目录

使用 OpenVINO™ 预处理 API 进一步提升 YOLOv5 推理性能

文章作者：杨雪锋 英特尔物联网行业创新大使
文章指导：武卓 英特尔 AI 软件布道师

1.1 概述

在《基于 OpenVINO™ 2022.1 实现 YOLOv5 推理程序》中详述了：

- YOLOv5 框架的安装和如何导出 YOLOv5.onnx 模型
- OpenVINO™ 2022.1 的安装以及如何编写 YOLOv5 模型的推理程序

本文将介绍如何使用 OpenVINO™ 2022.1 的预处理 API，进一步提升 YOLOv5 模型的推理计算性能

1.2 什么是预处理 API 函数？

OpenVINO™ 2022.1 之前版本不提供 OpenVINO™ Runtime 原生的用于数据预处理的 API 函数[1]，如图 1-1 所示，开发者必须通过第三方库(例如：OpenCV)来实现数据预处理。

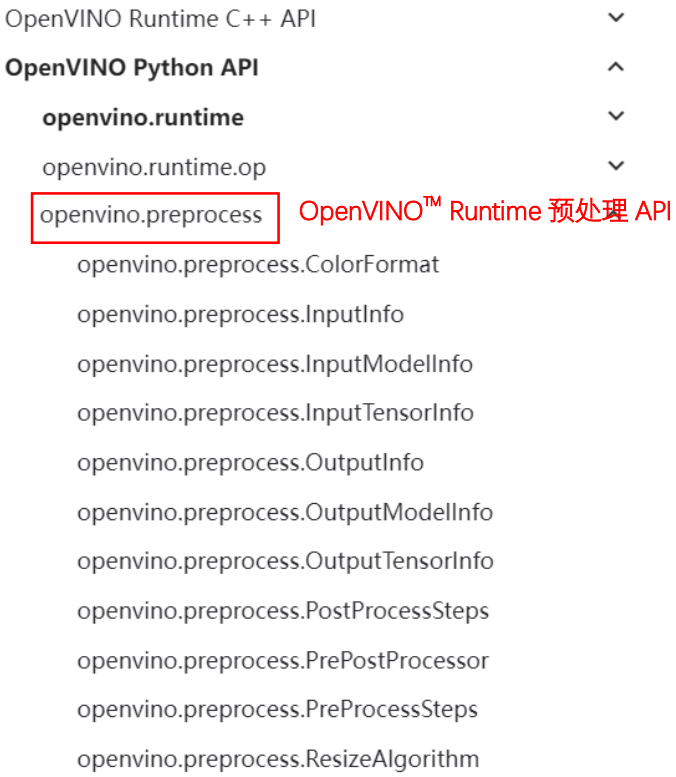

图 1-1 OpenVINO™ Runtime 预处理 API

假设没有预处理 API，那么输入数据的预处理操作只能放在 CPU 上实现，CPU 完成数据预处理后，再将预处理后的数据传给 iGPU、VPU 等 AI 加速计算设备进行推理计算。

有了预处理 API 后，就能将预处理操作集成到在模型执行图中，这样 iGPU、VPU 或即将发布的 Intel 独立显卡都能进行数据预处理，无需依赖 CPU，提高了执行效率，如图 1-2 所示。

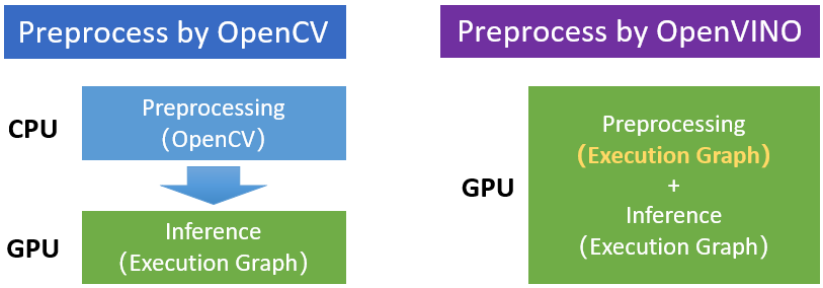

图 1-2　预处理 OpenCV vs OpenVINO

1.3　数据预处理的典型操作

由于输入数据的 Shape、Precision 等特征，与模型输入张量的要求不一致，所以需要通过预处理，将输入数据按照模型输入张量的要求进行转换，如图 1-3 所示。

