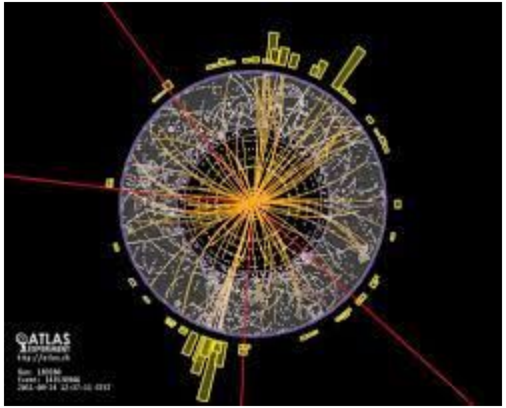
Estimation of energy cost and efficiency of HEP data compression ML algorithm (Baler)



Leonid Didukh
12.07.2023

Motivation

- The efficient computational resource utilisation is challenge in scientific and industrial research. It affects time, money and environment.
- Data preservation is difficult process as it requires the stable storage facilities, energy and finance.
- Massive energy and climate footprint of ML models. With increasing data and various deployed AI the energy consumption growth and CO_2 emission as well.
- The energy demand is growing exponentially for many ML architectures.
- The amount of AI-models is growing, the DNN becoming deeper, the amount of collected data is growing.
- There is the way to optimize the cost with respect to the performance metrics of ML model.

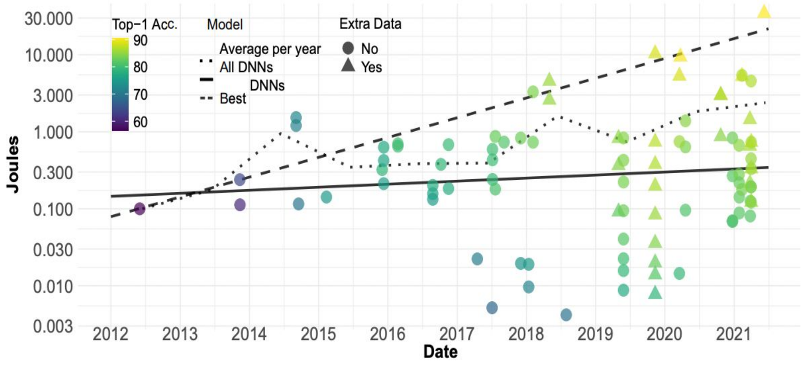


Figure 9: Estimated Joules of a forward pass (CV). The dashed line is a linear fit (logarithmic y-axis) for the models with highest accuracy per year. The solid line fits all models.

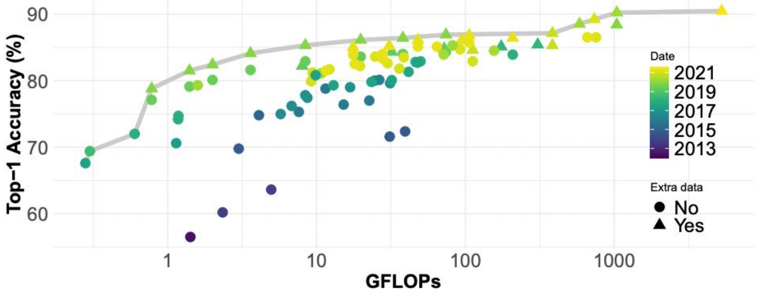


Figure 4: Relation between accuracy and GFLOPs.

Problems and Solutions

- Data preservation problem (https://arxiv.org/pdf/2302.03583.pdf, https://home.cern/news/news/computing/lhc-pushing-computing-limits)
    - The data moving is one of the most expensive operation. How to reduce the moving/copying operation?
    - We need to have a tool to keep and read/write data fastly and cheap -> Data compression is one of the solution.
- CPU challenge:
    - The number of FLOPS is growing with the accuracy of DNN. Computation (forward pass) becomes longer.
    - We need to have a way to do the fast computation preserving and improving the accuracy of model.
- Energy consumption estimation and reduction.
    - How to measure the energy consumption for DNN application?
    - What kind of metrics is the most representable?
    - How to optimise or develop green DNN?

