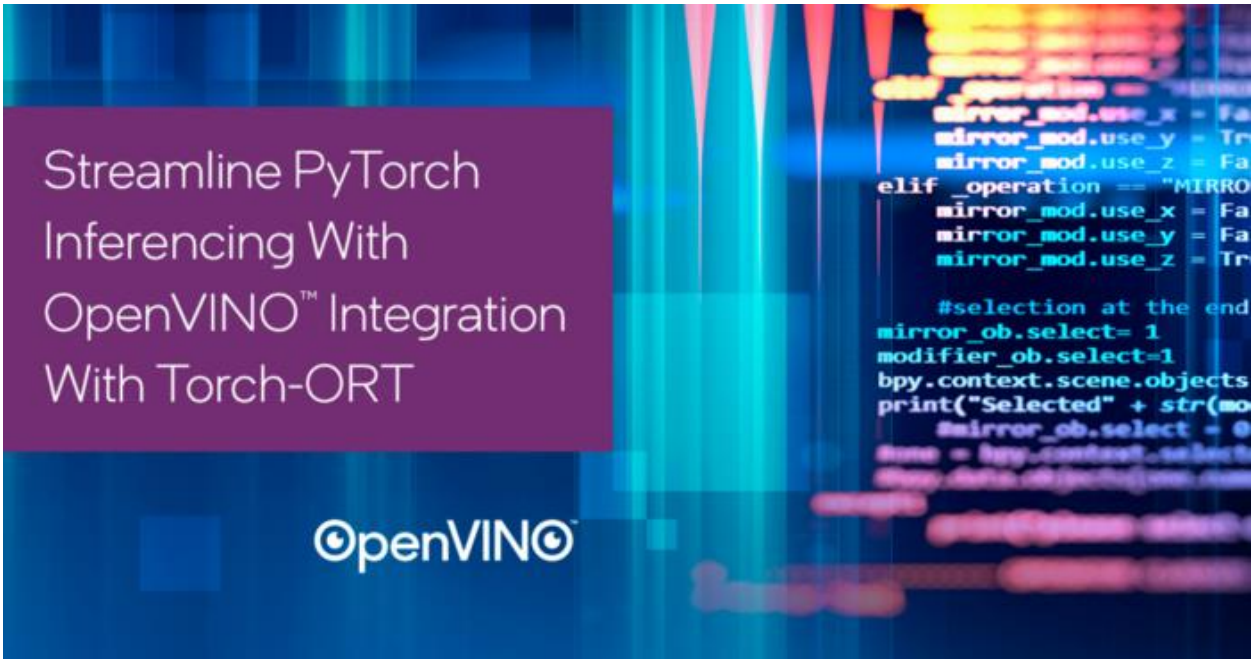
PyTorch 开发者福音, OpenVINO 整合 PyTorch 实现推理加速！

作者：Ashish Datta, Sai Jayanthi, Natalie Kershaw (Microsoft), Yamini Nimmagadda, Sesh Seshagiri

编译：李翊玮



介绍

您是否希望最少的代码更改下将 PyTorch API 与 OpenVINO™ 工具包 结合提升推理性，同时进行？不用再犹豫了，我们与微软紧密合作开发并很高兴地宣布，OpenVINO™与 ONNX Runtime 用于 PyTorch 的集成（简称 OpenVINO™与 Torch-ORT 集成）完成了。

OpenVINO 重塑了在英特尔®支持设备下人工智能推理的定义，获得了前所未有的开发者采用。如今，成千上万的开发者使用 OpenVINO 在几乎所有可以想象的用例中加速 AI 推理，从模拟人类视觉，自动语音识别，自然语言处理，推荐系统等等。该工具包基于最新一代的人工神经网络，包括卷积神经网络 （CNNs）、循环网络和基于注意力的网络，可跨英特尔硬件（英特尔 CPU、英特尔集成显卡、英特尔神经计算棒 2 代 和英特尔基于 Movidius™ VPU 视觉加速器设计）扩展视觉和非视觉工作负载，最大限度地提高性能。OpenVINO 通过从边缘部署到云的高性能、AI 和深度学习推理来加速应用程序。