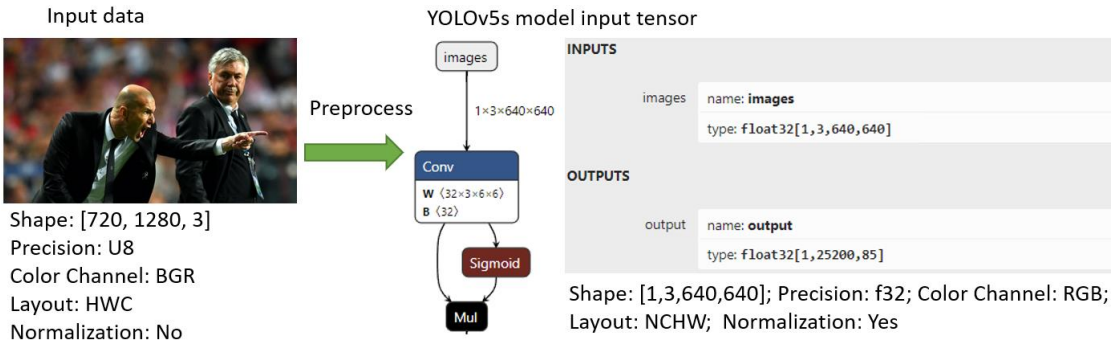

图 1-3　输入数据 vs 模型输入张量

从图 1-3 中可见，数据预处理的典型操作有：

- 改变输入数据的形状：[720, 1280,3] → [1, 3, 640, 640]
- 改变输入数据的精度：U8 → f32
- 改变输入数据的颜色通道顺序：BGR → RGB
- 改变输入数据的布局(layout)：HWC → NCHW
- 归一化数据：减去均值(mean)，除以标准差(std)

1.4　数据预处理 API 的使用方法

对应数据预处理的典型操作，OpenVINO™ 预处理 API 提供了相应的类，方便开发者快速使用，其主要流程有 6 步[2]，如图 1-4 所示，依次是：

1. 实例化 PrePostProcessor 对象；
2. 申明输入数据的信息
3. 指定模型的数据布局(layout)
4. 设置模型输出张量的信息
5. 定义预处理的具体步骤
6. 将预处理步骤集成到模型

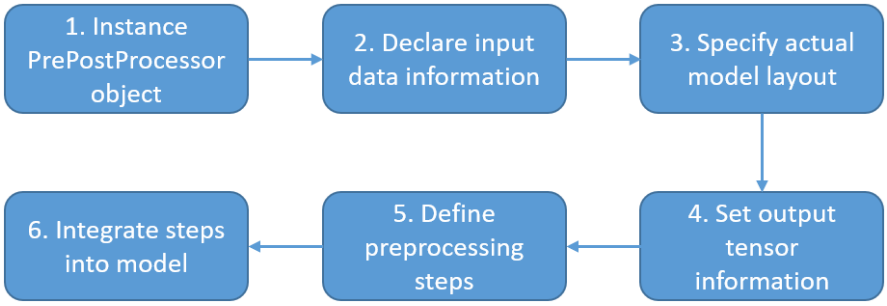

图 1-4　使用预处理 API 的流程

本文将按照上述顺序依次介绍。

1.4.1 实例化 PrePostProcessor 对象

实例化 PrePostProcessor 对象的 Python 代码，如代码清单 1-1 所示。

代码清单 1-1　实例化 PrePostProcessor 对象

```
from openvino.runtime import Core, Type, Layout
from openvino.preprocess import PrePostProcessor, ColorFormat

# Please modify the model path
model_path = "yolov5s.onnx"
model = core.read_model(model_path)
# Step1: Instance PrePostProcessor object
ppp = PrePostProcessor(model)
```

