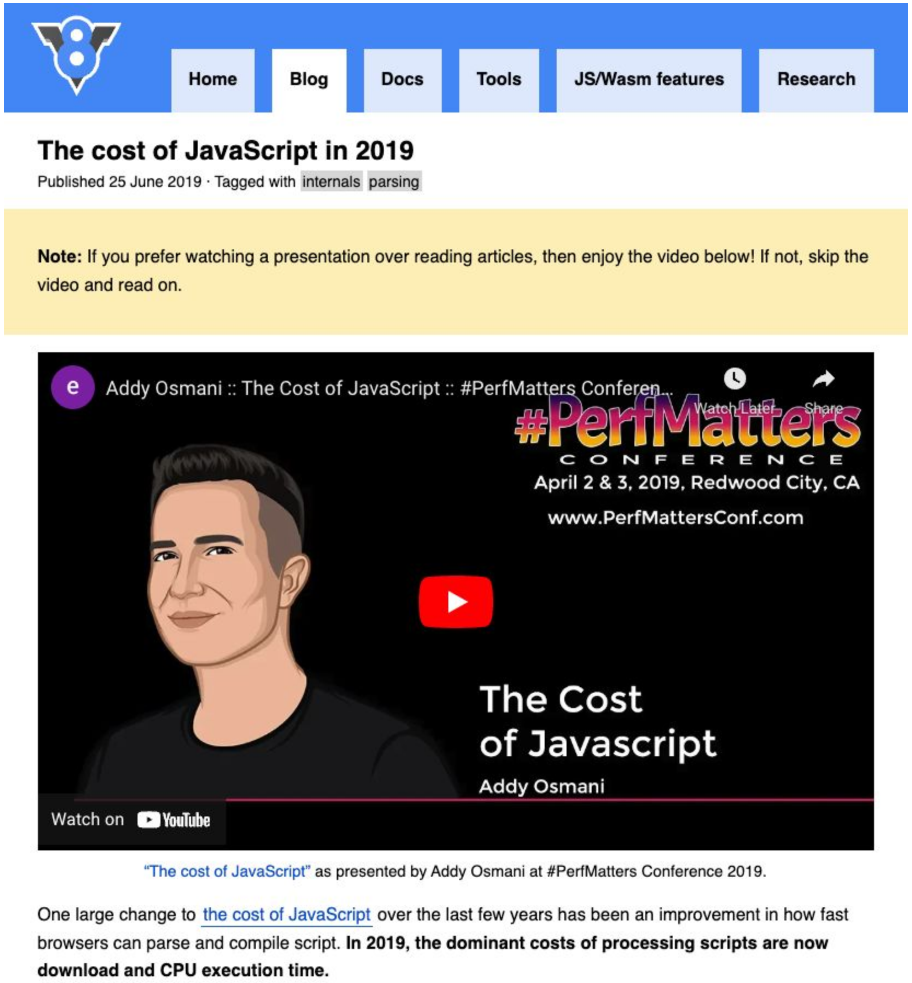
Project tracks

Ivano Malavolta

VRIJE UNIVERSITEIT AMSTERDAM

LOOKING FURTHER

WARMLY Suggested readings



The cost of JavaScript in 2019

Published 25 June 2019 · Tagged with internals parsing

Note: If you prefer watching a presentation over reading articles, then enjoy the video below! If not, skip the video and read on.

"The cost of JavaScript" as presented by Addy Osmani at #PerfMatters Conference 2019.

One large change to the cost of JavaScript over the last few years has been an improvement in how fast browsers can parse and compile script. **In 2019, the dominant costs of processing scripts are now download and CPU execution time.**

https://v8.dev/blog/cost-of-javascript-2019

Other interesting presentations: https://www.youtube.com/user/estellevw/videos

Tranco list

https://tranco-list.eu

Technical priority tasks (for all)

1. Check if you are able to apply the treatments to your subjects
   - Are you able to programmatically modify the index.html of a web app?
   - Are you able to run the ML algorithms in a larger pipeline?

3. Check if you can apply the treatments to your co-factors
   - Are you able to throttle the network?
   - Are you able to execute the subjects in both Chrome and Firefox?

4. Check if you can execute the experiment
   - Are you able to programmatically switch between the WebGL and WebGPU versions of a web app?
   - Are you able to execute a prefixed usage scenario in your apps?

5. Check if you can collect all the metrics of your experiment
   - Are you able to properly collect the page load time of a web page?
   - Do the energy measures you collected make sense?

     Do the math!

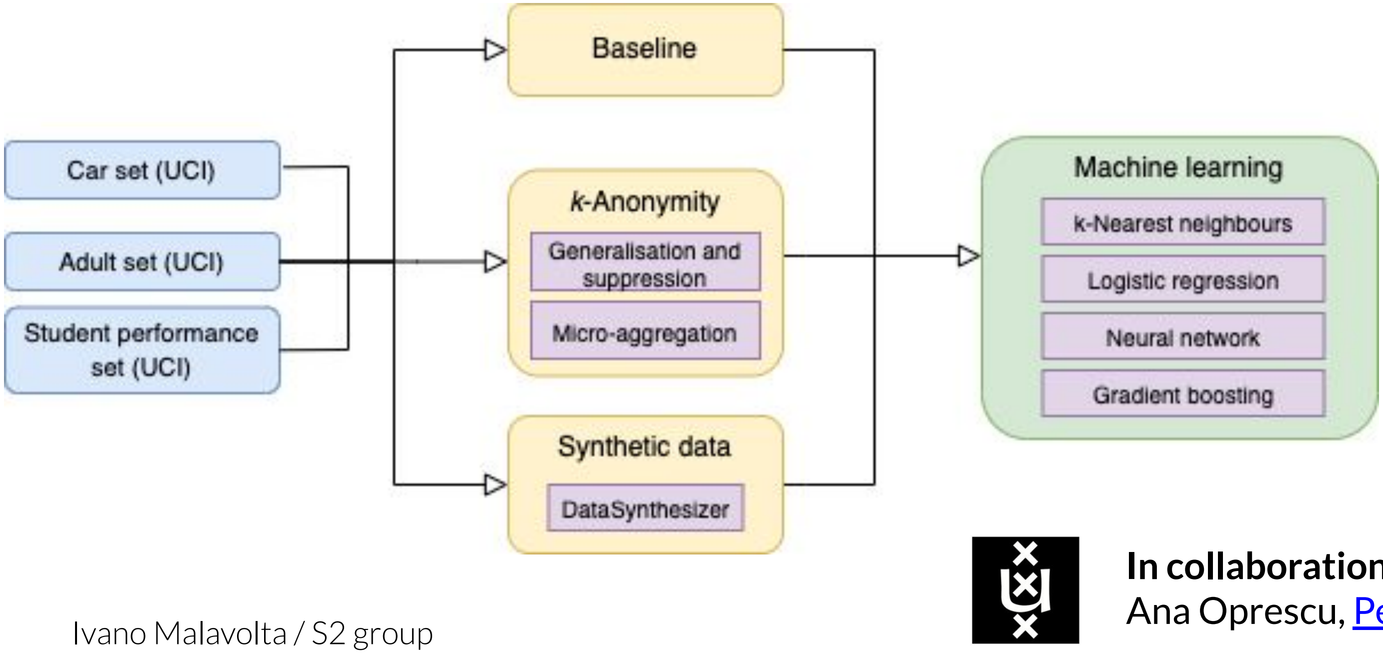
Team assignments

1. ML - anonymized
2. ML - synthesized
3. Nature - sorting algorithms
4. Accuracy of Code Carbon
5. High-performance Python libraries
6. Python loop optimization techniques
7. Python multithreading
8. LLM vs HTTP requests
9. Evolution of LLMs energy consumption
10. Memoization
11. Energy Efficient Job Scheduling
12. Frequency to Power Consumption ratio
13. Quantized LLMs
14. Ollama vs HF Transformers energy efficiency
15. Llama3.1 generated code energy efficiency
16. Energy Efficiency Comparison of LLMs
17. Greener ML software
18. Green molecular dynamics simulations
19. bioScience
20. ML Algos

1 - ML training on k-anonymised datasets

What is the impact of k-anonymity on the energy efficiency (and accuracy) of ML algorithms?

