

Hauke Kirchner

Getting started with profiling PyTorch

Deep Learning with GPU Cores

Course Agenda

	Deep Learning with GPU cores
09.30 - 09.45	Welcome
09.45 - 10.15 (30 min)	Deep Learning and Infrastructure
10.15 - 11.30 (60 min)	Practical: Working on the GPU
11.30 - 11.45	Short break ☕
11.45 - 12.00 (15 min)	Introduction to Profiling
12.00 - 12.45 (45 min)	Practical: Profiling Jobs
12.45 - 13.00	General Q&A

Table of contents

- 1 Motivation
- 2 Methods
- 3 Tool overview
- 4 PyTorch Profiler
- 5 DeepSpeed - FlopsProfiler
- 6 Practical

Why is it important to profile the training process of neural networks?

Training speed

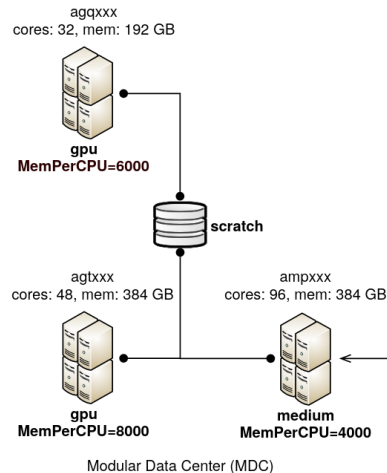


Energy efficiency



Training speed

- training of neural networks is computationally intensive
- workflow needs to be optimized for available hardware
 - ▶ How to make good use of clusters with heterogenous hardware?
 - ▶ How many GPUs are worth requesting?
- ⇒ optimization for available accelerators is critical in ML/DL

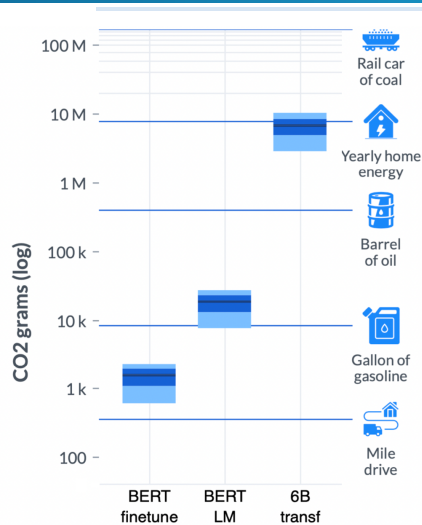


Structure and resources of a part of the Scientific Compute Cluster.

Image source: Adapted from <https://www.gwdg.de/web/guest/hpc-on-campus/scc>, accessed on: 09.11.2022

Energy efficiency ⚡

- idea of **GreenAI** (Schwartz et al. 2019)
- very different energy requirements (finetuning vs training)
- deep learning is emerging in several fields
 - ⇒ the impact on energy consumption and consequently, our climate is growing



CO₂ Relative Size Comparison

Image source: Adapted from Dodge et al. 2022

Outline

- 1 Motivation
- 2 Methods**
- 3 Tool overview
- 4 PyTorch Profiler
- 5 DeepSpeed - FlopsProfiler
- 6 Practical

Metrics

metric	purpose
Execution time FLOPS	traditionally
Throughput: $\frac{\text{images}}{\text{sec}}$	with the advent of GPUs
Time to Accuracy (TTA) Average Time to Multiple Thresholds (ATTMT)	deep learning

Image source: Adapted from https://snehilverma41.github.io/Metrics_ML_FastPath19.pdf

Outline

1 Motivation

2 Methods

3 Tool overview

4 PyTorch Profiler

5 DeepSpeed - FlopsProfiler

6 Practical

Tools

 PyTorch**DeepSpeed**

- PyTorch - Profiler ^a
- collection of performance metrics
- identification of expensive operators
- tracking of the kernel activity

^ahttps://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html

- FlopsProfiler ^a
- model speed (latency, throughput)
- efficiency (FLOPS^b)

^a<https://www.deepspeed.ai/tutorials/flops-profiler/>

^b floating-point operations per second

Outline

- 1 Motivation
- 2 Methods
- 3 Tool overview
- 4 PyTorch Profiler**
- 5 DeepSpeed - FlopsProfiler
- 6 Practical