Project goals

- Profile the baler on CPU:
  - Perform the parametrics test varying the number of epoch, input data size, batch size of baler (AE)
  - Estimate the runtime metrics, latencies, bandwidth
  - Build the dataset that contains the measured metrics
  - Find the hot spots, bottlenecks and the most expensive operations of Baler
- Profile the baler on GPU:
  - Use different profiler to estimate the GPU runtime and memory
  - Estimate the Temperature of GPU during training/inference
- Estimate the energy consumption and C02 emission of Baler using CPU and GPU:
  - RAPM is the widely used approach to measure the energy cost
  - Check the possible tools to reduce the resource consumption, for example energy-aware pruning

Metrics that used to describe the DNN performance

Energy:

- Energy per floating-point operations, number of weights of a model, kernel size, number of layers, number of arithmetic operations
- Power Usage Effectiveness

$$p_t = \frac{1.58t(p_c + p_r + gp_g)}{1000} \qquad CO_2e = 0.954p_t$$

- Energy (Joules) - total energy consumed by hardware and power consumption in Watt.
- Energy = Energy of Data + Energy of Layers
- Data moving energy (data flow) energy - hard to estimate
- Static and Dynamic Power
- Energy per CPU, GPU, DRAM and System

https://luiscruz.github.io/green-ai/publications/2019-07-garciamartin-estimation.html
https://arxiv.org/pdf/1906.02243.pdf

Computing resources:

- FLOP, FLOPS per second, a-FLOPS
- MAD (Multiply - Addition)
- CPU/GPU utilization as percentage
- CPU/GPU time in hours
- Inference time, wall - clock time
- Pipeline bubble - time how long the divide is idle
- MAC (Memory access time)

https://arxiv.org/pdf/2002.05651.pdf
https://luiscruz.github.io/green-ai/publications/2019-07-garciamartin-estimation.html
https://arxiv.org/pdf/1906.02243.pdf

Tool that used to describe the ML model energy cost

Paper	Metric	Hardware	Estimation details and outcomes	Framework type
https://arxiv.org/pdf/2304.00897.pdf	MAC, Joules	CPU	Layer wise, fixed data set Models:ResNet	experiment-impact-tracker codecarbon, psutil
https://arxiv.org/pdf/2002.05651.pdf	FLOPS, Watt-hours, Joules	CPU, GPU	Proposed framework for the energy cost estimation	codecarbon, experiment-impact-tracker
https://arxiv.org/pdf/2109.05472.pdf	FLOPS per Watt, Efficiency (Flops/Joule)	CPU,GPU	System	ptflops
https://arxiv.org/pdf/1910.09700.pdf	CO2eq emitted	CPU/GPU	System consumption for different regions. Tips for the CO2 emission reduction	Machine Learning Emissions Calculator, mlco2
https://arxiv.org/pdf/1906.02243.pdf	Power Usage Effectiveness, CO2	CPU,GPU 8 NVIDIA P100 GPUs	Models: Tranformer, ELMO,BERT, GPT-2	nvidia-smi RAPL power meter
https://arxiv.org/pdf/1710.05420.pdf	energy-precision ratio		Tranformer models	NeuralPower

Metrics that used to describe the DNN performance

Paper	Metric	Hardware	Estimation type	Framework type
https://arxiv.org/pdf/2304.00897.pdf	Joules	CPU	Layer wise, fixed data set Models: ResNet	experiment-impact-tracker codecarbon
https://arxiv.org/pdf/2002.05651.pdf	floating point operations	CPU, GPU		experiment-impact-tracker
https://arxiv.org/pdf/2109.05472.pdf	FLOPS, “peak FLOPS”	GPU	For whole system. Model: CV, NLP	nvidia tools
https://arxiv.org/pdf/2110.12894.pdf	FLOPS, MAC, CPU/GPU Type	CPU, GPU, or TPU	Transformers, Universal Transformers and Switch Transformers	efficiency misnomer

