通过 OpenVINO™ Model Server 和 TensorFlow Serving 简化部署

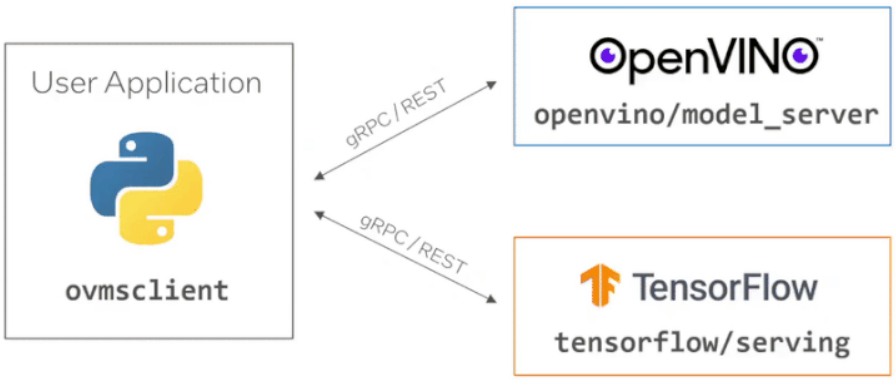
作者: ***Milosz Zeglarski***

翻译: ***李翊玮***

介绍

在这篇博客中，您将学习如何使用 OpenVINO Model Server 中的 gRPC API 对 JPEG 图像执行推理。Model servers 在顺利地将模型从开发环境引入生产方面发挥着重要作用。它们通过网络终结点提供模型，并公开用于与之交互的 API。提供模型后，需要一组函数才能从我们的应用程序调用 API。

OpenVINO™ Model Server 和 TensorFlow Serving 共享相同的前端 API，这意味着我们可以使用相同的代码与两者进行交互。对于 Python 开发人员来说，典型的起点是使用 tensorflow-serving-api pip 包。不幸的是，由于这个软件包包含 TensorFlow 作为依赖项，因此它的占用空间相当大。



迈向轻量级客户之路

由于**tensorflow-serving- api** 及其依赖项的大小约为 **1.3GB**，因此我们决定创建一个轻量级客户端，仅具有执行 API 调用所需的功能。在最新版本的 OpenVINO Model Server 中，我们引入 Model了Python客户端库的预览版 - **ovmsclient**。这个新软件包及其所有依赖项都不到 **100MB** - 使其 比 **tensorflow-serving-api 小 13 倍**。

tensorflow-serving-api

Package	Version
absl-py	0.15.0
astunparse	1.6.3
cached-property	1.5.2
cachetools	4.2.4
certifi	2021.10.8
charset-normalizer	2.0.9
clang	5.0
dataclasses	0.8
flatbuffers	1.12
gast	0.4.0
google-auth	1.35.0
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
grpcio	1.42.0
h5py	3.1.0
idna	3.3
importlib-metadata	4.8.2
keras	2.6.0
Keras-Preprocessing	1.1.2
Markdown	3.3.6
numpy	1.19.5
oauthlib	3.1.1
opt-einsum	3.3.0
pip	21.3.1
pkg_resources	0.0.0
protobuf	3.19.1
pyasn1	0.4.8
pyasn1-modules	0.2.8
requests	2.26.0
requests-oauthlib	1.3.0
rsa	4.8
setuptools	59.6.0
six	1.15.0
tensorboard	2.6.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.0
tensorflow	2.6.2
tensorflow-estimator	2.6.0
tensorflow-serving-api	2.6.2
termcolor	1.1.0
typing-extensions	3.7.4.3
urllib3	1.26.7
Werkzeug	2.0.2
wheel	0.37.0
wrapt	1.12.1
zipp	3.6.0

ovmsclient

Package	Version
certifi	2021.10.8
charset-normalizer	2.0.9
decorator	5.1.0
grpcio	1.43.0
idna	3.3
numpy	1.19.5
ovmsclient	0.2
pip	21.3.1
pkg_resources	0.0.0
protobuf	3.19.1
requests	2.26.0
setuptools	59.6.0
six	1.16.0
urllib3	1.26.7
validators	0.18.2
wheel	0.37.0

Python 环境比较。请注意与客户端一起安装的包数的差异。

除了具有更大的二进制大小之外，在应用程序中导入**tensorflow-serving- api** 还会消耗更多内存。请参阅下面为执行相同操作的 Python 脚本运行 top 命令的结果 —

一个使用 **tensorflow-serve-api**，另一个使用内存使用情况比较。

包	VIRT [KB]	RES [KB]	SHR [KB]
tensorflow-serving-api	4543760	293516	155708
ovmsclient	3500296	52008	23040

