Hauke Kirchner

Benchmarking tree species classification with synthetic data and deep learning

State of the art and plan for Scalable Computing Systems and Applications in AI, Big Data and HPC

Table of contents

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

Training speed

Energy efficiency

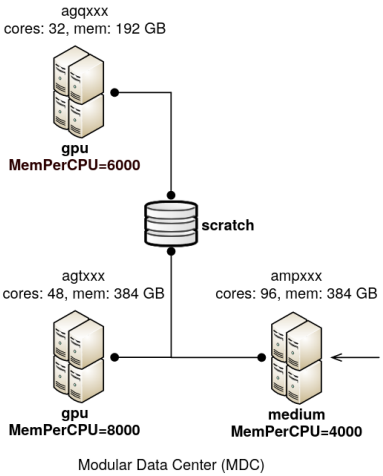
Training speed

- 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



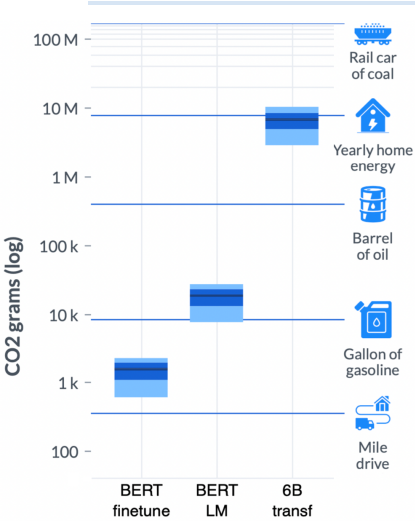
Structure and resources of a part of the Scientific Compute Cluster.

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

Energy efficiency ⚡

- for different neural networks, the required energy ranges from relatively low demands (finetuning) to very high demands (training of a complex network)
- depends on used hardware and energy source

⇒ As deep learning is emerging in several fields, the impact on energy consumption and consequently, our climate is growing



CO_2 Relative Size Comparison

Image source: Adapted from Dodge et al. 2022

Use case

Tree species classification from lidar

- monitoring of tree species is essential for forest restructuring towards more resilient forests in changing environments under climate change condition [a]
- lidar is already in use for forest inventories (White et al. 2016)
- Problem: lack of pre-trained models

Spruce and beech stand close to the national park "Hainich".

Image source: Adapted from Henkel, Hese, and Thiel 2021

[a] https://www.umweltbundesamt.de/angepasster-waldumbau

Outline

Workflow

Pre-training with synthetic data

1. scene generation with Arbaro[a]
2. synthetic lidar data is generated with the lidar operations simulator Helios++ (Esmorís et al. 2022)
3. pre-training of PointNet, which is a very popular architecture for point cloud analysis (Qi et al. 2016)

[a] https://github.com/wdiestel/arbaro

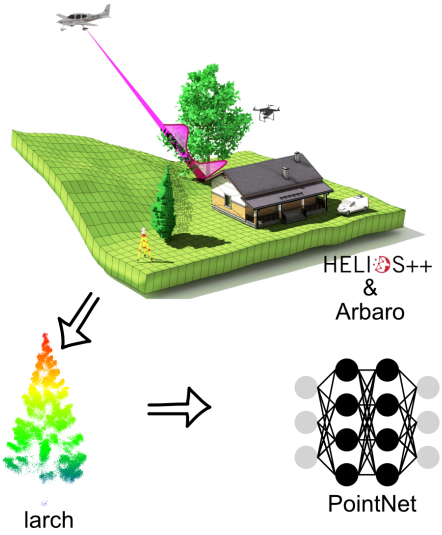


Image source: Adapted from Esmorís et al. 2022

Metrics

metric	purpose
Execution time FLOPS	traditionally
Throughput: $\frac{images}{sec}$	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

Tools - Overview

tool	metrics	scope
PyTorch Profiler With TensorBoard	performance metrics (e.g. time, memory)	PyTorch
fvcore	FLOPS	
Vtune	general	Intel-only
likwid	performance metrics	general

