标题：OpenVINO™ 常用 PythonAPI 详解与演示

OpenVINO™2022.1 API 简介

OpenVINO™2022.1 经过较大的升级改版后，API 的使用体验比之前版本好多了。而且功能比较丰富，特别是增加了对动态尺度(Dynamic shape)输入的支持，相比之前只支持固定尺度(Staticshape)输入来说，大大提升了易用性。OpenVINO™ 的 Python 版本的 API 简单易学，容易上手，只需要掌握下面几个函数就可以完成从模型加载到推理。

读者在学习常用 OpenVINO™PythonAPI 之前，请进入：https://github.com/openvinotoolkit/openvino_notebooks/tree/main/notebooks/002-openvino-api，并点击：“LaunchBinder”，进入 OpenVINO™PythonAPI 实验环境。

01 导入支持

要使用 Python SDK，首先需要导入 OpenVINO™ Runtime 语句，

```
fromopenvino.runtimeimport Core
ie= Core()
```

02 加载模型

OpenVINO™2022.1 版本加载模型提供了两种方法：read_model()与 compile_model()。

- read_model()：将模型从硬盘载入内存，并返回 model 对象。model 对象可以通过 compile_model()方法编译为可以在目标设备上执行的 compile_model 对象。
- compile_model()：将模型从硬盘载入内存，编译模型，并返回 compile_model 对象。

```
fromopenvino.runtimeimport Core

ie= Core()
classification_model_xml="model/classification.xml"

model =ie.read_model(model=classification_model_xml)
compiled_model=ie.compile_model(model=model, device_name="CPU")
```

03 模型的输入与输出层信息

通过 read_model()方法获得的 model 对象，和通过 compile_model()方法获得的 compiled model 对象，都支持直接访问属性获取输入与输出层信息。

```
fromopenvino.runtimeimport Core

ie= Core()
classification_model_xml="model/classification.xml"
model =ie.read_model(model=classification_model_xml)
model.input(0).any_name
input_layer=model.input(0)
print(f"input precision: {input_layer.element_type}")
```

```
print(f"input shape: {input_layer.shape}")
```

```
[out]
'input'
input precision: <Type: 'float32'>
input shape: {1, 3, 224, 224}
```

```
model.output(0).any_name
output_layer=model.output(0)
print(f"output precision: {output_layer.element_type}")
print(f"output shape: {output_layer.shape}")
```

```
[out]
'MobilenetV3/Predictions/Softmax'
output precision: <Type: 'float32'>
output shape: {1, 1001}
```