By looking at machine learning performance (energy & accuracy) on original data, k-anonymised and synthetic data, we study the impact of Privacy-Enhancing Techniques.



Car set (UCI)
Adult set (UCI)
Student performance set (UCI)

Baseline

k-Anonymity
- Generalisation and suppression
- Micro-aggregation

Synthetic data
- DataSynthesizer

Machine learning
- k-Nearest neighbours
- Logistic regression
- Neural network
- Gradient boosting

In collaboration with
Ana Oprescu, [Pepijn de Reus]

1 - ML training on k-anonymised datasets

- Related work shows that with K-anonymization
    - Microaggregation has better energy efficiency
    - Generalisation and suppression better ML accuracy, as long as hierarchy is well constructed
    - Can greatly reduce energy consumption (Zemanek & Hu, 2024)

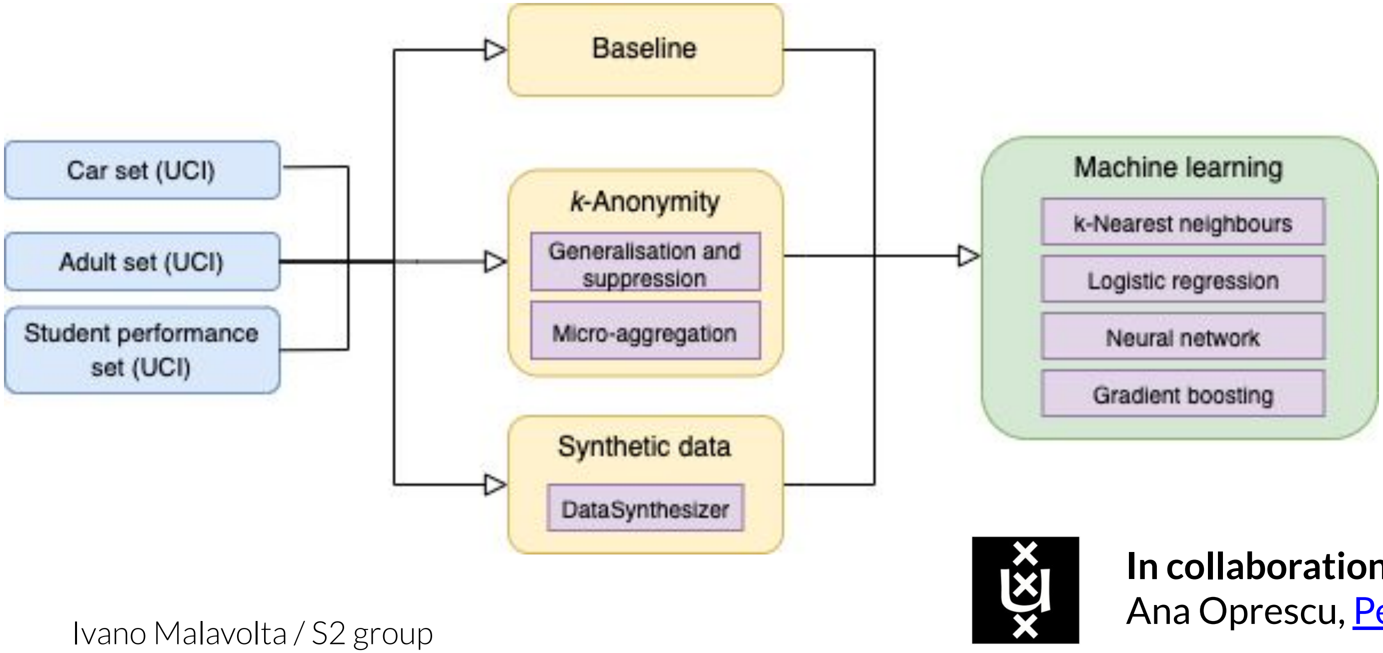
You could extend this by looking into:

- Does this hold for regression as well?
- Is there an optimal k-value based on the # features?
- How does k-anonymisation impact precision/F1-score?
- Does k-anonymisation improve fairness?
- Can we extend this framework to > 10 datasets?

2 - ML training on synthesized datasets

What is the impact of data synthesis techniques on the energy efficiency (**and accuracy**) of ML algorithms?

By looking at machine learning performance (energy & accuracy) on original data, k-anonymised and synthetic data, we study the impact of Privacy-Enhancing Techniques.



Car set (UCI)

Adult set (UCI)

Student performance set (UCI)

Baseline

k-Anonymity
- Generalisation and suppression
- Micro-aggregation

Synthetic data
- DataSynthesizer

Machine learning
- k-Nearest neighbours
- Logistic regression
- Neural network
- Gradient boosting

In collaboration with
Ana Oprescu, Pepijn de Reus

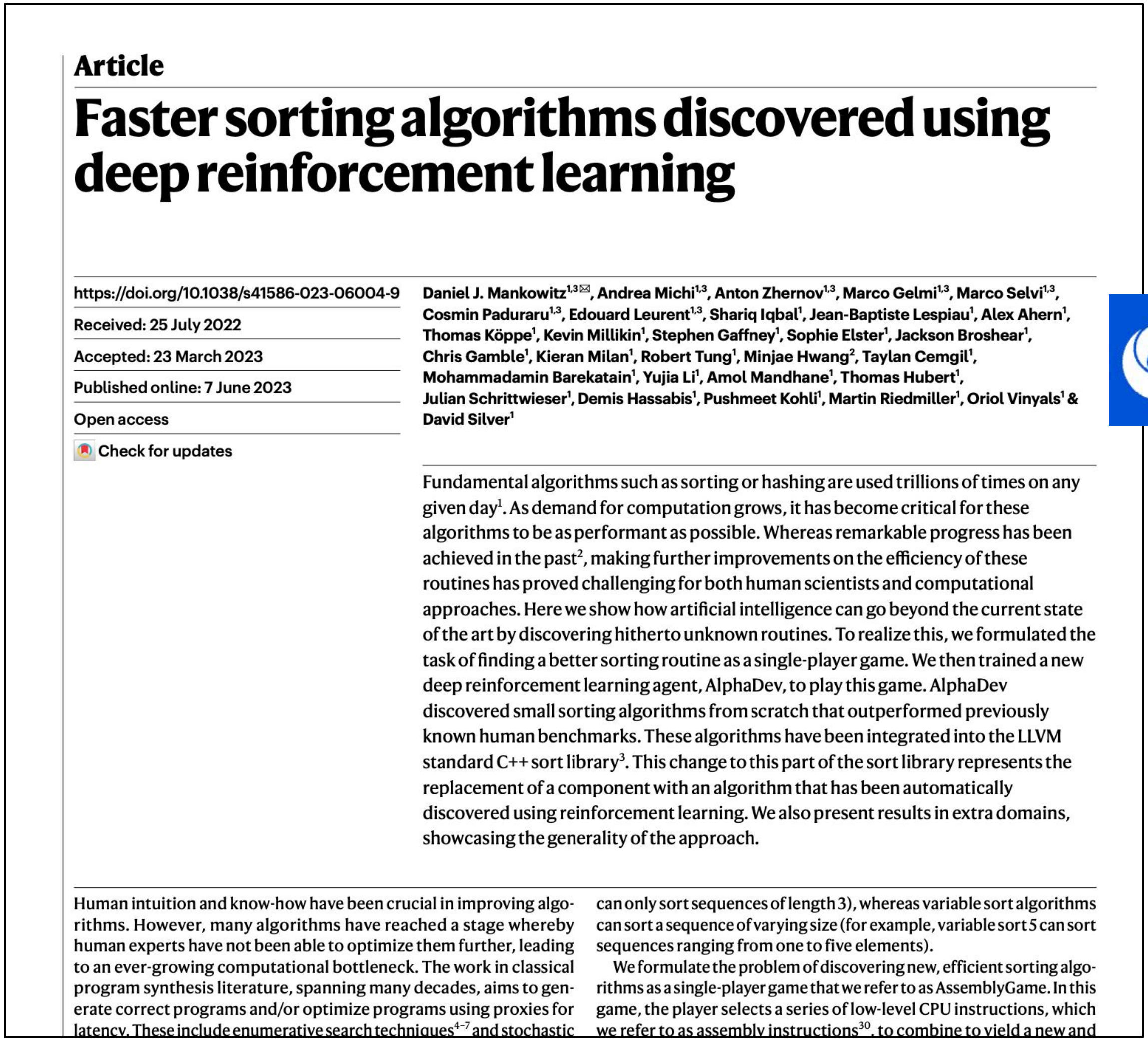
2 - ML training on synthesized datasets

- For DataSynthesizer, related work shows:
    - Lower to similar accuracy
    - higher energy consumption

You could extend this by looking into:

- Does this hold for regression as well?
- Can we extend this framework to > 10 datasets?
- What if we compare other synthetic data methods?
    - Synthetic Data Vault
    - YData
    - CTGAN etc. etc.