随着 OpenVINO 生态系统的发展，我们从 PyTorch 开发者听到希望使用更无缝的方法来加速 PyTorch 模型。在此之前，想要加速模型的 PyTorch 开发者必须将其模型转换为 ONNX，然后使用 OpenVINO™ 运行时运行它，或者将 ONNX 模型转换为 OpenVINO™ 工具包 IR 以进行推理。

这给 PyTorch 开发者带来了几个问题，例如

- 由于 ONNX 转换不支持的层(layers)/运算符(operators)而导致的 ONNX 转换失败
- 由于 OpenVINO 不支持的转换模型的(layers)/运算符(operators)而导致的 ONNX 推理失败
- 模型转换所需的额外步骤（PyTorch -> ONNX, ONNX -> OpenVINO IR 等）

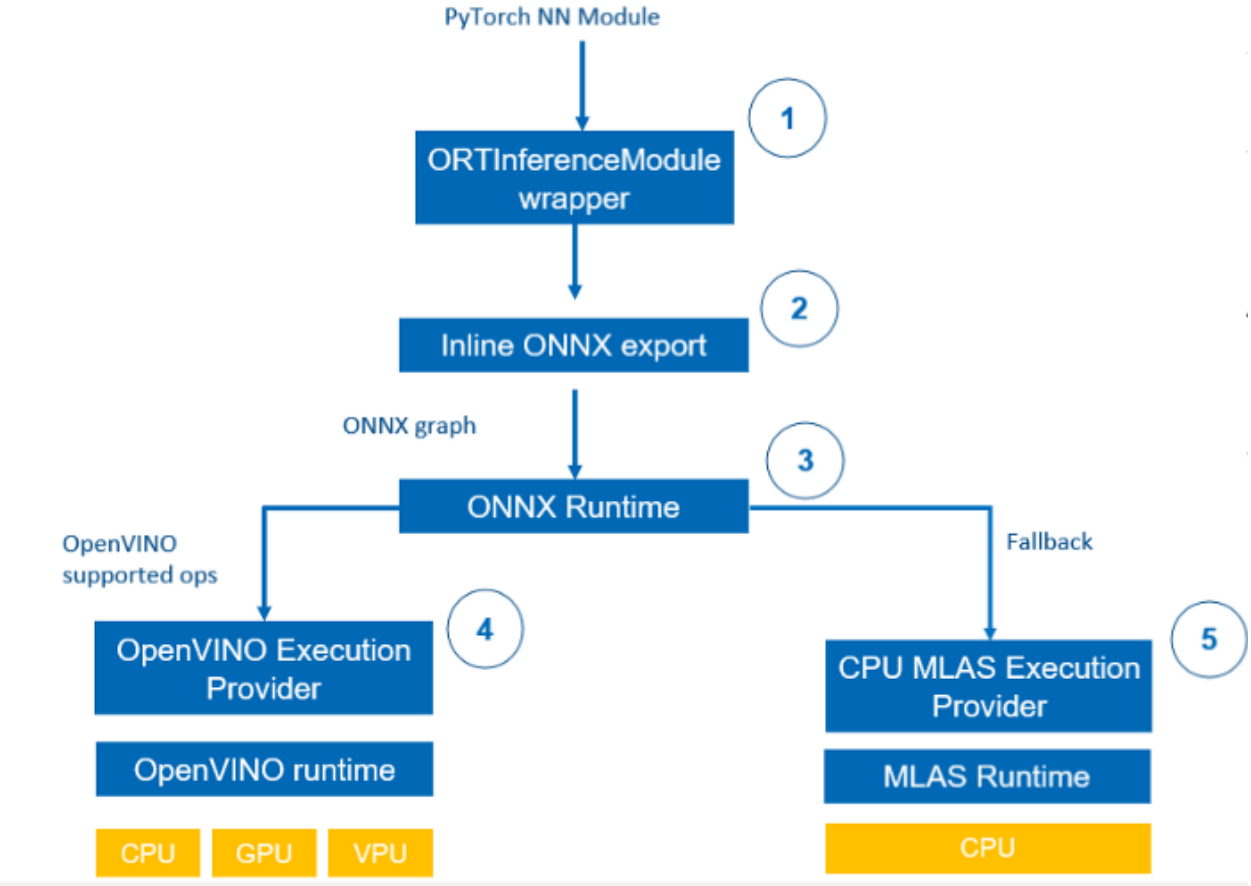
为什么 PyTorch 开发者应该使用 OpenVINO 与 Torch-ORT 的集成?

