Batch Inference with TorchServe

链接地址：

- Batch Inference with TorchServe

Contents of this document:

Introduction

批量推理是一个聚合推理请求的过程，并通过ML/DL框架一次性发送这些聚合的请求进行推理。TorchServe被设计为原生支持对传入的推理请求进行批处理。这种功能使你能够最佳地使用你的主机资源，因为大多数ML/DL框架都是为批量请求而优化的。对主机资源的这种优化使用反过来又减少了使用TorchServe托管推理服务的运营费用。

在本文档中，我们展示了在本地提供模型或使用 docker 容器时如何在 Torchserve 中使用批量推理的示例。

Prerequisites

在进入本文档之前，请阅读以下文档：

1. What is TorchServe ?
2. What is custom service code ?

Batch Inference with TorchServe's default handlers

TorchServe 的默认处理程序支持开箱即用的批量推理，除了 text_classifier 处理程序。

Batch Inference with TorchServe using ResNet-152 model

为了支持批量推理，TorchServe 需要以下内容：

1. TorchServe 模型配置：使用“POST /models”管理 API 或 config.properties 中的设置配置 `batch_size` 和 `max_batch_delay`。 TorchServe 需要知道模型可以处理的最大批处理大小以及 TorchServe 应该等待填充每个批处理请求的最长时间。
2. 模型处理程序代码：TorchServe 需要模型处理程序来处理批量推理请求。

有关具有批处理的自定义模型处理程序的完整工作示例，请参阅 Hugging face transformer generalized handler.

TorchServe Model Configuration

从 Torchserve 0.4.1 开始，有两种方法可以配置 TorchServe 以使用批处理功能：

1. 通过 POST /models API 提供批量配置信息。
2. 通过配置文件config.properties提供批量配置信息。

我们感兴趣的配置属性如下：

1. `batch_size`

   这是模型预期处理的最大批量大小。

2. `max_batch_delay`

   这是TorchServe等待接收 batch_size 数量的请求的最大批量延迟时间，以 `ms` 为单位。如果TorchServe在这个计时器结束前没有收到 `batch_size` 数量的请求，它将把收到的任何请求发送到模型处理程序。

让我们看一个通过管理 API 使用此配置的示例：

```
# The following command will register a model "resnet-152.mar" and configure TorchServe
to use a batch_size of 8 and a max batch delay of 50 milliseconds.

$ curl -X POST "localhost:8081/models?url=resnet-
152.mar&batch_size=8&max_batch_delay=50"
```

以下是通过 config.properties 使用此配置的示例：

```
# The following command will register a model "resnet-152.mar" and configure TorchServe
to use a batch_size of 8 and a max batch delay of 50 milli seconds, in the
config.properties.

models={\
  "resnet-152": {\
    "1.0": {\
        "defaultVersion": true,\
        "marName": "resnet-152.mar",\
        "minWorkers": 1,\
        "maxWorkers": 1,\
```

```
        "batchSize": 8,\
        "maxBatchDelay": 50,\
        "responseTimeout": 120\
    }\
  }\
}
```

这些配置在TorchServe和模型的自定义服务代码（又称处理程序代码）中都会用到。TorchServe将批量相关的配置与每个模型联系起来。然后，前端试图聚集批量大小的请求数量并将其发送到后端。

Demo to configure TorchServe ResNet-152 model with batch-supported model

在本节中，让我们启动模型服务器并启动 Resnet-152 模型，该模型使用默认的 image_classifier 处理程序进行批量推理。

Setup TorchServe and Torch Model Archiver

首先，按照主要的 Readme，安装所有需要的软件包，包括Torchserve。

Batch inference of Resnet-152 configured with management API

- 启动模型服务器。在此示例中，我们正在启动模型服务器，以在推理端口8080和管理端口8081上运行。

```
$ cat config.properties
...
inference_address=http://0.0.0.0:8080
management_address=http://0.0.0.0:8081
...
$ torchserve --start --model-store model_store
```

- Verify that TorchServe is up and running

```
$ curl localhost:8080/ping
{
  "status": "Healthy"
}
```

- 现在让我们启动resnet-152模型，我们已经建立了它来处理批量推理。因为这是一个例子，我们将启动1个工作器，它处理的批处理量为8，最大_batch_delay为10ms。

```
$ curl -X POST "localhost:8081/models?
url=https://torchserve.pytorch.org/mar_files/resnet-152-
batch_v2.mar&batch_size=3&max_batch_delay=10&initial_workers=1"
{
  "status": "Processing worker updates..."
}
```

- 验证 workers 是否正确开始。

```
$ curl http://localhost:8081/models/resnet-152-batch_v2
```


```
[
  {
    "modelName": "resnet-152-batch_v2",
    "modelVersion": "2.0",
    "modelUrl": "https://torchserve.pytorch.org/mar_files/resnet-152-
batch_v2.mar",
    "runtime": "python",
    "minWorkers": 1,
    "maxWorkers": 1,
    "batchSize": 3,
    "maxBatchDelay": 10,
    "loadedAtStartup": false,
    "workers": [
      {
        "id": "9000",
        "startTime": "2021-06-14T23:18:21.793Z",
        "status": "READY",
        "memoryUsage": 1726554112,
        "pid": 19946,
        "gpu": true,
        "gpuUsage": "gpuId::0 utilization.gpu [%]::0 % utilization.memory [%]::0 %
memory.used [MiB]::678 MiB"
      }
    ]
  }
]
```


- 现在进行功能测试：
  - 获取图像测试此服务

    ```
    $ curl -LJO
    https://github.com/pytorch/serve/raw/master/examples/image_classifier/kitten.j
    pg
    ```