3 – Nature - sorting algos



Article

Faster sorting algorithms discovered using deep reinforcement learning

https://doi.org/10.1038/s41586-023-06004-9

Received: 25 July 2022

Accepted: 23 March 2023

Published online: 7 June 2023

Open access

Check for updates

Daniel J. Mankowitz[1,3,✉], Andrea Michi[1,3], Anton Zhernov[1,3], Marco Gelmi[1,3], Marco Selvi[1,3], Cosmin Paduraru[1,3], Edouard Leurent[1,3], Shariq Iqbal[1], Jean-Baptiste Lespiau[1], Alex Ahern[1], Thomas Köppe[1], Kevin Millikin[1], Stephen Gaffney[1], Sophie Elster[1], Jackson Broshear[1], Chris Gamble[1], Kieran Milan[1], Robert Tung[1], Minjae Hwang[2], Taylan Cemgil[1], Mohammadamin Barekatain[1], Yujia Li[1], Amol Mandhane[1], Thomas Hubert[1], Julian Schrittwieser[1], Demis Hassabis[1], Pushmeet Kohli[1], Martin Riedmiller[1], Oriol Vinyals[1] & David Silver[1]

Fundamental algorithms such as sorting or hashing are used trillions of times on any given day[1]. As demand for computation grows, it has become critical for these algorithms to be as performant as possible. Whereas remarkable progress has been achieved in the past[2], making further improvements on the efficiency of these routines has proved challenging for both human scientists and computational approaches. Here we show how artificial intelligence can go beyond the current state of the art by discovering hitherto unknown routines. To realize this, we formulated the task of finding a better sorting routine as a single-player game. We then trained a new deep reinforcement learning agent, AlphaDev, to play this game. AlphaDev discovered small sorting algorithms from scratch that outperformed previously known human benchmarks. These algorithms have been integrated into the LLVM standard C++ sort library[3]. This change to this part of the sort library represents the replacement of a component with an algorithm that has been automatically discovered using reinforcement learning. We also present results in extra domains, showcasing the generality of the approach.

Human intuition and know-how have been crucial in improving algorithms. However, many algorithms have reached a stage whereby human experts have not been able to optimize them further, leading to an ever-growing computational bottleneck. The work in classical program synthesis literature, spanning many decades, aims to generate correct programs and/or optimize programs using proxies for latency. These include enumerative search techniques[4–7] and stochastic

can only sort sequences of length 3), whereas variable sort algorithms can sort a sequence of varying size (for example, variable sort 5 can sort sequences ranging from one to five elements).

We formulate the problem of discovering new, efficient sorting algorithms as a single-player game that we refer to as AssemblyGame. In this game, the player selects a series of low-level CPU instructions, which we refer to as assembly instructions[30], to combine to yield a new and

https://www.nature.com/articles/s41586-023-06004-9

3 – Nature - sorting algos

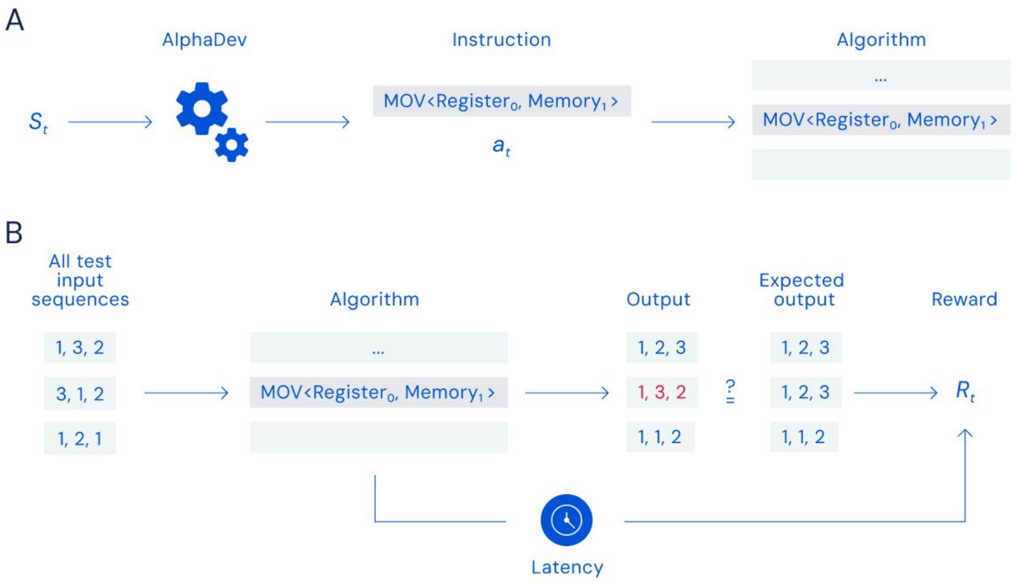


Figure A: The assembly game. The player, AlphaDev, receives the state of the system st as input and plays a move at by selecting an assembly instruction to add to the algorithm that has been generated thus far.

Figure B: The reward computation. After each move, the generated algorithm is fed test input sequences – for sort3, this corresponds to all combinations of sequences of three elements. The algorithm then generates an output, which is compared to the expected output of sorted sequences for the case of sorting. The agent is rewarded based on the algorithm's correctness and latency.

https://www.deepmind.com/blog/alphadev-discovers-faster-sorting-algorithms

3 – Nature - sorting algos

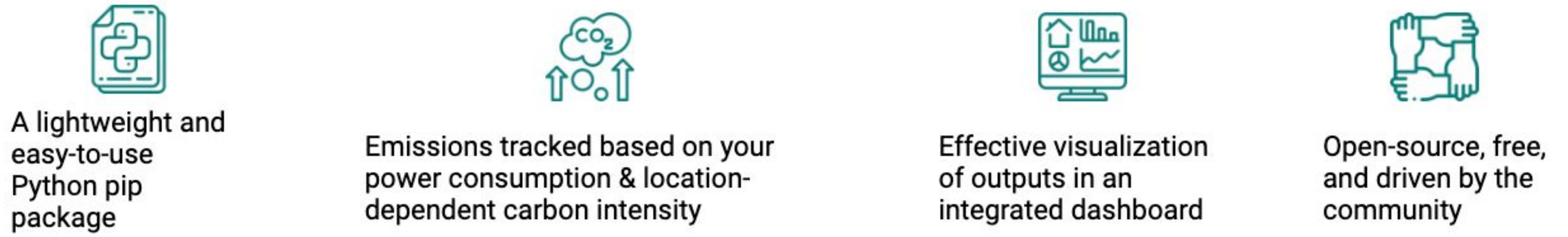
Informal RQ

Are the AI-generated sorting algorithms also more energy-efficient than traditional ones?

- Subjects:
  - The AI-generated sorting algorithms are available here
- Suggestions for main factors:
  - Size of the input sequence
  - Characteristics of the input sequence, e.g.: random, partially sorted, etc.
  - Hardware platform (as blocking factor): laptop, Raspberry Pi, server
- Subjects: a large set of sorting algorithms (at least 10, more is better)
- Related work:
  - Related 1
  - Related 2
  - Related 3

4 – Accuracy of Code Carbon

What is Code Carbon?