⇒ different tools for different use cases

⇒ Vtune and likwid can profile all kind of applications

⇒ here I will focus on profiling tools optimized for PyTorch

PyTorch Profiler With TensorBoard

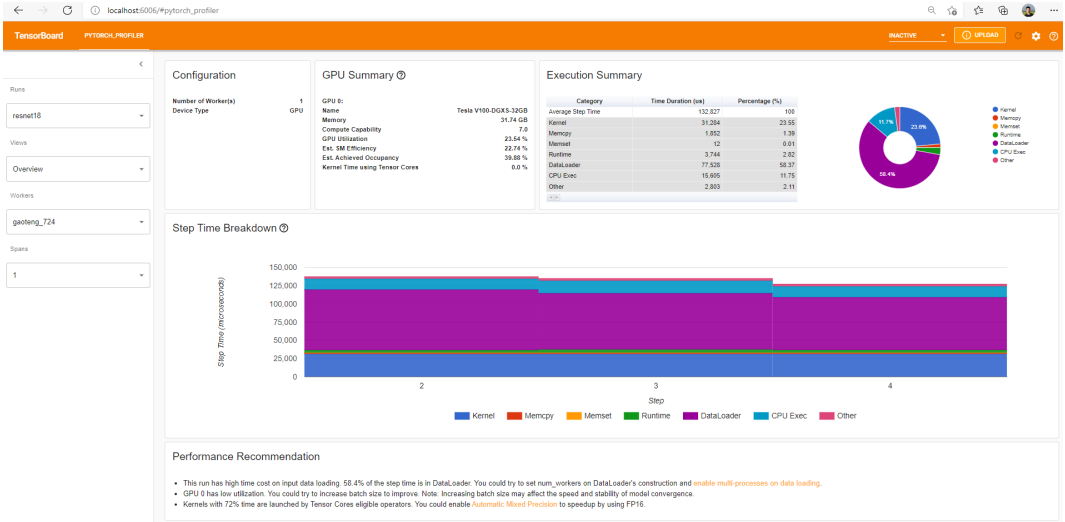


Image source: https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