OpenVINO 与 Torch-ORT 的集成使 PyTorch 开发者能够始终保持在他们选择的框架内，同时通过用于的内联优化来获得 OpenVINO™ 工具包的推理加速能力加速你的 PyTorch 应用程序。

通过英特尔® OpenVINO™ 与 Torch-ORT 集成加速 PyTorch 性能

- **易于安装** — 将 OpenVINO 与 Torch-ORT 与 PIP 集成
- **简单的 API** — 无需重构现有代码，只需导入 OpenVINO 与 Torch-ORT 的集成，将所需的目标设备设置为运行推理并包装模型即可开始使用!
- **性能** — 实现比原有 PyTorch 更高的推理性能
- **支持英特尔设备** — 英特尔 CPU、英特尔集成 GPU、英特尔 VPU
- **内联模型转换** — 无需显式模型转换步骤

它是如何运作的



1. 用户将他们的 nn.Model 包装于 torch_ort.ORTInferenceModule， 使用在准备用于推理 ONNX Runtime OpenVINO Execution Provider 的模块。

2. 该模块用 torch.onnx.export 导出至内存 ONNX 图形。

3. 启动 ONNX Runtime session 时，ONNX graph 会作为输入。ONNX Runtime 将用受支持和不受支持的运算符原则将 graph 划分为 subgraph。

4. 所有与 OpenVINO 兼容的节点都将由 OepnVINO Execution Provider 执行，并且可以在英特尔 CPU、GPU 或 VPU 上执行。

5. 所有其他节点将回退到默认的 CPU MLAS Execution Provider 程序。该程序将为默认运算符 set domain 中的节点调用 ONNX Runtime 自己的 CPU 内核。

开发者体验

开始使用 OpenVINO 与 Torch-ORT 的集成很容易，因为它遵循了 PyTorch 编程方式。下面是从 HuggingFace Transformers 中获取预训练的 NLP BERT 模型并使其与 OpenVINO™与 Torch-ORT 集成兼容的代码

代码示例

安装：

```
pip install torch-ort-infer[openvino]
python -m torch_ort.configure
```

定义模型：

```
from torch_ort import ORTInferenceModule
model = ORTInferenceModule(model)Provider options for different devices from torch_ort
import ORTInferenceModule, OpenVINOProviderOptions
provider_options = OpenVINOProviderOptions(backend = "GPU", precision = "FP16")
model = ORTInferenceModule(model, provider_options = provider_options)
```

代码示例：

通过简单地包装你的 nn.Module 与 ORTInferenceModule 您可以获得 OpenVINO 推理加速的优势。

```
from transformers
import AutoTokenizer, AutoModelForSequenceClassification
import numpy as np
from torch_ort import ORTInferenceModuletokenizer = AutoTokenizer.from_pretrained(
```

```
            "textattack/bert-base-uncased-CoLA")
model = AutoModelForSequenceClassification.from_pretrained(
        "textattack/bert-base-uncased-CoLA")
# Convert nn.Module to ORTInferenceModule to leverage OpenVINO on CPU
model = ORTInferenceModule(model)text = "Replace me any text by you'd like ."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input) # Post processing
logits = output.logits
logits = logits.detach().cpu().numpy() # predictions
pred = np.argmax(logits, axis=1).flatten()
print("Grammar correctness label (0=unacceptable, 1=acceptable)")
print(pred)
```

通过两行代码，OpenVINO 与 Torch-ORT 的集成可以帮助确保您获得出色的推理性能，并利用完整的 OpenVINO 工具包提供的大部分功能。如果您有兴趣尝试它，请查看 Github 存储库并阅读此处提供的详细文档。