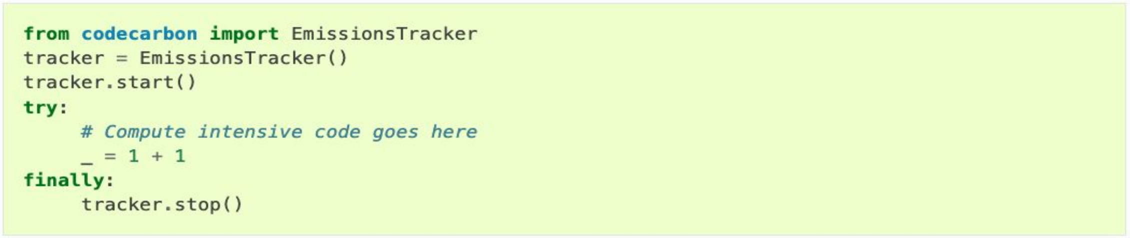


A lightweight and easy-to-use Python pip package

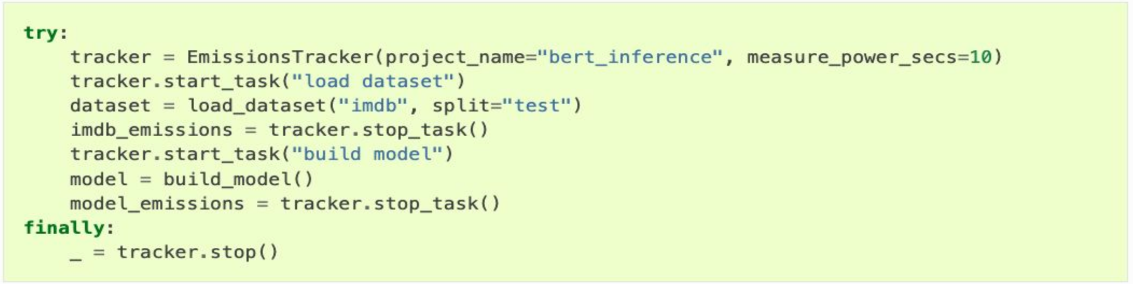
Emissions tracked based on your power consumption & location-dependent carbon intensity

Effective visualization of outputs in an integrated dashboard

Open-source, free, and driven by the community



```
from codecarbon import EmissionsTracker
tracker = EmissionsTracker()
tracker.start()
try:
     # Compute intensive code goes here
     _ = 1 + 1
finally:
     tracker.stop()
```



```
try:
    tracker = EmissionsTracker(project_name="bert_inference", measure_power_secs=10)
    tracker.start_task("load dataset")
    dataset = load_dataset("imdb", split="test")
    imdb_emissions = tracker.stop_task()
    tracker.start_task("build model")
    model = build_model()
    model_emissions = tracker.stop_task()
finally:
    _ = tracker.stop()
```

Problem: Behind the lines CC uses a bunch of low-level power profilers and each of those has its own level of accuracy, potentially impacting the final results of CC (see here for the details)

Can we trust them?

https://codecarbon.io

4 – Accuracy of Code Carbon

Informal RQ

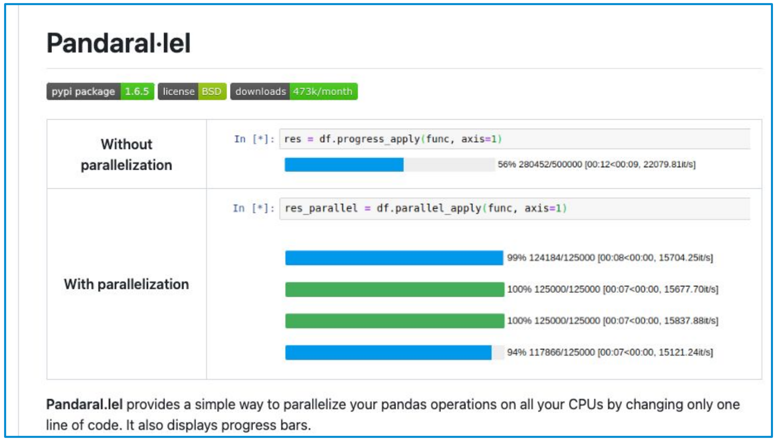
What is the accuracy of Code Carbon in terms of power consumption?

- Gist of the project: you run a set of Python functions for several minutes, while measuring their power consumption via CC and a hardware power meter **at the same time**, and then you measure how CC values differs from those of the HW meter
- Suggestions for main factors:
    - Different configurations of the tracker module of CC
        - For example: measure_power_secs, log_level, etc.
    - Hardware platform (as blocking factor): laptop, Raspberry Pi, jetson Nano
- Workload: reuse already-existing Python performance benchmarks (multiple ones, to mitigate bias), at best with different levels of usage resources
    - Include a baseline where the Python script is not doing anything
- Starting point in the literature

5 – High-performance Python libraries

Context: today there are several Python libraries with the goal of making Python code more performant, with very promising results

Two examples of such libraries:



Pandaral·lel

Without parallelization

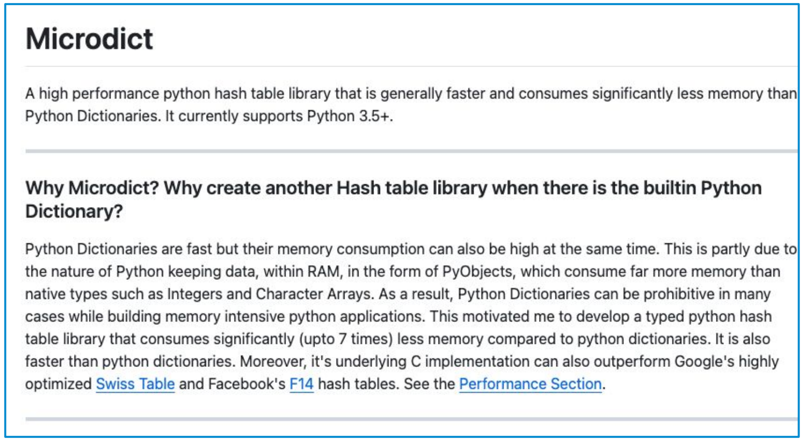
```
In [*]: res = df.progress_apply(func, axis=1)
56% 280452/500000 [00:12<00:09, 22079.81it/s]
```

With parallelization

```
In [*]: res_parallel = df.parallel_apply(func, axis=1)
99% 124184/125000 [00:08<00:00, 15704.25it/s]
100% 125000/125000 [00:07<00:00, 15677.70it/s]
100% 125000/125000 [00:07<00:00, 15837.88it/s]
94% 117866/125000 [00:07<00:00, 15121.24it/s]
```

Pandaral.lel provides a simple way to parallelize your pandas operations on all your CPUs by changing only one line of code. It also displays progress bars.

https://github.com/nalepae/pandarallel



Microdict

A high performance python hash table library that is generally faster and consumes significantly less memory than Python Dictionaries. It currently supports Python 3.5+.

Why Microdict? Why create another Hash table library when there is the builtin Python Dictionary?

Python Dictionaries are fast but their memory consumption can also be high at the same time. This is partly due to the nature of Python keeping data, within RAM, in the form of PyObjects, which consume far more memory than native types such as Integers and Character Arrays. As a result, Python Dictionaries can be prohibitive in many cases while building memory intensive python applications. This motivated me to develop a typed python hash table library that consumes significantly (upto 7 times) less memory compared to python dictionaries. It is also faster than python dictionaries. Moreover, it's underlying C implementation can also outperform Google's highly optimized Swiss Table and Facebook's F14 hash tables. See the Performance Section.

https://github.com/touqir14/Microdict

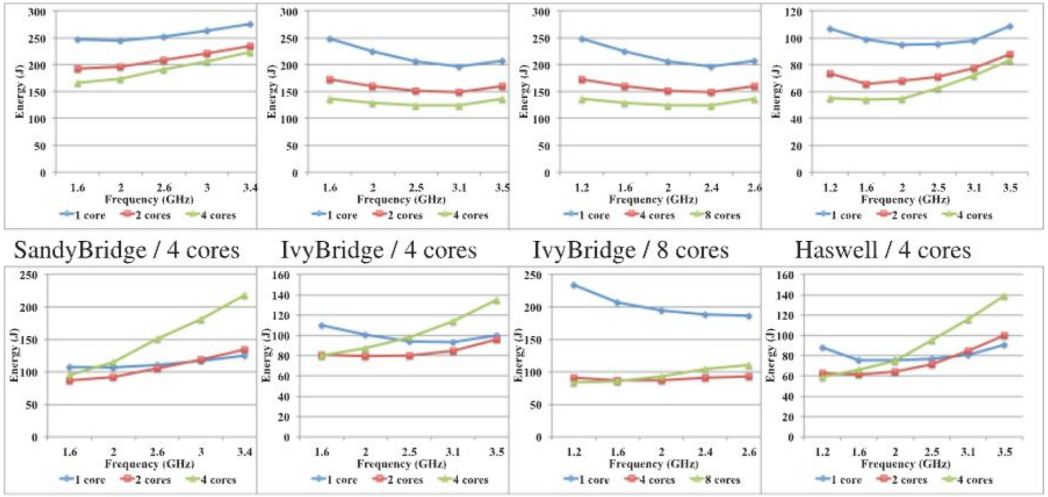
Check also: https://github.com/modin-project/modin

5 – High-performance Python libraries

However, being more performant or using all available CPU cores does not always mean less energy usage

Application 1

Application 2



https://dl.acm.org/doi/pdf/10.1145/3011017

How can we help Python developers in deciding when it makes sense to use a high-performance Python library? And which improvement should they expect?