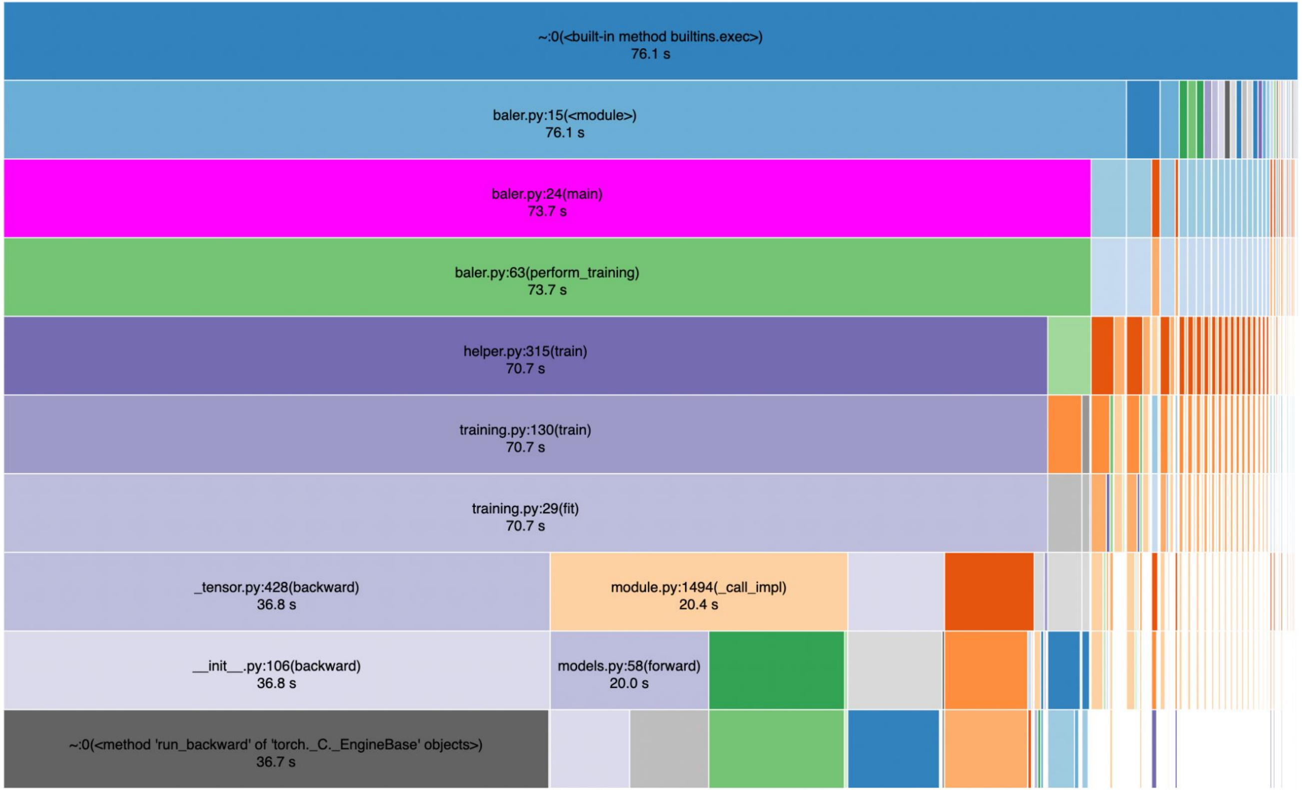
Setup:

DRAM: 16GB
CPU: M1
CPU count: 8
Platform system: macOS-10.16-x86_64-i386-64bit
Python version: 3.8.5

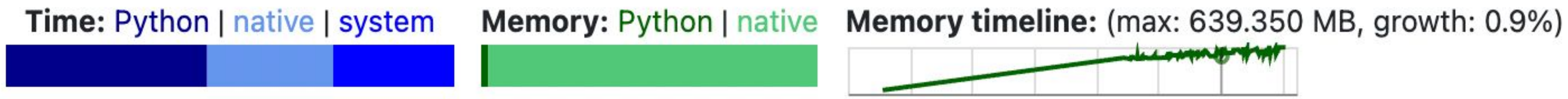
Model: AE
Number of parameters of Model: 61.54 k
Data: HEP data
Epoch: 5/100/500
Compression ratio: 2
Data dimensionality: 1
Batch Size:512/1024

```python
class AE(nn.Module):
    # This class is a modified version of the original class by George Dialektakis found at
    # https://github.com/Autoencoders-compression-anomaly/Deep-Autoencoders-Data-Compression-GSoC-2021
    # Released under the Apache License 2.0 found at https://www.apache.org/licenses/LICENSE-2.0.txt
    # Copyright 2021 George Dialektakis

    def __init__(self, n_features, z_dim, *args, **kwargs):
        super(AE, self).__init__(*args, **kwargs)

        self.activations = {}

        # encoder
        self.en1 = nn.Linear(n_features, 200, dtype=torch.float64)
        self.en2 = nn.Linear(200, 100, dtype=torch.float64)
        self.en3 = nn.Linear(100, 50, dtype=torch.float64)
        self.en4 = nn.Linear(50, z_dim, dtype=torch.float64)
        # decoder
        self.de1 = nn.Linear(z_dim, 50, dtype=torch.float64)
        self.de2 = nn.Linear(50, 100, dtype=torch.float64)
        self.de3 = nn.Linear(100, 200, dtype=torch.float64)
        self.de4 = nn.Linear(200, n_features, dtype=torch.float64)

        self.n_features = n_features
        self.z_dim = z_dim

    def encode(self, x):
        h1 = F.leaky_relu(self.en1(x))
        h2 = F.leaky_relu(self.en2(h1))
        h3 = F.leaky_relu(self.en3(h2))
        return self.en4(h3)

    def decode(self, z):
        h4 = F.leaky_relu(self.de1(z))
        h5 = F.leaky_relu(self.de2(h4))
        h6 = F.leaky_relu(self.de3(h5))
        out = self.de4(h6)
        return out

    def forward(self, x):
        z = self.encode(x)
        return self.decode(z)
```

