06-Flask部署模型推理的 CUDA out of memory 问题研究

问题描述：

- RuntimeError: CUDA out of memory...

参考资料：

- How to delete a Tensor in GPU to free up memory
- Difference between allocated and reserved CUDA memory
- Out of Memory (OOM) whrn repeatedly running large model
- RuntimeError: CUDA out of memory. Tried to allocate 12.50 MiB (GPU 0; 10.92 GiB total capacity; 8.57 MiB already allocated; 9.28 GiB free; 4.68 MiB cached)
- Solving "CUDA out of memory " Error

How to delete a Tensor in GPU to free up memory

演示事例：

```
import torch
t = torch.zeros([1024, 1024, 1024, 2], device='cuda:0')
del t
torch.cuda.empty_cache()
```

The GPU still has about 700M suage:

```
±-----------------------------±---------------------±------------------------------+
| 0 GeForce RTX 208... Off | 00000000:05:00.0 Off | N/A |
| 33% 49C P8 29W / 250W | 717MiB / 11016MiB | 0% Default |
±-----------------------------±---------------------±------------------------------+
```

A：

> **ptrblck:**
>
> 这 700MB 的设备内存被 CUDA 上下文用于 PyTorch 中的 CUDA 内核以及其他库（如 cudnn、NCCL 等），并且无法释放。

Q：

> **Flock1 Flock Anizak:**
>
> 如果我错了，请纠正我，但我加载一个图像，并将其转换为Torch tensor和cuda()。所以当我这样做并运行torch.cuda.memory_allocated()时，它从0到一些内存分配。但是，我用del删除图像，然后运行torch.cuda.reset_max_memory_allocated()和torch.cuda.empty_cache()，我看到torch.cuda.memory_allocated()没有变化。我应该怎么

做？

A:

ptrblck:

That's the right approach, which also works for me :

```
path = '...'
image = Image.open(path)

print(torch.cuda.memory_allocated())
> 0
print(torch.cuda.memory_reserved())
> 0

x = transformers.ToTensor()(image)
print(torch.cuda.memory_allocated())
> 0
print(torch.cuda.memory_reserved())
> 0

x = x.cuda()
print(torch.cuda.memory_allocated())
> 23068672
print(torch.cuda.memory_reserved())
> 23068672

del x
print(torch.cuda.memory_allocated())
> 0
print(torch.cuda.memory_reserved())
> 23068672

torch.cuda.empty_cache()
print(torch.cuda.memory_allocated())
> 0
print(torch.cuda.memory_reserved())
> 0
```

Q:

mouad marrakchi benazzouz:

有什么办法解决这个问题吗？我们在一个共享服务器上工作，有时我需要为其他用户释放gpu内存而不杀死整个内核。你的代码确实释放了保留的内存（torch.cuda.memory_reserved()返回0），但nvidia-smi仍然显示我的内核在占用内存。

PS：我使用 jupyter-lab，这就是为什么有时我的模型完成训练后仍然需要内核的原因。

A:

ptrblck:

nvidia-smi将显示所有进程的分配内存。如果你只运行PyTorch，那么CUDA上下文将仍然使用设备内存（~1GB，取决于GPU等），并且在不停止Python内核的情况下不能释放。

mouad marrakchi benazzouz:

谢谢你的答复。恐怕nvidia-smi显示的是我的笔记本占用的所有GPU内存。例如，如果我训练一个需要15GB GPU内存的模型，而我使用torch释放空间（按照你代码中的程序），torch.cuda.memory_reserved()将返回0，但nvidia-smi仍将显示15GB。

ptrblck:

nvidia-smi确实显示了所有分配的内存，所以如果它仍然显示15GB，那么一些应用程序仍然在使用它。如果你通过torch.cuda.memory_summary()没有看到任何内存使用情况（无论是分配的还是缓存的），那么另一个应用程序（或python内核）会使用设备内存。

Out of Memory (OOM) when repeatedly running large models

A:

josiahdavis:

这个操作顺序对我来说很有帮助，从GPU中移除参数和梯度:

1. delete objects
2. `gc.collect()`
3. `torch.cuda.empty_cache()`

奇怪的是，在删除对象（但没有调用gc.collect()或 empty_cache()）后，运行你的代码片段（for item in gc.garbage: print(item)）并没有打印出任何东西。

rlouf:

对不起，也许我没有完全说清楚：你需要在垃圾回收后立即运行它。

这显然困扰着其他用户，但我不能百分之百确定我们能轻易做些什么；你能不能试着用下面的函数初始化调度器，并告诉我你是否仍然有内存错误？(不使用gc.collect()).

```
def warmup_linear_schedule(optimizer, warmup_steps, t_total, last_epoch=-1):
    def lr_lambda(step):
        if step < warmup_steps:
            return float(step) / float(max(1, warmup_steps))
        return max(0.0, float(t_total - step) / float(max(1.0, t_total -
warmup_steps)))

    return LambdaLR(optimizer, lr_lambda, last_epoch=-1)
```