PyTorch Profiler With TensorBoard

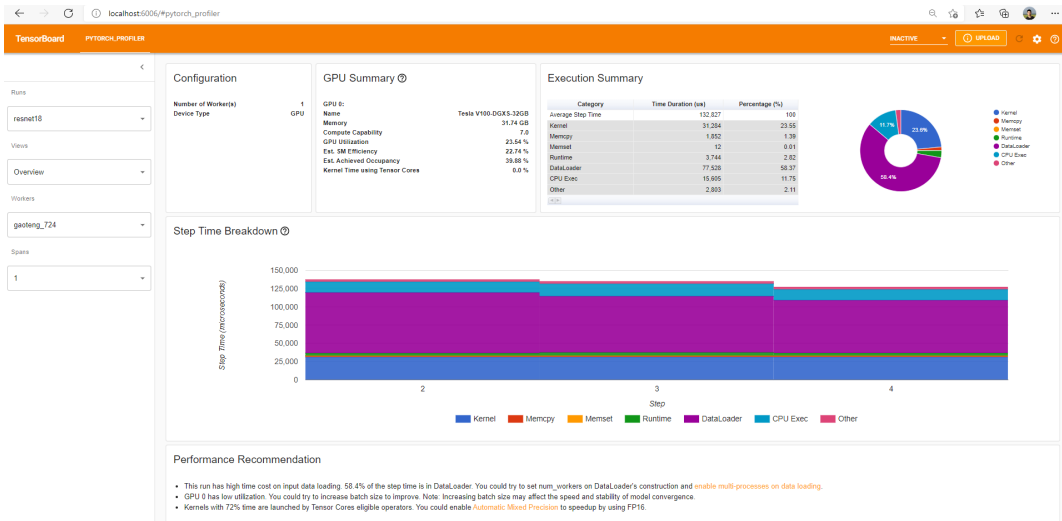


Image source: https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

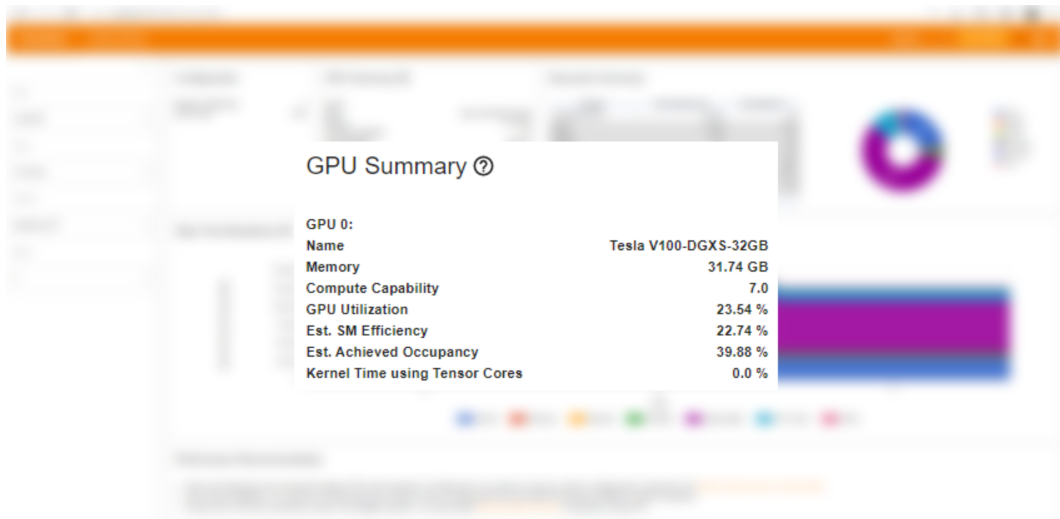


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	132,827	100
Kernel	31,284	23.55
Memcpy	1,852	1.39
Memset	12	0.01
Runtime	3,744	2.82
DataLoader	77,528	58.37
CPU Exec	15,605	11.75
Other	2,803	2.11