5 – High-performance Python libraries

Informal RQ

What is the impact of using (different configurations of) high-performance Python libraries on the energy consumption of a Python application?

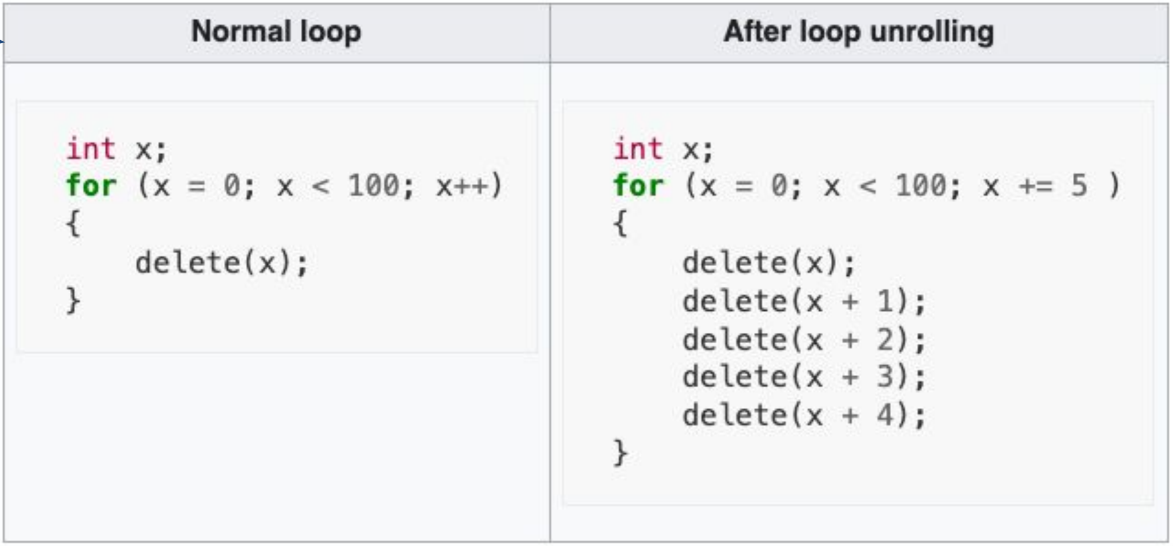
- **Subjects**: identify a set of 4-5 HPC Python libraries that are easily applicable to any piece of Python code
- **Objects**: identify a set of ~10 Python applications/benchmarks on which it makes sense to apply an HPC Python library
    - Select them in such a way that you have two different families of types of computation: memory-bound vs compute bound
- **Measurement tool**: Experiment Runner + PyJoules
- **Suggestions for main factors**:
    - Applied HPC library
        - Consider also combinations
    - Family of the object (mem-bound vs compute-bound)
- Example of related work: EASE 2024 publication

6 – Python loop optimization techniques

In the literature, there are several techniques for optimizing the execution of loops in terms of performance and memory usage. **Are they impacting also energy? If yes, how?**

From another research we identified the following loop optimization techniques for Python code:

- use a variable as loop termination condition
- loop unrolling
- loop unswitching
- early loop termination
- loop fusion
- etc.



Normal loop

```
int x;
for (x = 0; x < 100; x++)
{
    delete(x);
}
```

After loop unrolling

```
int x;
for (x = 0; x < 100; x += 5 )
{
    delete(x);
    delete(x + 1);
    delete(x + 2);
    delete(x + 3);
    delete(x + 4);
}
```

https://en.wikipedia.org/wiki/Loop_unrolling

6 – Python loop optimization techniques

Informal RQ

What is the impact of using different loop optimization techniques on the energy consumption of a Python application?

- **Subjects**: the set of loop optimization techniques
  - We manually extracted them from these two scientific studies:
    - Software Development Lifecycle for Energy Efficiency: Techniques and Tools
    - Toward efficient interactions between Python and native libraries
- **Objects**: identify or create a set of ~100 Python snippets containing loops with different operations in their bodies
- **Measurement tool**: Experiment Runner + PyJoules
- **Suggestions for main factors**:
  - Applied loop optimization techniques
    - Consider also combinations (when possible)
  - Size of the collection on which the loop is iterating
    - 1 item (as baseline), 10 items, 100 items, 1000 items, 10000 items, etc

7 – Python multithreading

Context: By default Python is single-threaded (aka, slow). A possible solution would be to use one of the several available multithreading techniques. However, it is not known (scientifically speaking) what is the impact of those techniques on the energy efficiency of Python applications.

Warm-up readings:

- https://www.geeksforgeeks.org/difference-between-multithreading-vs-multiprocessing-in-python/
- https://www.geeksforgeeks.org/multithreading-python-set-1/
- https://realpython.com/intro-to-python-threading/
- https://docs.python.org/3/library/threading.html
- https://wiki.python.org/moin/GlobalInterpreterLock
- https://www.sciencedirect.com/science/article/pii/S0167739X24003650
- https://link.springer.com/chapter/10.1007/978-3-662-48096-0_35

7 – Python multithreading

Informal RQ

What is the impact of using different multithreading techniques on the energy consumption of a Python application?

- **Examples of multithreading libraries**:
  - Parallel Python module
  - `Threading` and `multiprocessing` Python standard libraries
  - mpi4py
  - etc.
- **Objects**: identify a set of benchmarks tailored to multithreaded Python, such as:
  - https://github.com/corkymaigre/multithreading-benchmarks
  - https://www.nas.nasa.gov/software/npb.html
- **Measurement tool**: Experiment Runner + PyJoules
- **Suggestions for main factors**:
  - Used multithreading techniques (check which ones are easily applicable on an existing snippet of Python code!)
  - Type of tasks being performed (e.g., i/O-bound, CPU-bound, memory-bound)

8 – LLM vs HTTP requests

Motivation:

- Web and mobile apps often rely on HTTP requests to retrieve content from remote servers.