04 修改模型输入

model 对象的 reshape()方法支持修改模型输入形状，当前支持的修改参数包括：batch size、输入图像的宽和高。

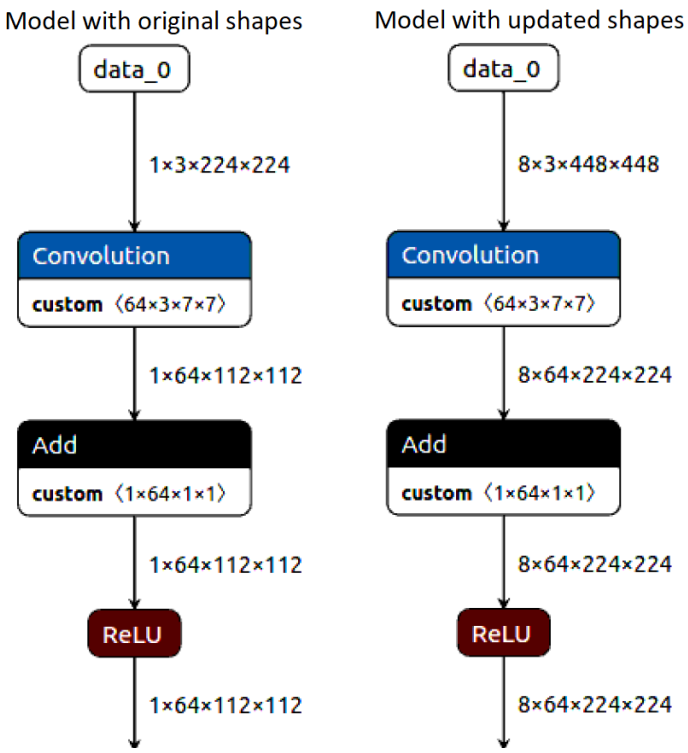
假设模型的原始输入为：[1x3x224x224

修改为：8x3x448x448

只需要调用 reshape()方法，一行代码即可完成：

```
model.reshape([8, 3, 448, 448])
```

模型输入形状修改前后对比示意图如下：



上述情形是从一种静态输入形状修改为另外一种静态输入形状。OpenVINO™ 的 reshape()方法还支持动态形状(不定长)的推理输入。

假设有模型，如下所示：

把输入格式：[?x3x640x640]
修改为:[4x3x640x?]
其中？表示不定长，可以用如下代码：

```
forinput_layerinmodel.inputs:
    input_shape=input_layer.partial_shape
    input_shape[0] =4
    input_shape[3] =-1
    model.reshape({input_layer: input_shape})
```

其中**-1** 表示不定长！

注意：修改输入/动态输入在 iGPU 上暂时还无法被支持，所以 AUTO 模式下修改以后可能会遇到推理失败的情况！这块建议参考官方文档说明。

05 模型推理

Python SDK 支持两种方式，一种是通过 complied model 直接推理，这种方式跟很多深度学习的推理方式非常类似，另外一种方式是先通过 compiled model 创建 InferRequest 实例对象，然后调用 infer 方法完成推理，个人推荐第一种方法，简单快捷明了，希望 OpenVINO 以后直接把第二种方法给 disable 了，同时官方的教程也更新为第一种方式推理！两种推理方式代码示意，

方法一：

```
results =compiled_model(input_data)
```

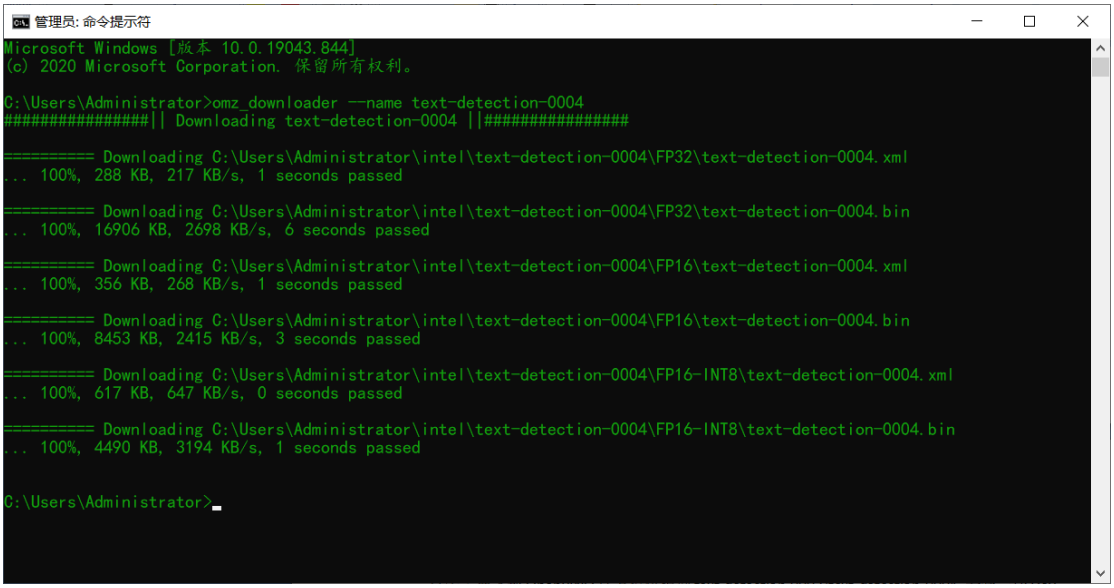
方法二：

```
infer_request=compiled_model.create_infer_request()
infer_request.infer()
output_tensor=infer_request.get_output_tensor()
```

06 场景文字检测模型演示

下面是基于 OpenVINO™2022.1 版本最新 Python SDK 调用 OpenVINO 官方提供的自带场景文字检测模型，完成了一个简单的场景文字检测 OpenVINO2022 版本 Python SDK 演示。

