深度之眼
deepshare.net

模型部署

导师：GAUSS

深度之眼
deepshare.net

目录

深度之眼
deepshare.net

模型部署简介

模型部署简介

- TensorFlow Serving

通过 tensorflow-serving 可以加载模型后提供网络接口API服务，通过任意编程语言发送网络请求都可以获取模型预测结果。

- TensorFlow Lite

通过 tensorflow-lite 可以在移动和嵌入式设备上加载并运行TensorFlow模型。

- TensorFlow in JavaScript

通过 tensorflow-js 可以用javascrip脚本加载模型并在浏览器中运行模型。

深度之眼
deepshare.net

准备模型文件

如何保存

- 导出模型（tf.saved_model.save）

```
mobilenet_save_path = os.path.join(tmpdir, "mobilenet/1/")
tf.saved_model.save(pretrained_model, mobilenet_save_path)
```

保存路径遵循TensorFlow Serving使用的约定，其中最后一个路径组件（1/此处）是模型的版本号-它允许Tensorflow Serving之类的工具推断相对新鲜度。

模型文件讲解

tf.keras.Model会自动指定服务签名，什么是签名？

SignatureDef 是对应了一种图的输入和输出。

```
MetaGraphDef with tag-set: 'serve' contains the following SignatureDefs:

signature_def['__saved_model_init_op']:
  The given SavedModel SignatureDef contains the following input(s):
  The given SavedModel SignatureDef contains the following output(s):
    outputs['__saved_model_init_op'] tensor_info:
        dtype: DT_INVALID
        shape: unknown_rank
        name: NoOp
  Method name is:

signature_def['serving_default']:
  The given SavedModel SignatureDef contains the following input(s):
    inputs['contents'] tensor_info:
        dtype: DT_INT32
        shape: (-1, 800)
        name: serving_default_contents:0
  The given SavedModel SignatureDef contains the following output(s):
    outputs['tf_op_layer_Softmax'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 10)
        name: StatefulPartitionedCall:0
  Method name is: tensorflow/serving/predict
```

SavedModel命令行界面的详细信息

您可以使用SavedModel命令行界面（CLI）检查并执行SavedModel。例如，您可以使用CLI检查模型的SignatureDef。CLI使您可以快速确认输入的Tensor dtype和shape与模型匹配。此外，如果要测试模型，则可以使用CLI进行健全性检查，方法是传入各种格式（例如Python表达式）的示例输入，然后获取输出。

SavedModel命令行界面的详细信息

show 命令

该show命令使您可以按层次结构顺序检查SavedModel的内容

usage: saved_model_cli show [-h] --dir DIR [--all]

[--tag_set TAG_SET] [--signature_def SIGNATURE_DEF_KEY]

SavedModel命令行界面的详细信息

run 命令

调用该run命令以运行图形计算，传递输入，然后显示（并选择保存）输出。语法如下：

usage: saved_model_cli run [-h] --dir DIR --tag_set TAG_SET --signature_def SIGNATURE_DEF_KEY [--inputs INPUTS]

[--input_exprs INPUT_EXPRS] [--input_examples INPUT_EXAMPLES] [--outdir OUTDIR] [--overwrite] [--tf_debug]

SavedModel命令行界面的详细信息

该run命令提供了以下三种将输入传递给模型的方式：

--inputs 选项使您可以在文件中传递numpy ndarray。

--input_exprs 选项使您可以传递Python表达式。

--input_examples选项使您能够通过tf.train.Example。

```
saved_model_cli run --dir ./keras_saved_graph --tag_set serve \
--signature_def serving_default \
--input_exprs 'flatten_input=np.ones((1,800))'
```

Tf.serving需要的模型文件

一个比较完整的SavedModel模型包含以下内容：

saved_model.pb是MetaGraphDef，它包含图形结构。variables文件夹保存训练所习得的权重。assets文件夹可以添加可能需要的外部文件，assets.extra是一个库可以添加其特定assets的地方。

```
├── assets/ # 所需的外部文件，例如说初始化的词汇表文件，一般无
├── assets.extra/ # TensorFlow graph 不需要的文件, 例如说给用户知晓的如何使用SavedModel的信息. Tensorflow 不使用这个目录下的文件。
├── saved_model.pb # 保存的是MetaGraph的网络结构
└── variables # 参数权重，包含了所有模型的变量(tf.Variable objects)参数
    ├── variables.data-00000-of-00001
    └── variables.index
```

深度之眼
deepshare.net

Docker的简单使用

Docker的简单使用

常见docker命令：

docker安装：参考https://www.runoob.com/docker/ubuntu-docker-install.html

docker ps:查看正在运行的容器

docker images:查看已下载的景象

docker stop 8779b492e4aa：停止正在运行的ID为8779b492e4aa的容器，ID可以通过docker ps查看

深度之眼
deepshare.net

TensorFlow Serving简介

TensorFlow Serving

TensorFlow Serving是一个针对机器学习模型的灵活，高性能的服务系统，专为生产环境而设计。使用TensorFlow Serving可以轻松部署新算法和实验，同时保持相同的服务器体系结构和API。TensorFlow Serving提供与TensorFlow模型的现成集成，但可以轻松扩展以服务其他类型的模型和数据。

