torch.utils.bottleneck

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)bottleneck.rst, line 4)

Unknown directive type "automodule".

```
.. automodule:: torch.utils.bottleneck
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)bottleneck.rst, line 5)

Unknown directive type "currentmodule".

```
.. currentmodule:: torch.utils.bottleneck
```

torch.utils.bottleneck is a tool that can be used as an initial step for debugging bottlenecks in your program. It summarizes runs of your script with the Python profiler and PyTorch's autograd profiler.

Run it on the command line with

```
python -m torch.utils.bottleneck /path/to/source/script.py [args]
```

where [args] are any number of arguments to *script.py*, or run `python -m torch.utils.bottleneck -h` for more usage instructions.

Warning

Because your script will be profiled, please ensure that it exits in a finite amount of time.

Warning

Due to the asynchronous nature of CUDA kernels, when running against CUDA code, the cProfile output and CPU-mode autograd profilers may not show correct timings: the reported CPU time reports the amount of time used to launch the kernels but does not include the time the kernel spent executing on a GPU unless the operation does a synchronize. Ops that do synchronize appear to be extremely expensive under regular CPU-mode profilers. In these case where timings are incorrect, the CUDA-mode autograd profiler may be helpful.

Note

To decide which (CPU-only-mode or CUDA-mode) autograd profiler output to look at, you should first check if your script is CPU-bound ("CPU total time is much greater than CUDA total time"). If it is CPU-bound, looking at the results of the CPU-mode autograd profiler will help. If on the other hand your script spends most of its time executing on the GPU, then it makes sense to start looking for responsible CUDA operators in the output of the CUDA-mode autograd profiler.

Of course the reality is much more complicated and your script might not be in one of those two extremes depending on the part of the model you're evaluating. If the profiler outputs don't help, you could try looking at the result of :func:`torch.autograd.profiler.emit_nvtx()` with `nvprof`. However, please take into account that the NVTX overhead is very high and often gives a heavily skewed timeline.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)bottleneck.rst, line 45); *backlink*

Unknown interpreted text role "func".

Warning

If you are profiling CUDA code, the first profiler that `bottleneck` runs (cProfile) will include the CUDA startup time (CUDA buffer allocation cost) in its time reporting. This should not matter if your bottlenecks result in code much slower than the CUDA startup time.

For more complicated uses of the profilers (like in a multi-GPU case), please see https://docs.python.org/3/library/profile.html or

:func:`torch.autograd.profiler.profile()` for more information.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)bottleneck.rst, line 58); *backlink*

Unknown interpreted text role "func".