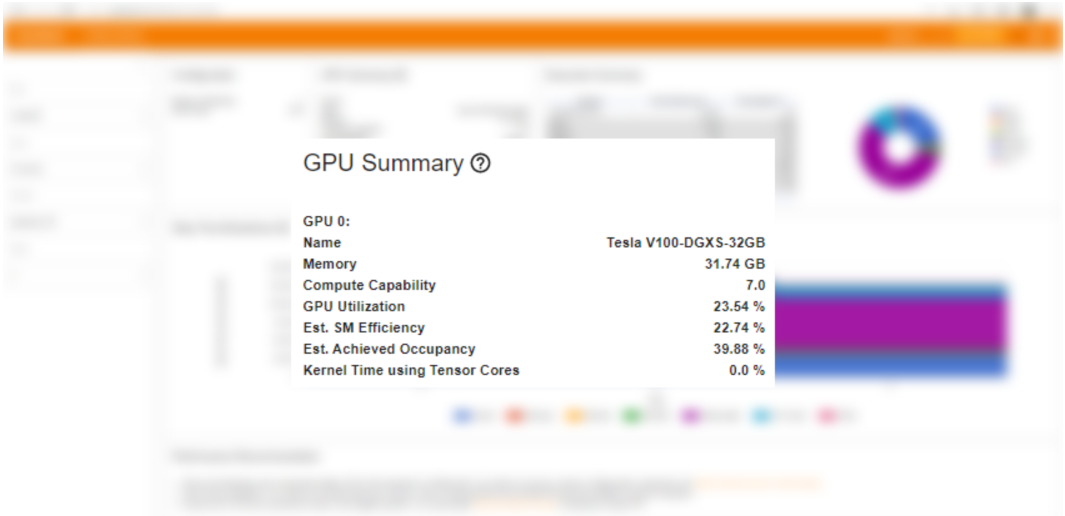


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

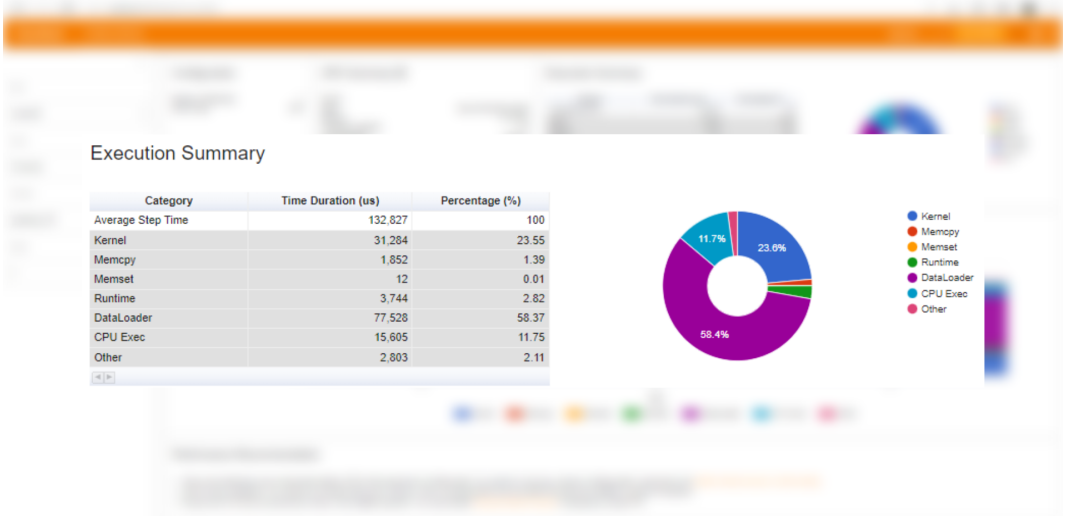


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

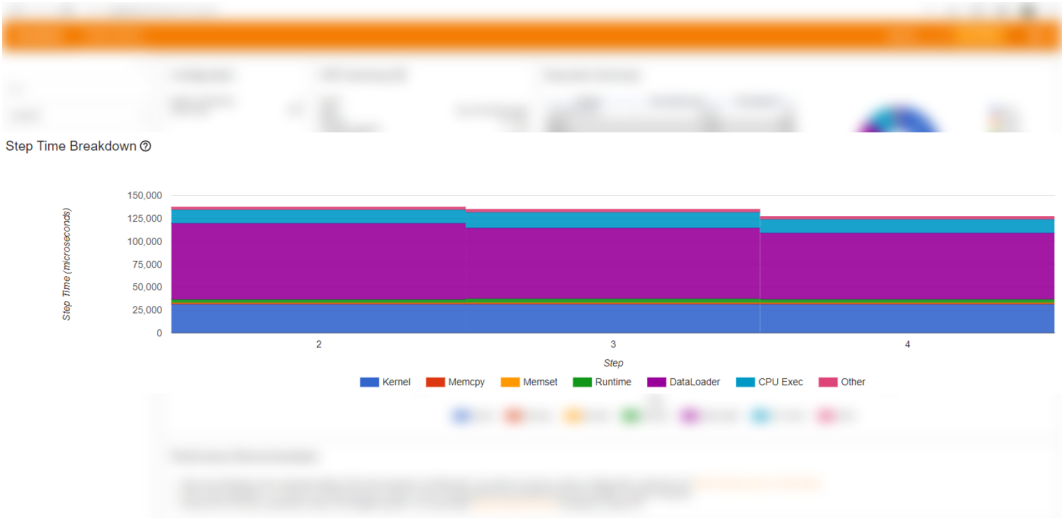


Image source: Adapted from https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html, accessed on: 22.11.2022