1.4.2 申明输入数据的信息

申明输入数据的信息的 Python 代码，如代码清单 1-2 所示。

代码清单 1-2　申明输入数据的信息

```
# Step2: Declare input data information:
ppp.input().tensor() \
    .set_color_format(ColorFormat.BGR) \
    .set_element_type(Type.u8) \
    .set_layout(Layout('NHWC'))
```

1.4.3 指定模型的数据布局(layout)

指定模型的数据布局(layout) 的 Python 代码，如代码清单 1-3 所示。

代码清单 1-3　指定模型的数据布局(layout)

```
# Step3: Specify actual model layout
ppp.input().model().set_layout(Layout('NCHW'))
```

1.4.4 设置模型输出张量的信息

设置模型输出张量的信息的 Python 代码，如代码清单 1-4 所示。

代码清单 1-4　设置模型输出张量的信息

```
# Step4: Set output tensor information:
# - precision of tensor is supposed to be 'f32'
ppp.output().tensor().set_element_type(Type.f32)
```

1.4.5 定义预处理的具体步骤

定义预处理的具体步骤的 Python 代码，如代码清单 1-5 所示。

代码清单 1-5　定义预处理的具体步骤

```
# Step5: Apply preprocessing modifing the original 'model'
# - Precision from u8 to f32
# - color plane from BGR to RGB
# - subtract mean
# - divide by scale factor
# - Layout conversion will be done automatically as last step
ppp.input().preprocess() \
    .convert_element_type(Type.f32) \
    .convert_color(ColorFormat.RGB) \
    .mean([0.0, 0.0, 0.0]) \
    .scale([255.0, 255.0, 255.0])
```

1.4.6 将预处理步骤集成到模型

将预处理步骤集成到模型的 Python 代码，如代码清单 1-6 所示。

代码清单 1-6　将预处理步骤集成到模型

```
# Step6: Integrate preprocessing steps into model
print(f'Build preprocessor: {ppp}')
model = ppp.build()
```

1.4.7 将集成了预处理步骤的模型导出

使用 serialize()函数，可以将集成了预处理步骤的模型导出，方便后续调用，如代码清单 1-7 所示。

代码清单 1-7　导出包含预处理步骤的模型

```
# Save the Model with preprocess
from openvino.offline_transformations import serialize
serialize(model, 'yolov5s.xml', 'yolov5s.bin')
```