TensorFlow与Docker服务

使用TensorFlow Serving的最简单方法之一是使用 Docker。

```
# 下载TensorFlow Serving Docker映像和仓库
# 演示模型的位置
# 启动TensorFlow Serving容器并打开REST API端口
# 使用预测API查询模型
# 返回=> {“预测”: [2.5, 3.0, 4.5]}
docker pull tensorflow/serving

git clone https://github.com/tensorflow/serving

TESTDATA="$(pwd)/serving/tensorflow_serving/servables/tensorflow/testdata"

docker run -t --rm -p 8501:8501 \
    -v "$TESTDATA/saved_model_half_plus_two_cpu:/models/half_plus_two" \
    -e MODEL_NAME=half_plus_two \
    tensorflow/serving &

curl -d '{"instances": [1.0, 2.0, 5.0]}' \
    -X POST http://localhost:8501/v1/models/half_plus_two:predict
```

拉取服务镜像

安装Docker之后，您可以通过运行以下命令来获取最新的TensorFlow Serving Docker映像：

```
docker pull tensorflow/serving
```

这将拉下安装了TensorFlow Serving的最小Docker。

请参阅Docker Hub tensorflow / serving存储库，以获取其他可以拉取的镜像版本。

http://hub.docker.com/r/tensorflow/serving/tags/

运行镜像

投放的镜像（CPU和GPU）都具有以下属性：

- 暴露给gRPC的端口8500
- REST API公开的端口8501
- 可选的环境变量MODEL_NAME（默认为model）
- 可选的环境变量MODEL_BASE_PATH（默认为/models）

运行镜像

你要做的是运行Docker容器，将容器的端口发布到主机的端口，然后将主机的路径安装到SavedModel到容器期望模型的位置。

```
docker run -p 8501:8501 \
  --mount type=bind,source=/path/to/my_model/,target=/models/my_model \
  -e MODEL_NAME=my_model -t tensorflow/serving
```

运行镜像

在这种情况下，我们启动了一个Docker容器，将REST API端口8501发布到主机的端口8501，并采用了一个名为my_model的模型并将其绑定到默认模型基本路径（$ {MODEL_BASE_PATH} / $ {MODEL_NAME} = / models / my_model）。最后，我们用my_model填充了环境变量MODEL_NAME，并将MODEL_BASE_PATH保留为其默认值。

运行镜像

如果要发布gRPC端口，可以使用-p 8500:8500。您可以同时打开gRPC和REST API端口，或者选择仅打开一个端口。

```
docker run -p 8500:8500 -p 8501:8501 \
  --mount type=bind,source=/path/to/my_model/,target=/models/my_model \
  --mount type=bind,source=/path/to/my/models.config,target=/models/models.config \
  -t tensorflow/serving --model_config_file=/models/models.config
```

创建自己的镜像

参考：https://www.tensorflow.org/tfx/serving/docker

深度之眼
deepshare.net

实战九：TensorFlow Serving 实战

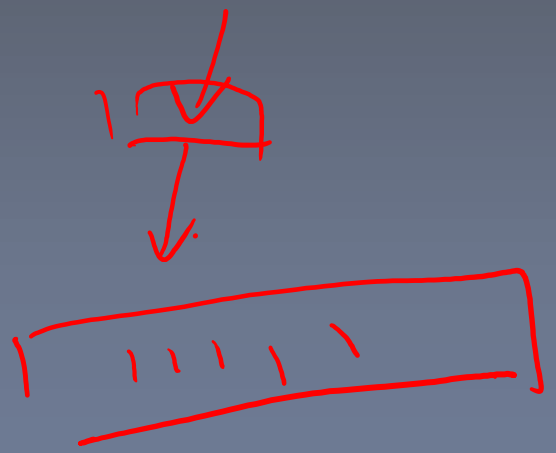
TensorFlow Serving实战


```
# 下载TensorFlow Serving Docker映像和仓库
# 演示模型的位置
# 启动TensorFlow Serving容器并打开REST API端口
# 使用预测API查询模型
# 返回=> {“预测”：[2.5，3.0，4.5]}
docker pull tensorflow/serving

git clone https://github.com/tensorflow/serving

TESTDATA="$(pwd)/serving/tensorflow_serving/servables/tensorflow/testdata"

docker run -t --rm -p 8501:8501 \
    -v "$TESTDATA/saved_model_half_plus_two_cpu:/models/half_plus_two" \
    -e MODEL_NAME=half_plus_two \
    tensorflow/serving &

curl -d '{"instances": [1.0, 2.0, 5.0]}' \
    -X POST http://localhost:8501/v1/models/half_plus_two:predict
```

新闻文本分类任务部署

在实战四任务，对于新闻文本分类任务进行部署：

- 保存模型
- saved_model_cli查看模型的输入输出
- 验证模型输入输出
- 进行tf.serving模型部署
- 请求模型服务



客户端调用 TensorFlow Serving 部署的模型

中间件Flask

这个中间件Flask的作用是连接了模型和request，并且在这中间件里面处理数据，变成model可以读取的形式，如将文本数据转化为数字传入。

深度之眼
deepshare.net

总结

本节小结

Summary

模型部署	模型部署简介	
	准备模型文件	
	Docker的简单使用	
	TensorFlow Serving简介	
	TensorFlow Serving实战	

结语

我 说

看过千万代码，不如实践一把！

深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q： 2677693114

公众号

客服微信