模型下载：
首先需要下载 OpenVINO 官方提供模型 text-detection-0003/text-detection-0004，安装完成 Dev Tools 前提下，下载模型只要执行如下命令行：
omz_downloader --name text-detection-0004



其--name 参数指定要下载模型名称为 text-detection-0004。

模型的输入与输出：
该模型是基于 PixelLink 构建，实现场景文字检测，其中它的输入图像格式要求如下：
维度：BxCxHxW=1x3x768x1280
通道顺序：BGR

模型输出包含两个输出层，名称与格式分布如下：
name: "model/link_logits_/add", shape: [1x16x192x320]
name: "model/segm_logits/add", shape: [1x2x192x320]
这里我只解析了 model/segm 的信息，获取文本与非文本分割的 mask，可以说是偷懒了，但是从实际效果看，已经非常好了。

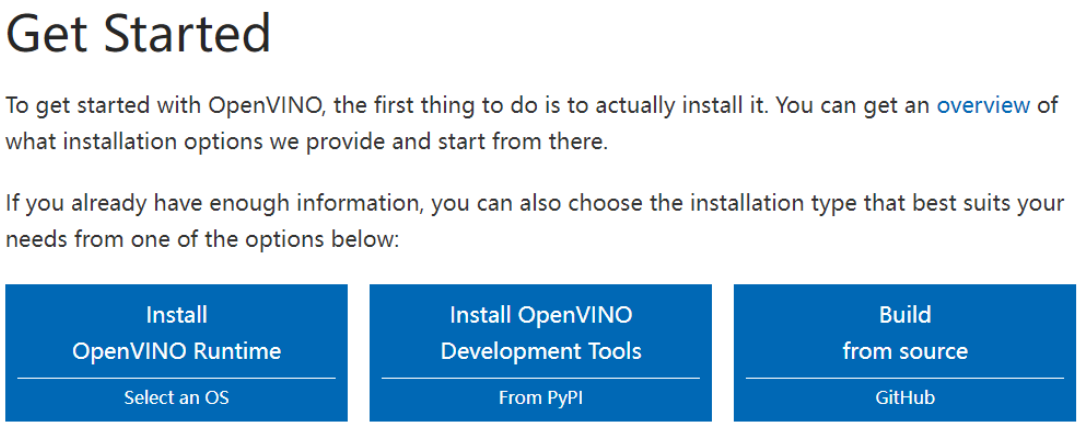
代码实现部分，我封装成了一个单独的类，在初始化中加载模型，调用模型推理之后返回每个文字区域的外接矩形框。完整的类代码与测试代码如下：

```
import cv2 as cv
importnumpyas np
fromopenvino.runtimeimport Core


classOpenVINOTextDetector():
    def__init__(self):
        # 预处理设置 0 - 放缩, 1- 保持比例
        self.preprocess_img_mode=0

        # 插值方式, 0 - 最近邻 1 - 线性, 2 - 立方
        self.interpolate_mode=cv.INTER_LINEAR
```