PyTorch Profiler With TensorBoard

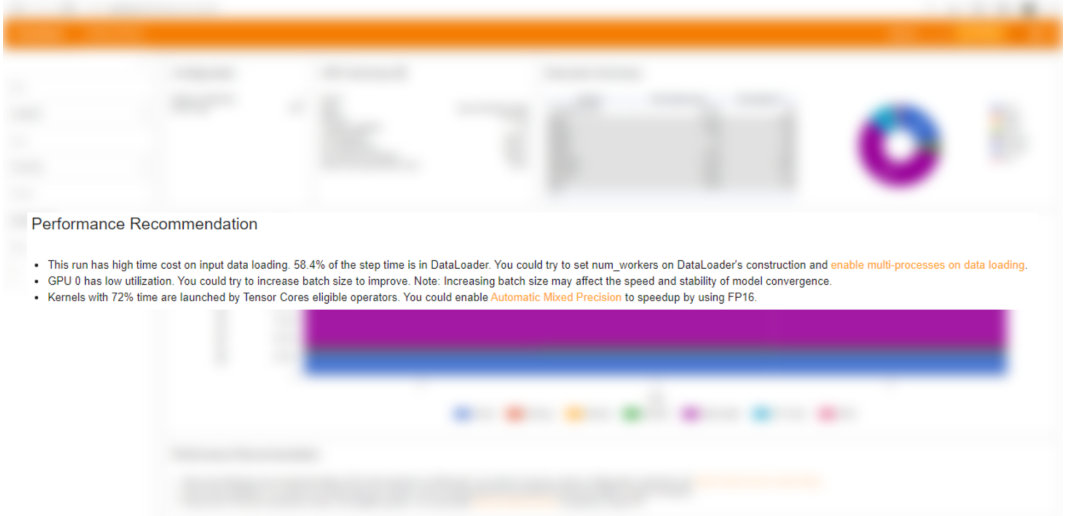


Image source: Adapted from `https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html`, accessed on: 22.11.2022

fvcore - Flop Counter for PyTorch Models

```
$ from fvcore.nn import flop_count_table
$ print(flop_count_table(flops))
| module          | #parameters or shape   | #flops   |
|:----------------|:-----------------------|:---------|
| model           | 81.194K                | 0.275M   |
|  conv1          |  60                    |  48.6K   |
|   conv1.weight  |   (6, 1, 3, 3)         |          |
|   conv1.bias    |   (6,)                 |          |
|  conv2          |  0.88K                 |  0.146M  |
|   conv2.weight  |   (16, 6, 3, 3)        |          |
|   conv2.bias    |   (16,)                |          |
|  fc1            |  69.24K                |  69.12K  |
|   fc1.weight    |   (120, 576)           |          |
|   fc1.bias      |   (120,)               |          |
|  fc2            |  10.164K               |  10.08K  |
|   fc2.weight    |   (84, 120)            |          |
|   fc2.bias      |   (84,)                |          |
|  fc3            |  0.85K                 |  0.84K   |
|   fc3.weight    |   (10, 84)             |          |
|   fc3.bias      |   (10,)                |          |
```

```
$ from fvcore.nn import flop_count_str
$ print(flop_count_str(flops))
Net(
  #params: 81.19K, #flops: 0.27M
  (conv1): Conv2d(
    1, 6, kernel_size=(3, 3), stride=(1, 1)
    #params: 60, #flops: 48.6K
  )
  (conv2): Conv2d(
    6, 16, kernel_size=(3, 3), stride=(1, 1)
    #params: 0.88K, #flops: 0.15M
  )
  (fc1): Linear(
    in_features=576, out_features=120, bias=True
    #params: 69.24K, #flops: 69.12K
  )
  (fc2): Linear(
    in_features=120, out_features=84, bias=True
    #params: 10.16K, #flops: 10.08K
  )
  (fc3): Linear(
    in_features=84, out_features=10, bias=True
    #params: 0.85K, #flops: 0.84K
  )
```

Image source: https://github.com/facebookresearch/fvcore/blob/main/docs/flop_count.md, accessed on: 22.11.2022

Research Goals

- Identify **profiling tools** that can help to **optimize an existing PyTorch workflow**.
- As this approach should help scientists, the **usibility is of high importance**. The simplest tool for doing the job is preferred.
- In contrast to training benchmark suites, such as MLPerf, I do not focus on benchmarking hardware but on optimizing an existing PyTorch workflow (viewpoint of a scientist)

Benchmarking 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

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.

Esmorís, Alberto M. et al. (2022). "Virtual LiDAR Simulation as a High Performance Computing Challenge: Toward HPC HELIOS++". In: *IEEE Access* 10, pp. 105052–105073. DOI: 10.1109/ACCESS.2022.3211072.

Henkel, Andreas, Sören Hese, and Christian Thiel (Jan. 2021). "Erhöhte Buchenmortalität im Nationalpark Hainich?" In.

Qi, Charles R. et al. (2016). *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. DOI: 10.48550/ARXIV.1612.00593. URL: https://arxiv.org/abs/1612.00593.

White, Joanne C. et al. (2016). "Remote Sensing Technologies for Enhancing Forest Inventories: A Review". In: *Canadian Journal of Remote Sensing* 42.5. https://doi.org/10.1080/07038992.2016.1207484, pp. 619–641. DOI: 10.1080/07038992.2016.1207484. eprint: https://doi.org/10.1080/07038992.2016.1207484.