使用 Netron 打开导出模型，可以看到预处理步骤已经集成到执行图中，如图 1-5 所示。

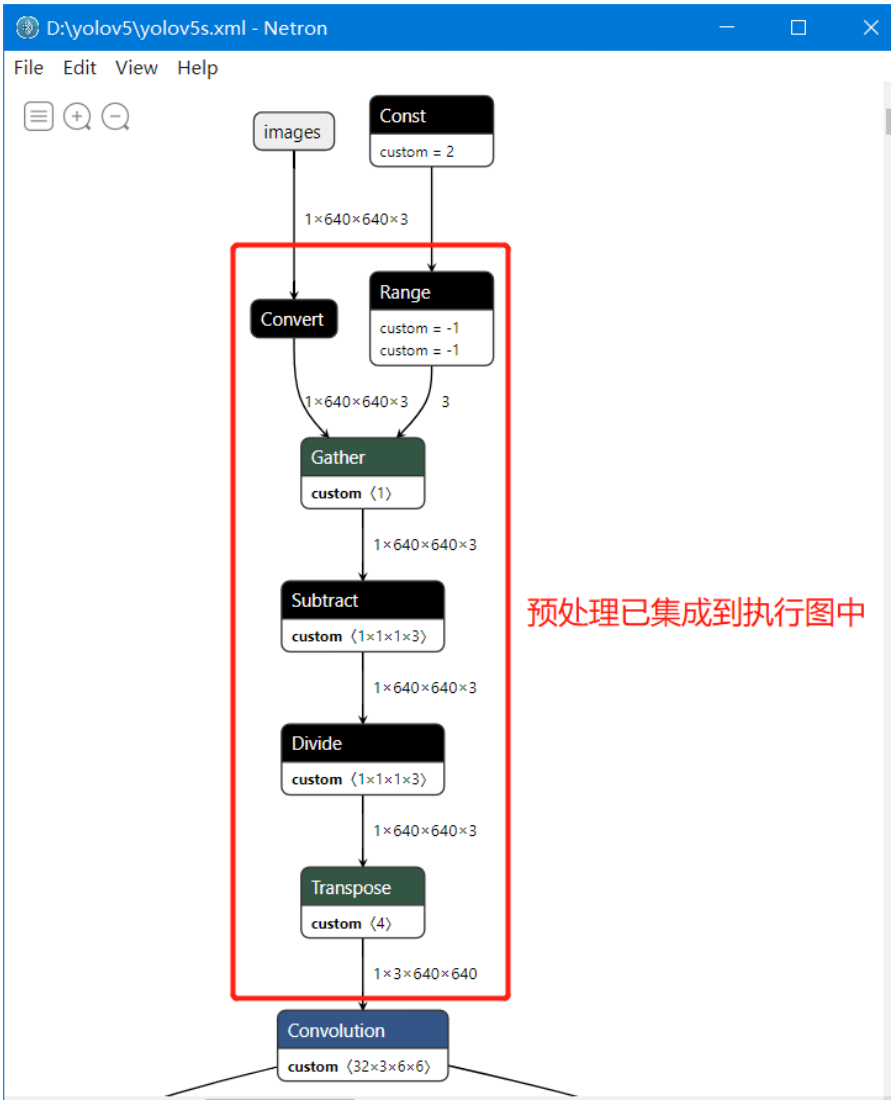

图 1-5　预处理集成到执行图中

导出集成预处理模型的完整源代码：https://gitee.com/ppov-nuc/yolov5_infer/blob/main/preprocessing_with_saving_to_IR.py

1.5　完整范例代码和测试结果

本文随附使用 OpenVINO™ 2022.1 预处理 API 实现 YOLOv5s 推理程序的完整源代码，参见：https://gitee.com/ppov-nuc/yolov5_infer/blob/main/infer_with_openvino_preprocess.py。

表 1-1 使用 OpenVINO™ 2022.1 预处理 API 和使用 OpenCV 实现预处理的性能对比

操作系统：Windows10；Python 版本：3.8；OpenVINO 版本：2022.1

模型：yolov5s.onnx

预处理方式	CPU/iGPU 型号	推理设备	迭代次数	运行时间
OpenVINO™ 预处理 API	i7-1165G7 Intel®Iris®Xe	device=”GPU”	500	12.68s

OpenCV	i7-1165G7 Intel®Iris®Xe	device=”GPU”	500	13.43s

执行命令，将 yolov5s.onnx 转换为 FP16 精度的 yolov5s.xml 后

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

操作系统：Windows10；Python 版本：3.8；OpenVINO 版本：2022.1

模型：yolov5s.xml @ FP16

预处理方式	CPU/iGPU 型号	推理设备	迭代次数	运行时间
OpenVINO™ 预处理 API	i7-1165G7 Intel®Iris®Xe	device=”GPU”	500	7.41s
OpenCV	i7-1165G7 Intel®Iris®Xe	device=”GPU”	500	8.24s

1.6 总结

本文完整介绍了什么是 OpenVINO™ 预处理 API 和为什么推荐使用预处理 API 将预处理操作集成到模型执行图中，然后详细介绍了使用步骤并提供了完整范例源代码。

通过运行源代码，可以看到，使用了 OpenVINO™预处理 API，使输入数据预处理操作不再依赖 CPU，可以由推理设备(如 GPU/VPU)完成，提高了推理计算效率，减少了运行时间。

参考文献：

[1] https://docs.openvino.ai/nightly/api/ie_python_api/_autosummary/openvino.preprocess.html

[2] https://docs.openvino.ai/nightly/openvino_docs_OV_UG_Preprocessing_Details.html

[3] https://github.com/openvinotoolkit/openvino_notebooks/blob/main/notebooks/002-openvino-api/002-openvino-api.ipynb