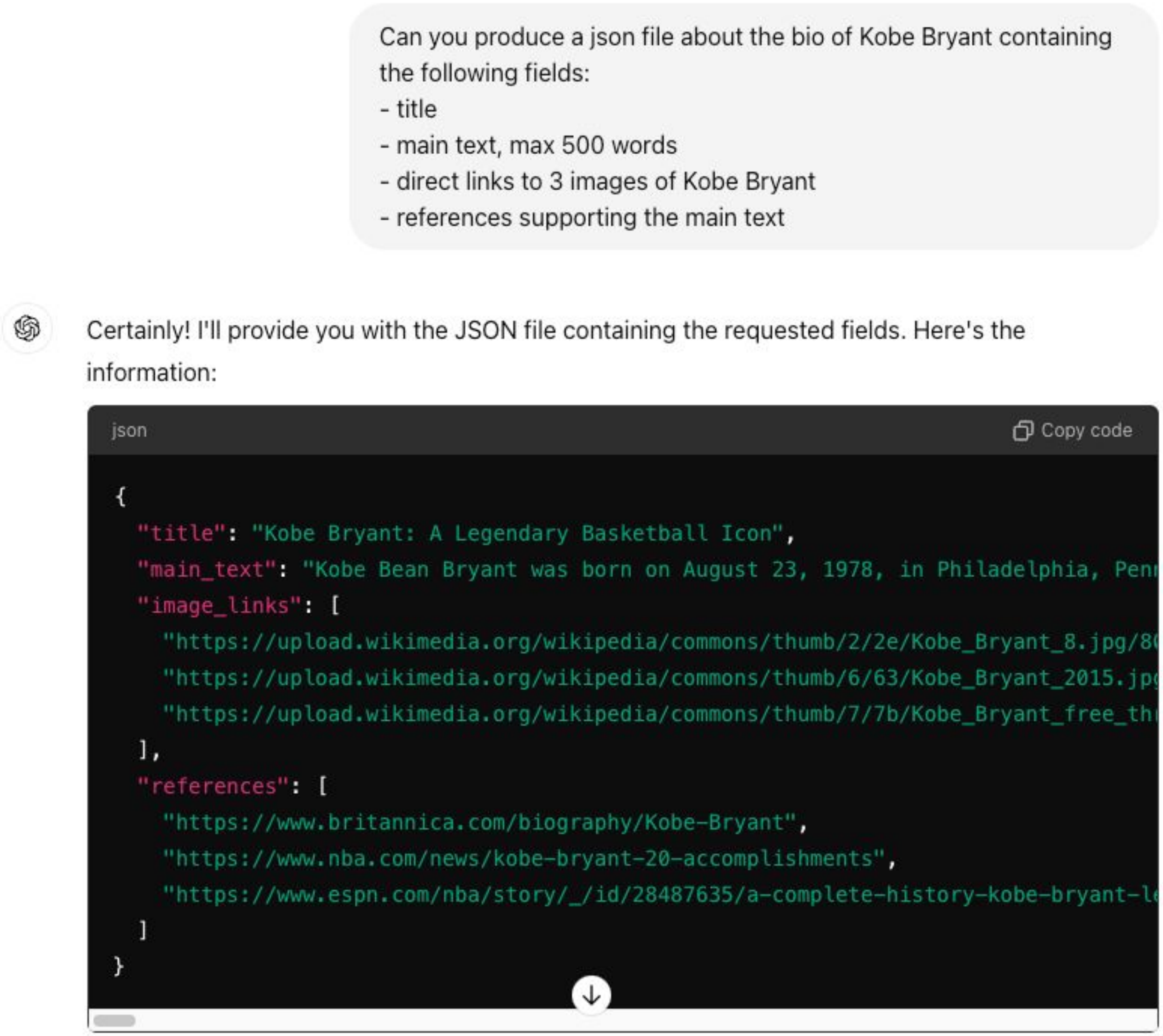
- LLMs can generate (textual) content. It is likely that in the future operating systems will embed a LLM, executed locally and accessible by user-level applications through an API.

- In this scenario, **apps could leverage the LLM to generate content locally, rather than retrieving it from remote servers**. This could potentially lead to an energy saving, due to the reduced communication cost.

- EXAMPLE: prompt the LLM to generate the bio of Kobe Bryant VS fetching the bio from Wikipedia

8 – LLM vs HTTP requests

Example with ChatGPT



Can you produce a json file about the bio of Kobe Bryant containing the following fields:
- title
- main text, max 500 words
- direct links to 3 images of Kobe Bryant
- references supporting the main text

Certainly! I'll provide you with the JSON file containing the requested fields. Here's the information:

```json
{
  "title": "Kobe Bryant: A Legendary Basketball Icon",
  "main_text": "Kobe Bean Bryant was born on August 23, 1978, in Philadelphia, Pen
  "image_links": [
    "https://upload.wikimedia.org/wikipedia/commons/thumb/2/2e/Kobe_Bryant_8.jpg/8
    "https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/Kobe_Bryant_2015.jp
    "https://upload.wikimedia.org/wikipedia/commons/thumb/7/7b/Kobe_Bryant_free_th
  ],
  "references": [
    "https://www.britannica.com/biography/Kobe-Bryant",
    "https://www.nba.com/news/kobe-bryant-20-accomplishments",
    "https://www.espn.com/nba/story/_/id/28487635/a-complete-history-kobe-bryant-l
  ]
}
```

8 – LLM vs HTTP requests

Goal: measure and compare the energy consumption of retrieving content through HTTP vs generating the content locally leveraging a LLM

Experiment sketch:

- Select one or more LLMs that can be run locally on your laptop (e.g., Meta's Llama, or those hosted by Ollama)
- Write prompts to generate structured content that can be displayed by a web or mobile application (see example in the previous slide)
- Retrieve the same content through HTTP :
  - Example: retrieve the same content from WikiMedia: https://en.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exlimit=max&explaintext&titles=Kobe_Bryant
- Measure and compare the energy consumption of the two approaches
- The focus is on the Client side, so both the LLM and the website being measured will run on your (Unix-based) laptop
- Possible co-factors: size of the contents in bytes, type of content (text-based, multimedia-heavy), domain (e.g., sport, news, history, etc.)

Challenge: Likely, there is also a tradeoff in terms of quality of the data shown to the user: is it acceptable? Can we measure it?

9 – Evolution of LLMs energy consumption

Context: LLM models such as Chat-GPT, Gemini, and Llama are continuously updated to newer versions to improve their functionality. Part of burden that has to be paid for the technological breakthrough LLMs entail is their considerable energy consumption. *As models are continuously updated to make them smarter, are they also becoming more energy efficient or not?*

9 – Evolution of LLMs energy consumption

Informal RQ

As LLMs are improving with every version, are they also becoming more energy efficient or not?

The focus is on the inference, not training of the LLMs.

Experiment Sketch:

1. Select one or more LLM with *incremental versions* that can be *run locally* (e.g., Meta's Llama could be a good option)
2. Deploy the models locally and set up energy measurement framework via Experiment Runner
3. Automatically benchmark the models automatically by executing prompts (e.g., by sign the library DeepEval)

Challenges:

- LLMs with many parameters, e.g. Llama 3.1 405B, might not be deployable on available hardware. Models might be selected by considering a significance-feasibility trade-off, or use quantized versions (e.g., Ollama models)
- LLMs testing frameworks might require some getting used to

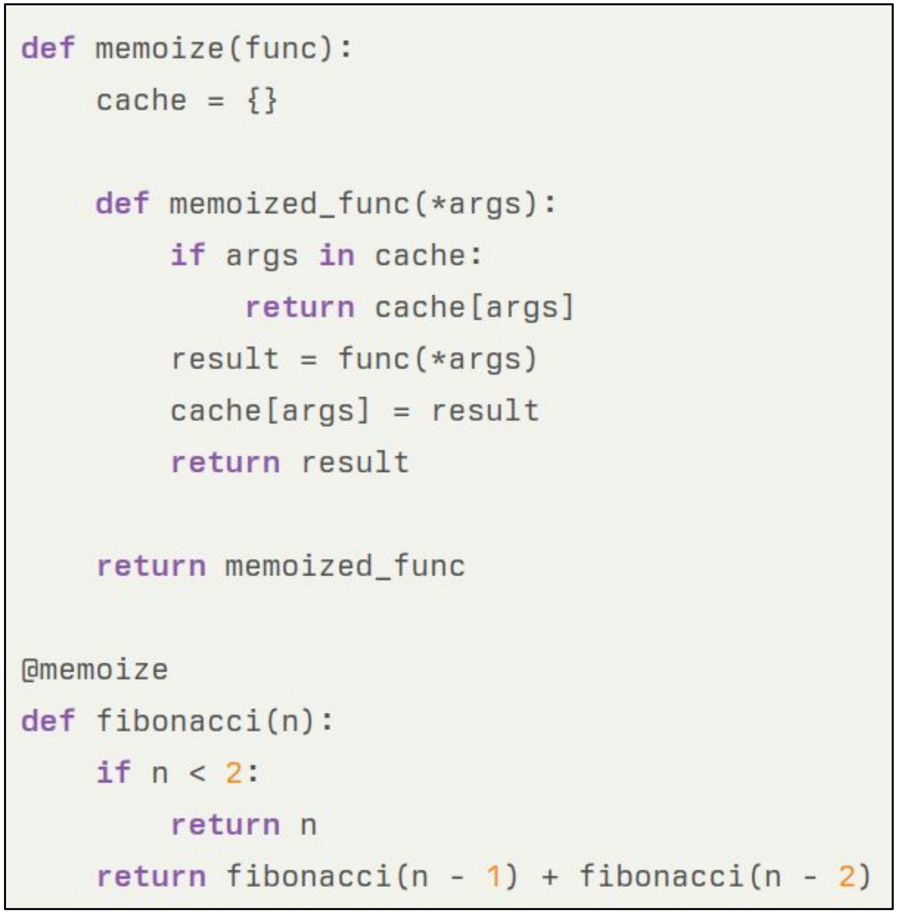
10 – Memoization

Memoization consists of storing frequently calculated results in a cache that is used to avoid redundant operations.