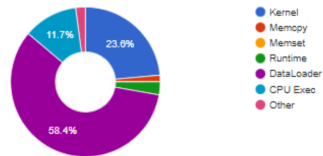


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

Step Time Breakdown ②



Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

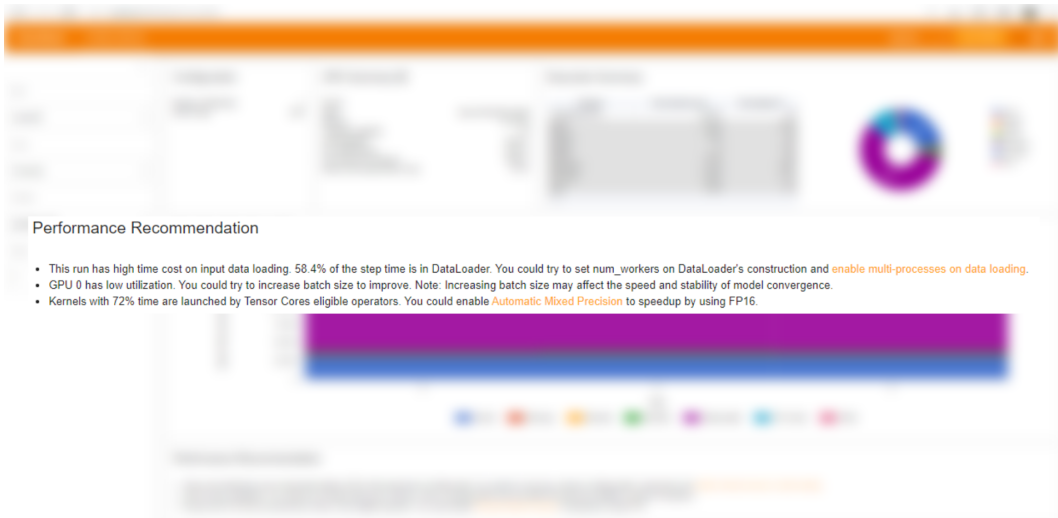


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

Outline

- 1 Motivation
- 2 Methods
- 3 Tool overview
- 4 PyTorch Profiler
- 5 DeepSpeed - FlopsProfiler**
- 6 Practical

Summary

```
1  ----- DeepSpeed Flops Profiler -----
2  Profile Summary at step 10:
3  Notations:
4  data parallel size (dp_size), model parallel size(mp_size),
5  number of parameters (params), number of multiply-accumulate operations(MACs),
6  number of floating-point operations (flops), floating-point operations per second (FLOPS),
7  fwd latency (forward propagation latency), bwd latency (backward propagation latency),
8  step (weights update latency), iter latency (sum of fwd, bwd and step latency)
9
10 world size: 1
11 data parallel size: 1
12 model parallel size: 1
13 batch size per GPU: 80
14 params per gpu: 336.23 M
15 params of model = params per GPU * mp_size: 336.23 M
16 fwd MACs per GPU: 3139.93 G
17 fwd flops per GPU: 6279.86 G
18 fwd flops of model = fwd flops per GPU * mp_size: 6279.86 G
19 fwd latency: 76.67 ms
20 bwd latency: 108.02 ms
```

Aggregated Profile per GPU

```
28 ----- Aggregated Profile per GPU -----
29 Top modules in terms of params, MACs or fwd latency at different model depths:
30 depth 0:
31     params      - {'BertForPreTrainingPreLN': '336.23 M'}
32     MACs        - {'BertForPreTrainingPreLN': '3139.93 GMACs'}
33     fwd latency - {'BertForPreTrainingPreLN': '76.39 ms'}
34 depth 1:
35     params      - {'BertModel': '335.15 M', 'BertPreTrainingHeads': '32.34 M'}
36     MACs        - {'BertModel': '3092.96 GMACs', 'BertPreTrainingHeads': '46.97 GMACs'}
37     fwd latency - {'BertModel': '34.29 ms', 'BertPreTrainingHeads': '3.23 ms'}
38 depth 2:
39     params      - {'BertEncoder': '302.31 M', 'BertLMPredictionHead': '32.34 M'}
40     MACs        - {'BertEncoder': '3092.88 GMACs', 'BertLMPredictionHead': '46.97 GMACs'}
41     fwd latency - {'BertEncoder': '33.45 ms', 'BertLMPredictionHead': '2.61 ms'}
42 depth 3:
43     params      - {'ModuleList': '302.31 M', 'Embedding': '31.79 M', 'Linear': '31.26 M'}
44     MACs        - {'ModuleList': '3092.88 GMACs', 'Linear': '36.23 GMACs'}
45     fwd latency - {'ModuleList': '33.11 ms', 'BertPredictionHeadTransform': '1.83 ms'}
46 depth 4:
```