实例代码：在 Python 中使用 Torch-ORT 推理模块进行 Resnet 图像分类

1. 本演示展示了如何使用英特尔® OpenVINO™ 与 Torch-ORT 的集成对图像中的对象进行分类。
2. 我们使用 Torchvision 的图像分类模型 ResNet-50 和 ImageNet 标签对对象进行分类。在标签文件中，您将找到 Imagenet 竞赛中使用的 1,000 个不同类别 。

导入必要的资源：

```
import os
import time
import torch
import wget
import argparse
from PIL import Image
from torchvision import transforms
import torchvision.models as models
from torch_ort import ORTInferenceModule, OpenVINOProviderOptions
```

下载图像标签档案：

- 从 github 下载 imagenet 分类档案

```
def download_labels(labels):
    if not labels:
        labels = "imagenet_classes.txt"
        if not os.path.exists(labels):
            labelsUrl = (

"https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt"
            )
            # Download the file (if we haven't already)
```

```
            wget.download(labelsUrl)
        else:
            print("\nReusing downloaded imagenet labels")

    # Read the categories
    with open(labels, "r") as f:
        categories = [s.strip() for s in f.readlines()]
      return categories
```

预处理功能：

- 调整输入大小
- 裁剪输入
- 将图像输入转换为张量并归一化

```
    def preprocess(img):
    transform = transforms.Compose(
        [
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
        ]
    )
    return transform(img)
```

推理功能：

- 对输入图像运行推理
- 打印推理结果

```
def infer(model, image, categories):
    # warmup
    model(image)

    # Start inference
    t0 = time.time()
    outputs = model(image)
    t1 = time.time() - t0
    print("\nInference time: {:.4f}ms\n".format(t1 * 1000))

    # The output has unnormalized scores. Run a softmax on it for probabilities.
    probabilities = torch.nn.functional.softmax(outputs[0], dim=0)

    # Show top categories per image
    top5_prob, top5_catid = torch.topk(probabilities, 5)
    print("Top 5 Results: \nLabels , Probabilities:")
    for i in range(top5_prob.size(0)):
        print(categories[top5_catid[i]], top5_prob[i].item())
```

获取输入档案：

```
#Create List of files in the directory
files = os.listdir('.')
```

```
#Get the necessary files into the directory if they don't already exist
if ('plane.jpg' not in files):
  !wget https://media.wired.com/photos/62b25f4c18e6fafaa97a6477/master/pass/Air-Serbia-Plane-Russian-Sanctions-Safety-Hazard-Business-1239498184.jpg -O
```

运行结果

```
--2022-08-18 16:35:40-- https://media.wired.com/photos/62b25f4c18e6fafaa97a6477/master/pass/Air-Serbia-Plane-Russian-Sanctions-Safety-Hazard-Business-1239498184.jpg
Resolving media.wired.com (media.wired.com)... 151.101.0.239, 151.101.64.239, 151.101.128.239, ...
Connecting to media.wired.com (media.wired.com)|151.101.0.239|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 319129 (312K) [image/jpeg] Saving to: 'plane.jpg' plane.jpg 100%[===================>] 311.65K --.-KB/s in 0.003s
2022-08-18 16:35:40 (98.3 MB/s) - 'plane.jpg' saved [319129/319129]
```

选择输入代码：

- 使用 input_file 选择要推理的输入图像
- 可用的后端精度
  - CPU： FP32
  - GPU（不协作）： FP32， FP16
- 可用的推理执行提供程序
  - OpenVINO

```
input_file = "plane.jpg"
backend ="CPU"
precision = "FP32"
```

运行结果

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:209: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and will be removed in 0.15, please use 'weights' instead.
f"The parameter '{pretrained_param}' is deprecated since 0.13 and will be removed in 0.15, "
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and will be removed in 0.15. The current behavior is equivalent to passing `weights=ResNet50_Weights.IMAGENET1K_V1`. You can also use `weights=ResNet50_Weights.DEFAULT` to get the most up-to-date weights. warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
```

使用原生 PyTorch 运行推理：