(see page 387 in *Introduction To Algorithms* (3e), Cormen et al.)

Memoization introduces a performance improvement. However, to our knowledge, nobody studied the impact of memoization on energy efficiency.

Study **to what extent** memoization affects the energy efficiency, CPU, and memory utilization of Python code.



```python
def memoize(func):
    cache = {}

    def memoized_func(*args):
        if args in cache:
            return cache[args]
        result = func(*args)
        cache[args] = result
        return result

    return memoized_func

@memoize
def fibonacci(n):
    if n < 2:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)
```

10 – Memoization

Analyze **a set of Python** functions and their memoized counterparts with respect to their energy consumption, CPU, and memory utilization.

Challenges/Suggestions:

- The selected functions should be **complex** and **different** (i.e., a mix of CPU and I/O-bound code) and that can be memoized.

- It should be possible to assess the correctness of the code. For example, a function should come with a set of tests.

- The data structure used for memoization might have an impact on the results.

- You can use existing Python constructs to apply memoization, such as functools.cache and quantify their impact

Reading:

Luca Della Toffola, Michael Pradel, and Thomas R. Gross. 2015. Performance problems you can fix: a dynamic analysis of memoization opportunities. SIGPLAN Not. 50, 10 (October 2015), 607–622. https://doi.org/10.1145/2858965.2814290

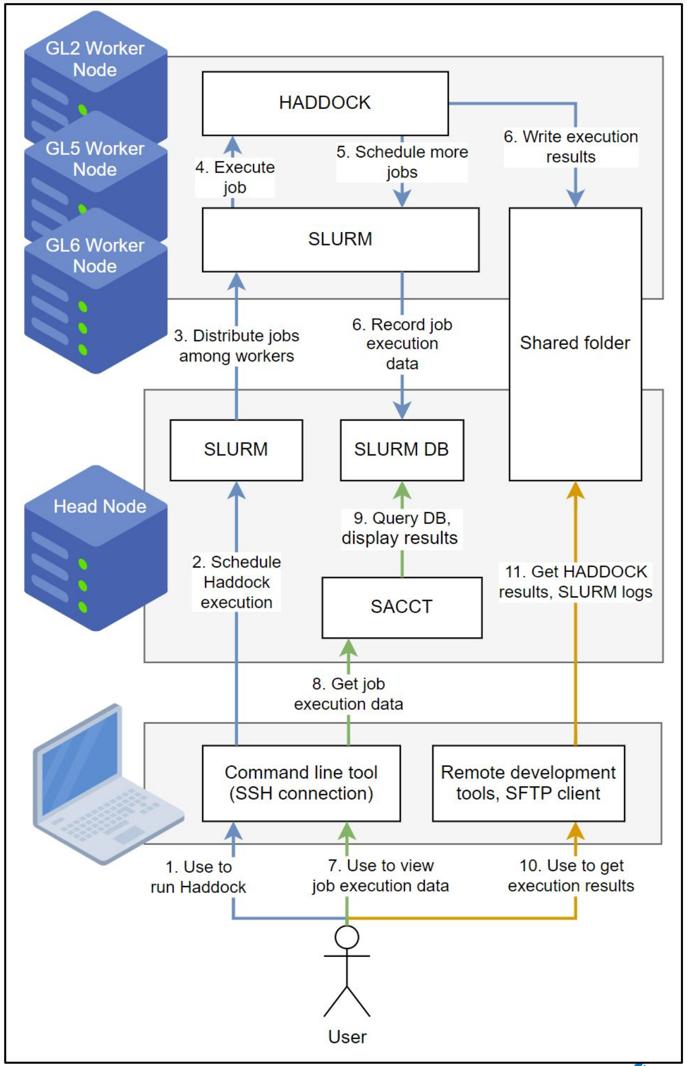
11 – Energy Efficient Job Scheduling (HADDOCK)

High-Performance Computing (HPC) software enables the **processing of large data volumes** and performs computations much faster than standard computers.

It typically **uses resources across multiple machines** and employs workload managers, like SLURM, for efficient resource allocation.

SLURM assigns to each job (i.e., computational tasks) an amount of resources.

HADDOCK is an HPC tool for molecular docking that uses SLURM.



GL2 Worker Node
GL5 Worker Node
GL6 Worker Node
HADDOCK
4. Execute job
5. Schedule more jobs
6. Write execution results
SLURM
Shared folder
3. Distribute jobs among workers
6. Record job execution data
SLURM
SLURM DB
Head Node
9. Query DB, display results
2. Schedule Haddock execution
SACCT
11. Get HADDOCK results, SLURM logs
8. Get job execution data
Command line tool (SSH connection)
Remote development tools, SFTP client
1. Use to run Haddock
7. Use to view job execution data
10. Use to get execution results
User

11 – Energy Efficient Job Scheduling (HADDOCK)

Goal: given a set of HADDOCK jobs, understand the energy and performance trade-offs of executing the jobs sequentially using all the available CPU cores or execute them in parallel distributing the cores among jobs. Other people had the same question.

Example. execute 4 jobs sequentially (one at a time) using 32 cores vs. execute 4 jobs at the same time assigning 8 cores to each job.

Challenges:

- Use HADDOCK3 as subject of your experiment
- Study how to use SLURM, a workload manager.
- Study how to allocates CPU cores to the jobs.

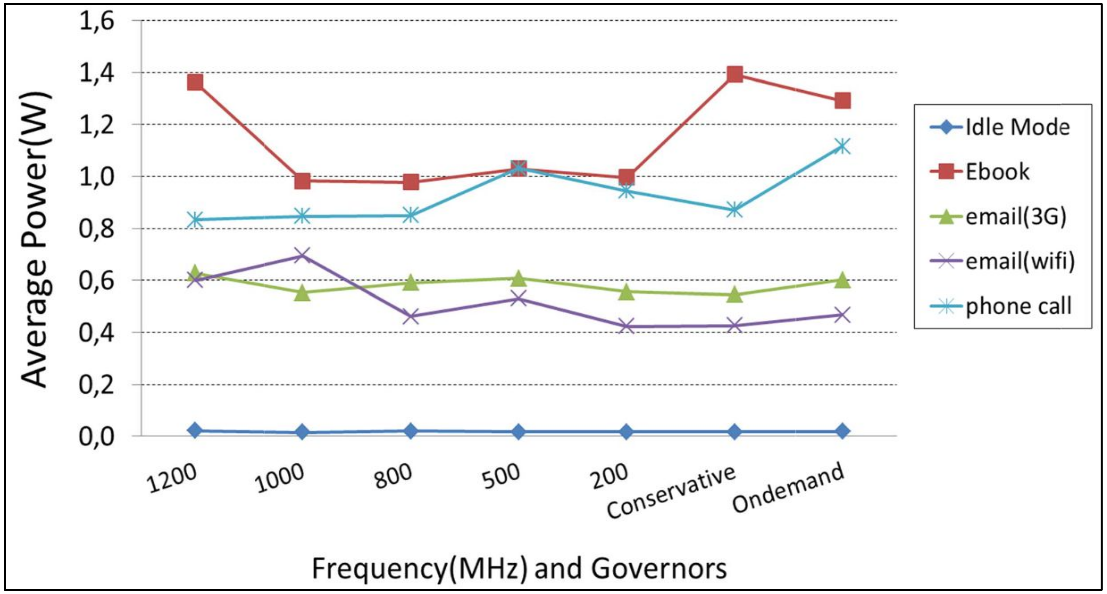
Suggestions:

- Define the characteristics of the jobs.
- Blocking Factor: CPU Frequency Governor
- Profiling at core-level: use \`sar\`
- Do not trust CPU Utilization

12 – Energy Consumption in Microservices Architectures

Dynamic Voltage and Frequency Scaling (DVFS) aims to reduce power consumption by adjusting voltage and frequency dynamically during software execution.