请注意 RES [KB] 列中的差异，该列指示任务使用的物理内存量。

导入占用空间较大的包也会增加初始化时间，这是使用轻量级客户端的另一个好处。新的Python客户端包也比**tensorflow-serving- api** 更易于使用，因为它为与模型服务器的端到端交互提供了实用程序。使用 **ovmsclient** 时，应用程序开发人员不需要知道服务器 API 的详细信息。该软件包为交互的每个阶段提供了一组方便的功能 -从设置连接和进行 API 调用，到以标准格式解压缩结果。以前，开发人员需要知道哪个服务接受什么类型的请求，或者如何手动准备请求和处理响应。

让我们使用 ovmsclient
要使用 ResNet-50 图像分类模型运行预测，请使用该模型部署 OpenVINO Model Server。您可以使用以下命令执行此操作：

```
docker run -d --rm -p 9000:9000 openvino/model_server:latest \
--model_name resnet --model_path gs://ovms-public-eu/resnet50-binary \
--layout NHWC:NCHW --port 9000
```

此命令使用从 **Google Cloud Storage** 上的公共存储桶下载的 ResNet-50 模型启动服务器。使用模型服务器侦听 端口 9000 上的 gRPC 调用，您可以开始与服务器交互。接下来，让我们使用 pip 安装 **ovmsclient** 包：

```
pip3 install ovmsclient
```

在运行客户端之前，下载图像进行分类和相应的ImageNet标签文件，以解释预测结果：

```
wget
https://raw.githubusercontent.com/openvinotoolkit/model_server/main/demos/
common/static/images/zebra...
wget
```

```
https://raw.githubusercontent.com/openvinotoolkit/model_server/main/demos/common/python/classes.py
```

用于预测的斑马的图片

步骤 1：创建与服务器的 gRPC 连接：

现在，您可以打开 **Python interpreter**并创建 与模型服务器的 gRPC 连接。

```
$ python3
Python 3.6.9 (default, Jul 17 2020, 12:50:27)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>> from ovmsclient import make_grpc_client
>> client = make_grpc_client("localhost:9000")
```

步骤 2：请求模型元数据：

客户端对象有三种方法： get_model_status、get_model_metadata 和预测 。
要创建有效的推理请求，您需要知道模型输入。为此，让我们请求模型元数据：

```
>> client.get_model_metadata(model_name= "resnet")
{'model_version': 1, 'inputs': {'0': {'shape': [1, 224, 224, 3], 'dtype':
'DT_FLOAT'}}, 'outputs': {'1463': {'shape': [1, 1000], 'dtype': 'DT_FLOAT
'}}}
```

步骤 3：将 JPEG 图像发送到服务器：

从模型元数据中，我们了解到模型有一个输入和一个输出。输入名称为“0”，它需要形状为 （1，224，224，3）的数据，并带有浮点数据类型。

现在，您拥有了运行推理所需的所有信息。让我们使用在上一步中下载的斑马的图像。在

此示例中，我们将使用模型服务器二进制输入功能，该功能只需要加载 JPEG 并请求对编码的二进制文件进行预测 - 使用二进制输入时不需要预处理。

```
>> with open("zebra.jpeg", "rb") as f:
... img = f.read()
...
>> output = client.predict(inputs={ "0": img}, model_name= "resnet")
>> type(output)
<class 'numpy.ndarray'>
>> output.shape
(1, 1000)
```

步骤 4：将输出映射到 imagenet 类：

预测成功返回，输出形状为 numpy ndarray （1， 1000） -
与“输出”部分的模型元数据中描述的相同。下一步是解释模型输出并提取分类结果。

输出的形状为 （1， 1000 ），其中第一个维度表示批大小（处理的图像数），第二个维度表示图像属于每个 ImageNet 类的概率。若要获取分类结果，需要获取输出第二维度中最大值的索引。然后 使用 imagenet_classes 上一步中下载的 classes.py 字典执行索引号到类名的映射并查看结果。

```
>>> import numpy as np
>>> from classes import imagenet_classes
>>> result_index = np.argmax(output[0])
>>> imagenet_classes[result_index]
'zebra'
```

结论
新的 **ovmsclient** 包比 **tensorflow-serving-api**
更小，消耗更少的内存，并且更易于使用。在这篇博客中，我们学习了如何获取模型元数据，以及如何通过 OpenVINO 模型服务器中的 gRPC 接口对二进制编码的 JPEG
图像运行预测。

查看有关使用 NumPy 数组运行预测、检查模型状态以及在 GitHub 上使用 REST API
的更多详细信息示例：https://github.com/openvinotoolkit/model_server/tree/main/client/python/ovmsclient/samples

要了解有关 **ovmsclient** 功能的更多信息，请参阅 API 文档：
https://github.com/openvinotoolkit/model_server/blob/main/client/python/ovmsclient/lib/docs/README.m...

这是客户端库的第一个版本。它将 随着时间的推移而发展，但 已经能够使用 OpenVINO Model Server和TensorFlow Serving 运行预测，只需一个最小的 Python 包。有问题或建议吗？请在 GitHub 上提出问题。