当我们对LambdaLR进行子类化时，可能会出现引用混淆的情况；使用闭包可以避免这种情况。

josiahdavis:

是的，它确实解决了我的问题! 我目前正在我的GPU上运行一些训练工作（感谢你的帮助！）。一旦我完成了这些工作，我将尝试关闭的方法，并在这个主题上与你联系。

我刚刚测试了一下，它和预期的一样有效。现在从GPU中删除参数和梯度时，不需要再调用gc.collect()。非常感谢@rlouf，我赞扬你在解决这个问题和更新你的例子方面的细心。

RuntimeError: CUDA out of memory. Tried to allocate 12.50 MiB (GPU 0; 10.92 GiB total capacity; 8.57 MiB already allocated; 9.28 GiB free; 4.68 MiB cached)

问题描述：

- CUDA Out of Memory 错误，但 CUDA 内存几乎为空，我目前正在对大量的文本数据（大约70GiB的文本）进行轻量级模型训练。为此，我正在使用集群上的一台机器（grid5000集群网络的'grele'）。

A:

EMarquer:

当我停止在RAM中存储预处理的数据时，这个问题就消失了。

@OmarBazaraa，我不认为你的问题与我的相同，因为：

- 我正在尝试分配 12.50 MiB，其中 9.28 GiB 可用
- 你试图分配195.25MB的数据，但还有170.14MB的可用数据。

根据我以前处理这个问题的经验，要么你没有释放CUDA内存，要么你试图在CUDA上放太多的数据。

我所说的没有释放CUDA内存，是指你可能在CUDA中仍有对不再使用的张量的引用。这些都会阻碍分配的内存通过删除张量而被释放。

Q:

faaizhashmi:

我试图在数据集 "点击率预测 "上运行 "场感知神经分解机"。Total size 为1.28GB。Training set entries = 16171587.

已经尝试的事情：

1. 改变 batch size 的大小
2. 删除/清理缓存

结果：无效。

我还发现所需的内存和分配的内存似乎会随着批量大小的变化而变化

A:

Anticonformiste:

我使用上下文管理器解决了这个问题，因为volatile标志已被废弃。

```
with torch.no_grad():
 # Your eval code here
```

Solving "CUDA out of memory" Error

如果你尝试在 GPU 上训练多个模型，你很可能会遇到类似下面的错误：

RuntimeError: CUDA out of memory. Tried to allocate 978.00 MiB (GPU 0; 15.90 GiB total capacity; 14.22 GiB already allocated; 167.88 MiB free; 14.99 GiB reserved in total by PyTorch)

我搜索了几个小时试图找到解决这个问题的最佳方法。以下是我的发现：

1. 使用这段代码可以看到内存使用情况（需要联网安装软件包）：

```
!pip install GPUtil

from GPUtil import showUtilization as gpu_usage
gpu_usage()
```

2. 使用此代码清除内存：

```
import torch
torch.cuda.empty_cache()
```

3. 也可以使用以下代码清理内存：

```
from numba import cuda
cuda.select_device(0)
cuda.close()
cuda.select_device(0)
```

4. 以下是释放 CUDA 内存的完整代码：

```
!pip install GPUtil

import torch
from GPUtil import showUtilization as gpu_usage
from numba import cuda

def free_gpu_cache():
    print("Initial GPU Usage")
    gpu_usage()

    torch.cuda.empty_cache()

    cuda.select_device(0)
    cuda.close()
    cuda.select_device(0)

    print("GPU Usage after emptying the cache")
    gpu_usage()

free_gpu_cache()
```

Response from JoelS:

就上下文而言，我使用的是 torchvision.models.detection.fastrcnn_resnet50_fpn（ResNet50）。在我的训练循环中，我已经添加了垃圾收集，如下所示。

```
def train_loop(files, batch_size, model):
 size = len(files)
 for batch in range(size//batch_size):
     im_paths = files[batch*batch_size : (batch+1)*batch_size]
     X,y = load_batch(im_paths)
     # Compute prediction and loss
     output = model(X, y)
     del X
     del y
     gc.collect()
     torch.cuda.empty_cache()
```

我可以看到现在的VRAM使用量在跳动，所以这在某种程度上肯定是在工作。Batch_size为8时仍会导致溢出。如果消耗量增加，我可能会尝试延迟输出，但它似乎在3.4GB左右徘徊。

Response from TommyPinkman:

> 如果你确定你的模型不是太大，不会超出内存，你可以在验证前加上 "with torch.no_grad()"，这是我的错误。

Response from Jordan Dahan:

> 对我来说，在每次网络培训之后，以下几行对我有帮助：
>
> ```
> model_mobileNet = model_mobileNet.to(cpu)
> gc.collect()
> torch.cuda.empty_cache()
> ```