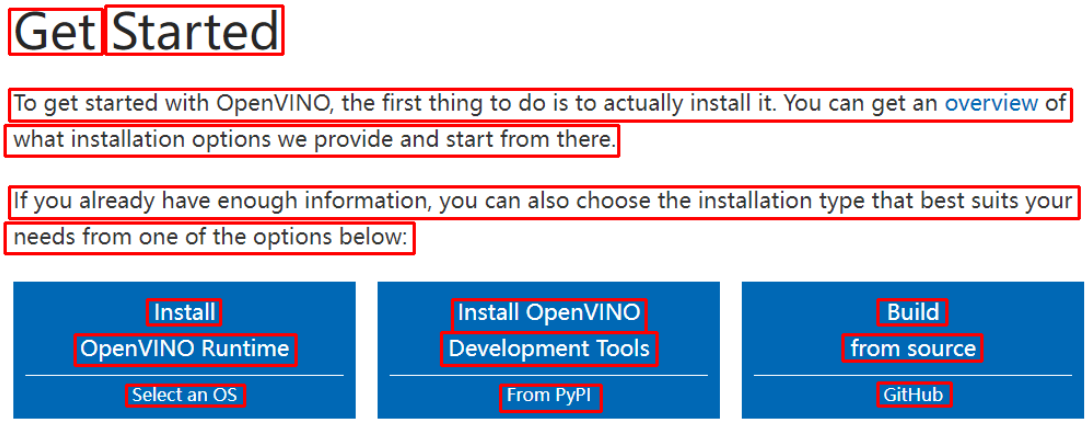
```
        self.score_threshold=0.5
        self.init_text_detector()

    definit_text_detector(self):
        ie= Core()
        model =
ie.read_model(model="D:/python/openvm/models/text-detection-0004.xml",
                                  weights="D:/python/openvm/models/text-d
etection-0004.bin")

        self.compiled_model=ie.compile_model(model=model,
device_name="CPU")
        self.input_layer=next(iter(self.compiled_model.inputs))

        # model/segm_logits/add, model/link_logits_/add
        # 1, 192, 320, 2
        it =iter(self.compiled_model.outputs)
        self.output_layer1 =next(it)
        self.output_layer2 =next(it)

    defformat_input(self, image):
        n, h, w, c =self.input_layer.shape
        ifself.preprocess_img_mode==0:
            resized_image=cv.resize(image, (w, h),
interpolation=self.interpolate_mode)
        ifself.preprocess_img_mode==1:
            resized_image=np.zeros((h, w, 3), np.uint8)
            rows, cols, _ =image.shape
            rate_y= rows / h
            rate_x= cols / w
            ifrate_y<=1.0andrate_x<=1.0:
                resized_image[0:rows, 0:cols] = image
            ifrate_y>1.0orrate_x>1.0:
                rate_max=max(rate_x, rate_y)
                rh =int(rows /rate_max)
                rw=int(cols /rate_max)
                rimg=cv.resize(image, (rw, rh))
                resized_image[0:rh, 0:rw] =rimg
        input_data=np.expand_dims(resized_image, 0).astype(np.float32)
        returninput_data

    defexec(self, image:np.ndarray) ->dict:
        ih, iw, _ =image.shape
        input_data=self.format_input(image)
```

```
        outputs =self.compiled_model([input_data])
        out1 =np.squeeze(outputs[self.output_layer1])
        _, oh, ow, _ = outputs[self.output_layer1].shape
        pixel_mask=np.zeros((oh, ow), dtype=np.uint8)

        for row inrange(oh):
            for col inrange(ow):
                pv2 = out1[row, col, 1]
                if pv2 >self.score_threshold:
                    pixel_mask[row, col] =255
        mask =cv.resize(pixel_mask, (iw, ih))
        contours, hiearchy=cv.findContours(mask, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
        text_boxes= []
        forcntinrange(len(contours)):
            x, y, w, h =cv.boundingRect(contours[cnt])
            text_boxes.append((x, y, w, h))
        returntext_boxes


if__name__=="__main__":
    mt =OpenVINOTextDetector()
    image =cv.imread("D:/openvino_test.png")
    boxes =mt.exec(image)
    for box in boxes:
        x, y, w, h = box
        cv.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2, 8, 0)

    cv.imshow("OpenVINO2022 Python SDK -Text Detect Demo", image)
    cv.imwrite("D:/result.png",image)
    cv.waitKey(0)
    cv.destroyAllWindows()
```

输入图像：



场景文字检测结果：



一个 trick 的地方，当你修改为动态输入的时候有时候会遇到这个错误：

ValueError: get_shape was called on a descriptor::Tensor with dynamic shape

```
D:\>python text_detect_openvino.py
Traceback (most recent call last):
  File "text_detect_openvino.py", line 79, in <module>
    boxes = mt.exec(image)
  File "text_detect_openvino.py", line 56, in exec
    input_data = self.format_input(image)
  File "text_detect_openvino.py", line 35, in format_input
    n, h, w, c = self.input_layer.shape
ValueError: get_shape was called on a descriptor::Tensor with dynamic shape
```

这个时候你**需要把 Core 初始化为全局类属性变量**或者一个全局变量一般情况下就会修正这个错误，这个是使用动态输入推理最有玄机的地方！原因我也解释不清楚，也许 OpenVINO 还需要持续改进，提升开发者满意度！

范例代码下载链接：https://gitee.com/ppov-nuc/resnet_pretrained_demo/blob/master/text_detect_openvino.py