  - 运行推理以测试模型

```
$ curl http://localhost:8080/predictions/resnet-152-batch_v2 -T kitten.jpg
  {
      "tiger_cat": 0.5848360657691956,
      "tabby": 0.3782736361026764,
      "Egyptian_cat": 0.03441936895251274,
      "lynx": 0.0005633446853607893,
      "quilt": 0.0002698268508538604
  }
```

Batch inference of Resnet-152 configured through config.properties

- 在这里，我们首先在config.properties中设置batch_size和max_batch_delay，确保mar文件位于model-store中，models设置中的版本与创建的mar文件的版本一致。要阅读更多关于配置的信息，请参考此文件。\

```
load_models=resnet-152-batch_v2.mar
models={\
  "resnet-152-batch_v2": {\
    "2.0": {\
        "defaultVersion": true,\
        "marName": "resnet-152-batch_v2.mar",\
        "minWorkers": 1,\
        "maxWorkers": 1,\
        "batchSize": 3,\
        "maxBatchDelay": 5000,\
        "responseTimeout": 120\
    }\
  }\
}
```

- 然后将通过使用 --ts-config 标志传递 config.properties 来启动 Torchserve

```
$ torchserve --start --model-store  model_store  --ts-config  config.properties
```

- 验证TorchServe是否启动和运行

```
$ curl localhost:8080/ping
{
  "status": "Healthy"
}
```

- 验证 workers 是否正确开始。

```
$ curl http://localhost:8081/models/resnet-152-batch_v2
```

```
[
  {
    "modelName": "resnet-152-batch_v2",
    "modelVersion": "2.0",
```

```
    "modelUrl": "resnet-152-batch_v2.mar",
    "runtime": "python",
    "minWorkers": 1,
    "maxWorkers": 1,
    "batchSize": 3,
    "maxBatchDelay": 5000,
    "loadedAtStartup": true,
    "workers": [
      {
        "id": "9000",
        "startTime": "2021-06-14T22:44:36.742Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 19116,
        "gpu": true,
        "gpuUsage": "gpuId::0 utilization.gpu [%]::0 % utilization.memory [%]::0 %
memory.used [MiB]::678 MiB"
      }
    ]
  }
]
```


- 现在测试服务：
  - 获取测试图片

    ```
    $ curl -LJO
    https://github.com/pytorch/serve/raw/master/examples/image_classifier/kitten.j
    pg
    ```

  - 运行推理来进行模型测试

    ```
    $ curl http://localhost:8080/predictions/resnet-152-batch_v2 -T kitten.jpg
      {
          "tiger_cat": 0.5848360657691956,
          "tabby": 0.3782736361026764,
          "Egyptian_cat": 0.03441936895251274,
          "lynx": 0.0005633446853607893,
          "quilt": 0.0002698268508538604
      }
    ```

Demo to configure TorchServe ResNet-152 model with batch-supported model using Docker

在这里，我们在使用Docker Contianers服务模型时显示了如何在批处理推理支持下注册模型。我们在config.properties中设置batch_size和max_batch_delay，类似于上一节，该部分正在 dockered_entrypoint.sh 使用。

Batch inference of Resnet-152 using docker container

- 在 dockered_entrypoint.sh 中引用的config.properties中设置批处理batch_size和max_batch_delay。

```
inference_address=http://0.0.0.0:8080
management_address=http://0.0.0.0:8081
metrics_address=http://0.0.0.0:8082
number_of_netty_threads=32
job_queue_size=1000
model_store=/home/model-server/model-store
load_models=resnet-152-batch_v2.mar
models={\
  "resnet-152-batch_v2": {\
    "1.0": {\
        "defaultVersion": true,\
        "marName": "resnet-152-batch_v2.mar",\
        "minWorkers": 1,\
        "maxWorkers": 1,\
        "batchSize": 3,\
        "maxBatchDelay": 100,\
        "responseTimeout": 120\
    }\
  }\
}
```

- 从这里构建目标的docker镜像，这里我们使用gpu镜像。

```
$ ./build_image.sh  -g  -cv  cu102
```

- 开始用容器为模型服务，并将config.properties传递给容器

```
$  docker run --rm -it --gpus all -p 8080:8080 -p 8081:8081 --name mar -v
/home/ubuntu/serve/model_store:/home/model-server/model-store  -v $ path to
config.properties:/home/model-server/config.properties  pytorch/torchserve:latest-
gpu
```

- Verify that the workers were started properly.

```
$ curl http://localhost:8081/models/resnet-152-batch_v2
```

```
[
  {
    "modelName": "resnet-152-batch_v2",
    "modelVersion": "2.0",
    "modelUrl": "resnet-152-batch_v2.mar",
    "runtime": "python",
    "minWorkers": 1,
    "maxWorkers": 1,
    "batchSize": 3,
    "maxBatchDelay": 5000,
    "loadedAtStartup": true,
    "workers": [
      {
```

```
        "id": "9000",
        "startTime": "2021-06-14T22:44:36.742Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 19116,
        "gpu": true,
        "gpuUsage": "gpuId::0 utilization.gpu [%]::0 % utilization.memory [%]::0 %
memory.used [MiB]::678 MiB"
      }
    ]
  }
]
```


- Now let's test this service.
  - Get an image to test this service

    ```
    $ curl -LJO
    https://github.com/pytorch/serve/raw/master/examples/image_classifier/kitten.j
    pg
    ```

  - Run inference to test the model

    ```
    $ curl http://localhost:8080/predictions/resnet-152-batch_v2 -T kitten.jpg
      {
          "tiger_cat": 0.5848360657691956,
          "tabby": 0.3782736361026764,
          "Egyptian_cat": 0.03441936895251274,
          "lynx": 0.0005633446853607893,
          "quilt": 0.0002698268508538604
      }
    ```