```
# Infer
infer(model, img_trans, categories)
```

运行结果

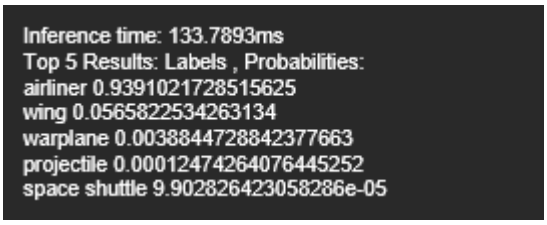
```
Inference time: 234.3407ms
Top 5 Results: Labels , Probabilities:
airliner 0.9391021728515625
wing 0.0565822534263134
warplane 0.0038844654336571693
projectile 0.0001247427571797743 4
space shuttle 9.90280750556849e-05
```

使用 Torch-ORT module 运行推理：

```
# Select OpenVINO as inference execution provider
if backend and precision:
    provider_options = OpenVINOProviderOptions(backend, precision)
    model_ort = ORTInferenceModule(model, provider_options=provider_options)
else:
    model_ort = ORTInferenceModule(model)

# Infer
infer(model_ort, img_trans, categories)
img.close()
```

运行结果



在这个案例里, 由以上两种推理结果可以看出推理时间因使用了 Torch-ORT 模块得到了加速, 用 PyTorch 的小伙伴们, 赶快动起来试试, 看是否你的模型也用此方案能得到性能提升

您还能从这里期待什么：

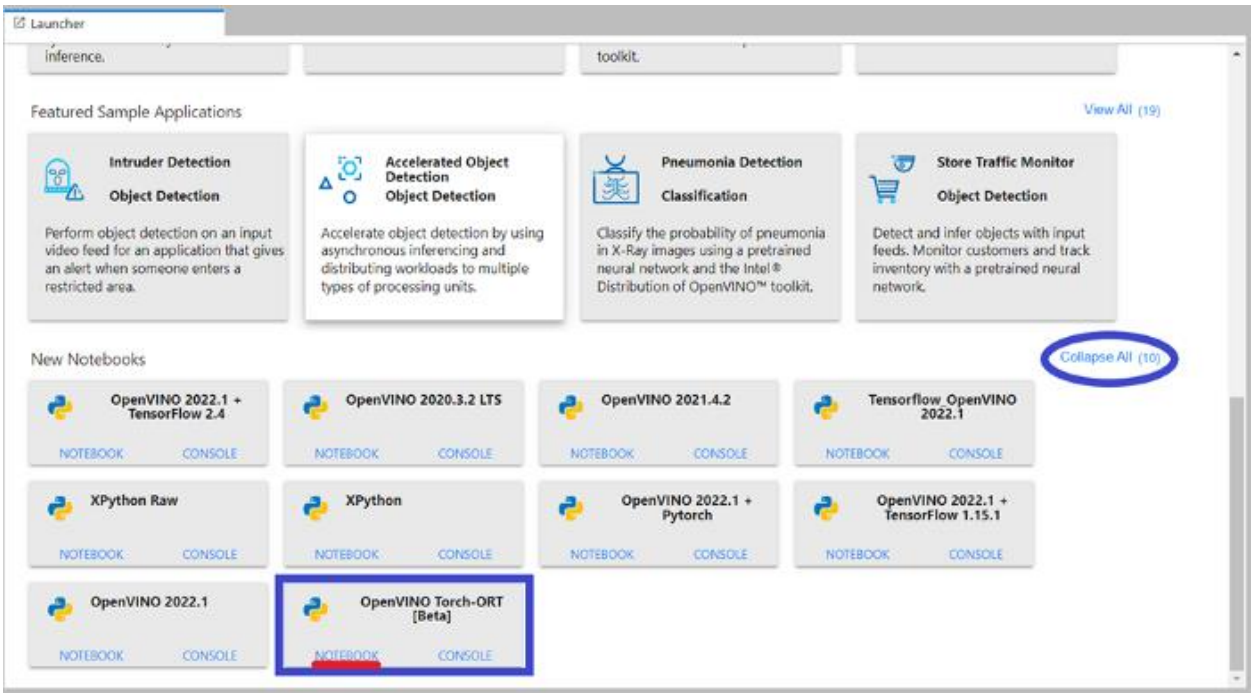
- 其他代码示例（Python scripts、Jupyter Notebooks）
- 更多 TorchVision 和 HuggingFace Transformers 模型覆盖范围

开始使用的英特尔® DevCloud

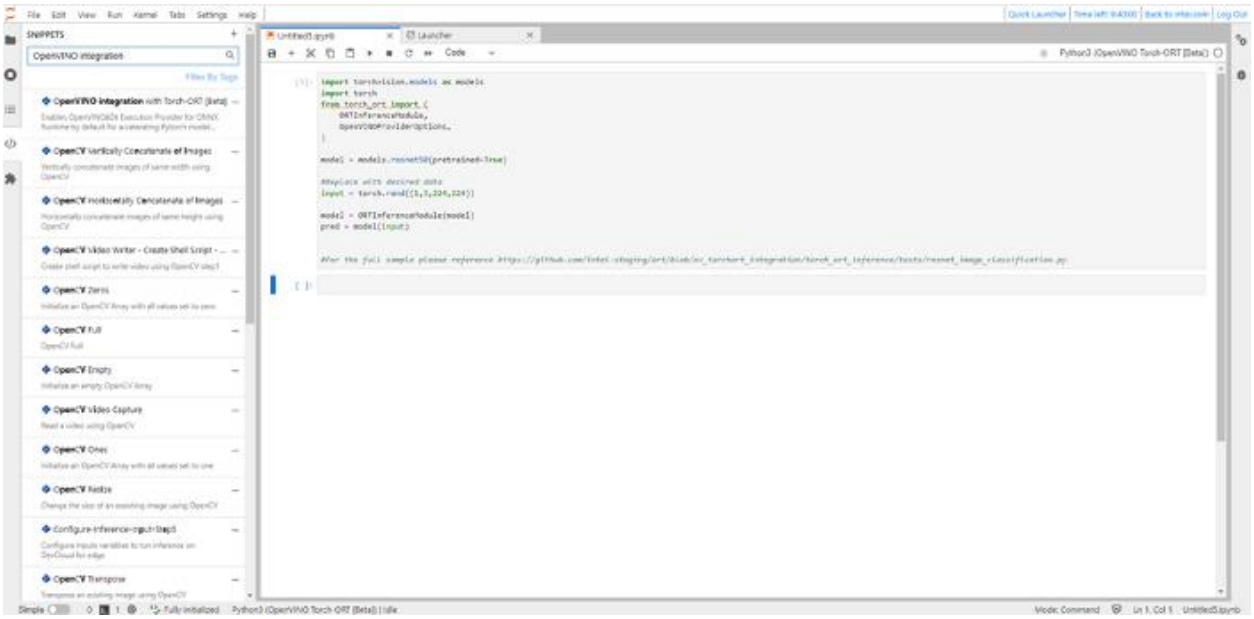
英特尔® DevCloud 是一个云开发环境，使开发者能够以最少的设置在英特尔硬件上开发和基准测试人工智能和计算机视觉解决方案的性能。面向边缘的 Devcloud 预装了 OpenVINO™ 集成与 Torch-ORT [beta]以及代码片段，可帮助您加快开发过程。要开始使用，请按照以下步骤操作:

步骤 1.在此处注册并登录免费访问

步骤 2.向下滚动并选择 OpenVINO™集成 Torch - ORT [beta] Notebook



第 3 步.打开代码片段，搜索“OpenVINO™ Torch - ORT [beta]”，将代码片段拖放到笔记本中



步骤 4.运行单元

如果您遇到任何问题，请联系英特尔® DevCloud 支持团队。

通知和免责声明：

英特尔技术可能需要支持的硬件、软件或服务激活。

没有任何产品或组件是绝对安全的。

您的费用和结果可能会有所不同。