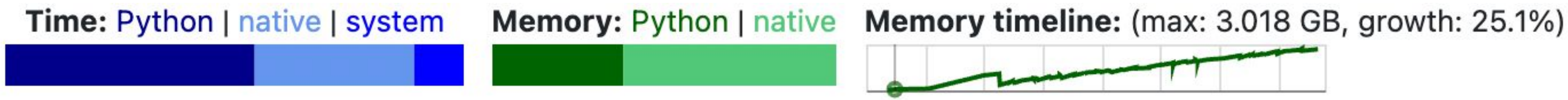
cProfile analysis (training)



Profiling using Scalene. Compression and Decompression



Time: 44,28,27 %
Memory:504MB, 22,9GB
Compression took: 0.307 minutes



Time:54%,34%,10%
Memory:4.074GB, 6.625GB
Decompression took: 0.826 minutes

Energy Estimation using code-carbon

For 5 epoch training operation:

Energy consumed for RAM : 0.000042 kWh. RAM Power : 6.0 W
Energy consumed for all CPUs : 0.000035 kWh. Total CPU Power : 5.0 W
0.000078 kWh of electricity used since the beginning.

Compression:

Energy consumed for RAM : 0.000026 kWh. RAM Power : 6.0 W
Energy consumed for all CPUs : 0.000021 kWh. Total CPU Power : 5.0 W
0.000047 kWh f electricity used since the beginning.

Decompression:

Energy consumed for RAM : 0.000075 kWh. RAM Power : 6.0 W
Energy consumed for all CPUs : 0.000063 kWh. Total CPU Power : 5.0 W
0.000138 kWh of electricity used since the beginning.

Timeline

- **19.06 - 30.06**
    - Hands-on tests and code analysis of Baler and writing of document on parallelization improvements. Test the framework locally using laptop/desktop. Conduct the research about possible ways how to improve the efficiency of the training/inference of Baler.
- **03.07 - 17.07**
    - Work on the deliverable number 2a. Define and test profiling metrics for CPU (list metrics, tests different ones, choose a final metric), compiling results into a presentation. The task was performed in collaboration with Google Summer of Code HSF student Manas Pratim Biswas.
- **17.07 - 31.07**
    - Work on the deliverable number 2b. Define and test if of the profiling metrics for GPU. Perform GPU tests and compile results into a presentation.
- **31.07 - 14.08**
    - Test code and scripts (from R. Cardoso) related to the green software analysis for estimation of GPU energy consumption,   link to GPU profiling done before.
- **14.08 - 28.08**
    - Compare results from R. Cardoso's scripts and references to green software metrics. Compile findings into a presentation.
- **14.08 - 28.08**
    - Apply ML-specific energy consumption metrics to Baler and compile results into a presentation.
- **31.08 - 15.09**
    - Wrap up results into a report

Current status

- **Reviewed the possible profilers for the CPU and GPU. Make a reading list and documents with the paper review**

- **Profiled the baler using CPU, spotted some weak points that related to the redundant copying operation**

- **Estimated the energy consumption and CO2 emission for training/inference using CPU/GPU**

- **Working repository: https://github.com/software-energy-cost-studies/profiling**

Thank you for the attention!