.00 ms
[ WARNING ] No input files were given for input 'images'!. This input will be filled with random values!
[ INFO ] Fill input 'images' with random values
[Step 10/11] Measuring performance (Start inference asynchronously, 8 inference requests, inference only
 True, limits: 60000 ms duration)
[ INFO ] Benchmarking in inference only mode (inputs filling are not included in measurement loop).
[ INFO ] First inference took 7.53 ms
[Step 11/11] Dumping statistics report
Count:          18168 iterations
Duration:       60091.08 ms
Latency:
    Median:     15.11 ms
    AVG:        26.32 ms
    MIN:        8.69 ms
    MAX:        102.64 ms
Throughput: 302.34 FPS
```

1.9 使用 OpenVINO Runtime API 开发 YOLOv5 的同步推理程序

基于 OpenVINO Runtime API 实现同步推理计算程序的典型流程，主要有三步：

1. 创建 Core 对象；
2. 载入并编译模型；
3. 执行推理计算获得推理结果；



完整范例程序如下所示，总共不到 50 行。本范例程序使用了 yolort(https://github.com/zhiqwang/yolov5-rt-stack)中的 non_max_suppression 实现后处理，运行范例程序前，请先安装 yolort: *pip install -U yolort*

请将范例代码下载后放入 *yolov5* 文件夹下再运行：

https://gitee.com/ppov-nuc/yolov5_infer/blob/main/yolov5_ov2022_sync_dGPU.py

```
import cv2
import time
import yaml
import torch
from openvino.runtime import Core
# https://github.com/zhiqwang/yolov5-rt-stack
from yolort.v5 import non_max_suppression, scale_coords
# Load COCO Label from yolov5/data/coco.yaml
with open('./data/coco.yaml','r', encoding='utf-8') as f:
    result = yaml.load(f.read(),Loader=yaml.FullLoader)
class_list = result['names']
# Step1: Create OpenVINO Runtime Core
core = Core()
# Step2: Compile the Model, using dGPU
net = core.compile_model("yolov5s.xml", "GPU.1")
output_node = net.outputs[0]
# color palette
```

```
colors = [(255, 255, 0), (0, 255, 0), (0, 255, 255), (255, 0, 0)]
#import the letterbox for preprocess the frame
from utils.augmentations import letterbox
```

```
start = time.time() # total excution time =  preprocess + infer + postprocess
frame = cv2.imread("./data/images/zidane.jpg")
# preprocess frame by letterbox
letterbox_img, _, _= letterbox(frame, auto=False)
# Normalization + Swap RB + Layout from HWC to NCHW
blob = cv2.dnn.blobFromImage(letterbox_img, 1/255.0, swapRB=True)
# Step 3: Do the inference
outs = torch.tensor(net([blob])[output_node])
# Postprocess of YOLOv5:NMS
dets = non_max_suppression(outs)[0].numpy()
bboxes, scores, class_ids= dets[:,:4], dets[:,4], dets[:,5]
# rescale the coordinates
bboxes = scale_coords(letterbox_img.shape[:-1], bboxes, frame.shape[:-1]).astype(int)
end = time.time()
#Show bbox
for bbox, score, class_id in zip(bboxes, scores, class_ids):
    color = colors[int(class_id) % len(colors)]
    cv2.rectangle(frame, (bbox[0],bbox[1]), (bbox[2], bbox[3]), color, 2)
    cv2.rectangle(frame, (bbox[0], bbox[1] - 20), (bbox[2], bbox[1]), color, -1)
    cv2.putText(frame, class_list[class_id], (bbox[0], bbox[1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, .5, (255, 255, 255))
# show FPS
fps = (1 / (end - start))
fps_label = "FPS: %.2f" % fps
cv2.putText(frame, fps_label, (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
print(fps_label+ "; Detections: " + str(len(class_ids)))
cv2.imshow("output", frame)
cv2.waitKey()
cv2.destroyAllWindows()
```

1.10 总结

第一：OpenVINO 主要应用于 AI 模型的优化和部署。

第二：OpenVINO 易学易用：

- ✓ **掌握两个命令行工具**：mo 和 benchmark_app 就可以优化并测试模型的性能；
- ✓ **掌握三个 Python API 函数**，就可以完成 AI 模型的同步推理计算程序；
- ✓ 整个范例程序不超过 80 行，清晰易读。即便零基础，也能快速掌握并应用。

第三：通过“**AUTO:-CPU**”设备插件和“**Cumulative throughput**”性能倾向选择，可以非常容易的**同时使用多个 GPU 进行推理计算**，并释放出宝贵的 CPU 资源。

第四：**蝰蛇峡谷**是首款英特尔独显 NUC 迷你电脑，AI 推理性能强大，请参见 benchmark_app 的性能测试结果。