Detailed Profile per GPU

```
59 ----- Detailed Profile per GPU -----
60 Each module profile is listed after its name in the following order:
61 params, percentage of total params, MACs, percentage of total MACs, fwd latency,
62   ↪ percentage of total fwd latency, fwd FLOPS
63 BertForPreTrainingPreLN(
64   336.23 M, 100.00% Params, 3139.93 GMACs, 100.00% MACs, 76.39 ms, 100.00% latency, 82.21
65   ↪ TFLOPS,
66   (bert): BertModel(
67     335.15 M, 99.68% Params, 3092.96 GMACs, 98.50% MACs, 34.29 ms, 44.89% latency, 180.4
68     ↪ TFLOPS,
69     (embeddings): BertEmbeddings(...)
70     (encoder): BertEncoder(
71       302.31 M, 89.91% Params, 3092.88 GMACs, 98.50% MACs, 33.45 ms, 43.79% latency,
72       ↪ 184.93 TFLOPS,
73       (FinalLayerNorm): FusedLayerNorm(...)
74       (layer): ModuleList(
75         302.31 M, 89.91% Params, 3092.88 GMACs, 98.50% MACs, 33.11 ms, 43.35% latency,
76         ↪ 186.8 TFLOPS,
```

Torch profiler

```
1  [...]
2  for epoch in range(num_training_epochs):
3      [...]
4      prof = torch.profiler.profile(
5          schedule=torch.profiler.schedule(wait=1, warmup=1, active=3, repeat=2),
6          on_trace_ready=torch.profiler.tensorboard_trace_handler(logdir + "/profiler"),
7          record_shapes=True,
8          profile_memory=True,
9          with_stack=True)
10     prof.start()
11     for i, batch in enumerate(train_loader):
12         [...] # Load data, classify data, calculate loss
13         loss.backward(); optimizer.step()
14         prof.step()
15     prof.stop()
```

Tutorial: https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 24.03.2023

DeepSpeed - FlopsProfiler

```
1 from deepspeed.profiling.flops_profiler import FlopsProfiler
2 [...]
3 flop_prof = FlopsProfiler(model)
4 profile_step = 5; print_profile= True
5 for epoch in range(num_training_epochs):
6     [...]
7     for i, batch in enumerate(train_loader):
8         if i == profile_step:
9             flop_prof.start_profile()
10            [...] # Load data, classify data, calculate loss
11            if i == profile_step: # end profiling and print output
12                flop_prof.stop_profile()
13                flops = flop_prof.get_total_flops()
14                macs = flop_prof.get_total_macs()
15                params = flop_prof.get_total_params()
16                if print_profile:
17                    flop_prof.print_model_profile(profile_step=profile_step)
18                flop_prof.end_profile()
19            loss.backward(); optimizer.step()
```

Tutorial: <https://www.deepspeed.ai/tutorials/flops-profiler/>, accessed on: 24.03.2023

Profiling is the first step of optimizing



Image generated with stable diffusion:

"Sherlock Holmes locates the best graphical processing unit inside the data center for his deep learning workflow"

- "Stable Diffusion v1 version of the model requires 150,000 A100 GPU Hours for a single training session"^a

⇒ Optimization of deep learning workflows is of growing importance for energy efficiency.

^a<https://syncedreview.com/2022/11/09/almost-7x-cheaper-colossal-ais-open-source-solution-accelerates-aigc-at-a-low-cost-diffusion-pretraining-and-hardware-fine-tuning-can-be/>, accessed on: 10.11.2022

Practical

Task 1: Run the training job with the profilers. (15 min)

- Add the code for using the profilers to `train.py`¹
- Set the number of epochs to 1 and interrupt the training after 5 batches of data (see `fast_run` option in `train_with_logger.py`)
- decrease time requested in SLURM to 20 minutes
 - ▶ in `code/submit_train.sh` → `#SBATCH -t 00:20:00`

Practical

Task 2: Analyze the profilers output (15 min)

- Download the output of the PyTorch profiler and visualize it with TensorBoard
- The output of DeepSpeed's FlopsProfiler is in the slurm output file
- An example output of the profilers can be found in our course directory
 - ▶ Instructions on how to use them can be found in the [README](#)

References

Dodge, Jesse et al. (2022). *Measuring the Carbon Intensity of AI in Cloud Instances*. DOI: 10.48550/ARXIV.2206.05229. URL: <https://arxiv.org/abs/2206.05229>.

Schwartz, Roy et al. (2019). "Green AI". In: *CoRR* abs/1907.10597. arXiv: 1907.10597. URL: <http://arxiv.org/abs/1907.10597>.

Course Agenda

	Deep Learning with GPU cores
09.30 - 09.45	Welcome
09.45 - 10.15 (30 min)	Deep Learning and Infrastructure
10.15 - 11.30 (60 min)	Practical: Working on the GPU
11.30 - 11.45	Short break ☕
11.45 - 12.00 (15 min)	Introduction to Profiling
12.00 - 12.45 (45 min)	Practical: Profiling Jobs
12.45 - 13.00	General Q&A