Linux provides several governors that dynamically set CPU frequency. Mahajan et al. evaluate the effects on performance and power consumption of three Linux governors on the execution of NOSQL queries.



Bezerra et al. "Dynamic frequency scaling on android platforms for energy consumption reduction." In *Proceedings of the 8th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pp. 189-196. 2013.

12 – Energy Consumption in Microservices Architectures

This projects investigates the effects of DVFS and, thus, Linux governors on the energy efficiency and performance of software.

Example: In this project you can choose a set of applications at will.

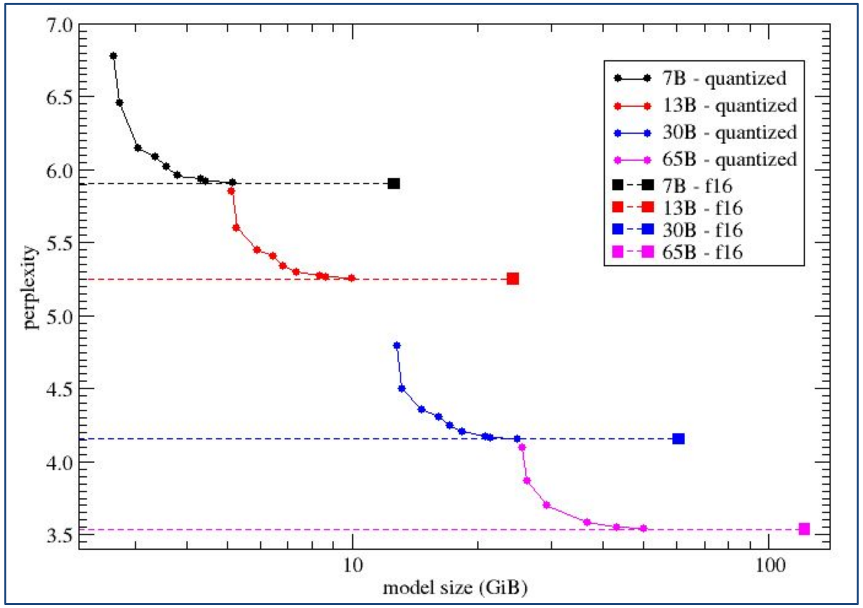
We usually use Train-Ticket Booking System. It is important to clarify which kind of workload the application undergoes and which resources are more stressed. It would be preferable to see the effects of DVFS under different workload: memory, disk, and CPU intensive or, if the evaluation involves a single resource, different percentages of utilization.

Train-Ticket Booking System can be stressed with HTTP requests generated using Locust or JMeter.

Dependent Variable: Energy Consumption, CPU and Memory Utilization
Independent Variables: Governors, Workloads

13 – Quantized LLMs

LLMs typically use high-precision values, such as 32-bit floating-point numbers. This level of precision is often unnecessary for the model's overall performance. By applying **quantization**, these values are converted into lower-precision formats, such as 8-bit or even 4-bit integers.



Quantization can lead to performance degradation. *What about energy efficiency and other resource utilization?*

[**Perplexity**: quantifies how well a probability model predicts a sample]

13 – Quantized LLMs

We want to quantify whether quantization translates to increased energy efficiency and whether it is worth sacrificing accuracy to benefit energy efficiency.

We want to compare the energy consumption of NLP models against their quantized version. Focus on Post-Training Quantization (PTQ). A set of quantize models can be found on

HuggingFace, Ollama, Llama.cpp.
Examples:

- Llama-3-8b vs Llama3-8b-Quantized
- Mixtral-8x7B-Instruct vs Mixtral-8x7B-Instruct-Quantized

Reading:

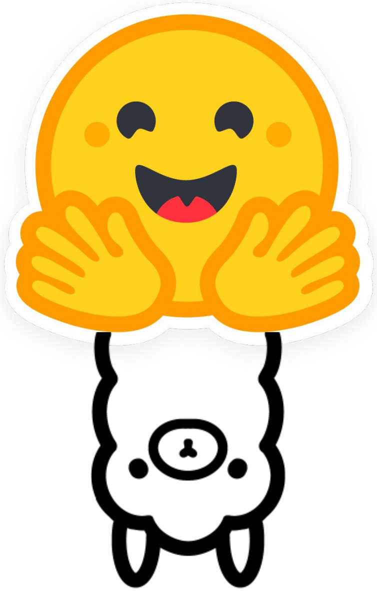
- Huang et al. - An Empirical Study of LLaMA3 Quantization: From LLMs to MLLMs
- https://www.tensorops.ai/post/what-are-quantized-llms

Example of Experiment Design:

- The models can be stressed using the GLUE benchmark or the benchmarks used by Huang et al.
- *Dependent Variable:* Energy Consumption, Accuracy, CPU Utilization
- *Independent Variables*: Tasks, Models

14 – Ollama vs HuggingFace

Motivation: Running LLMs locally is a very intensive task that requires the used hardware to be reasonably up to date, but even so, what strategy is the best for deploying models locally. Ollama or Hugging Face transformers (HF)?

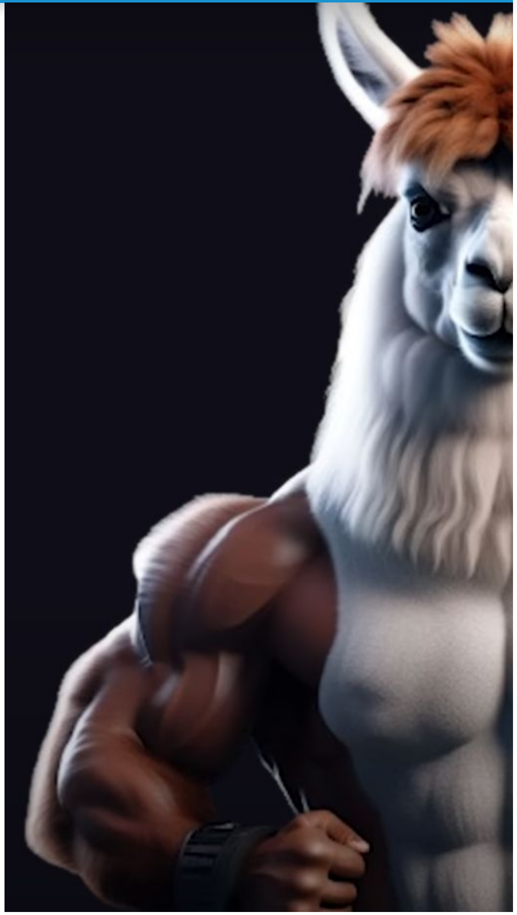


Possible questions that could be answered

Do the quantized models in Ollama provide the same outputs as HF?

Is Ollama more energy efficient than HF for certain tasks, or for different output lengths?

15 – Llama 3.1 generated code EE



Motivation: With the (reasonably) recent release of the Llama 3.1 models, and with developers using LLMs more and more, we'd like to investigate if the latest and greatest in terms of LLMs can stack up against the challenge of energy efficiency (EE).

Possible Experiment Design:

Baseline - select a set of coding problems, prompt the LLM to solve them, and measure EE of generated code.

Variations - modify the prompt to guide the LLM to a more EE solution, and measure EE of generated code

16 – Energy Efficiency Comparison of LLMs

Motivation: In the past year, large language models have become much more popular and user-friendly. The 2023 Stack Overflow annual developer survey revealed that 44% of the programmers are already utilizing AI tools.

Cursaru et al. investigated the energy efficiency of code generated by Code Llama, comparing it with human-written code across various problems and programming languages.

There is still a gap in understanding how different LLMs compare in terms of energy efficiency during code execution.

16 – Energy Efficiency Comparison of LLMs

Goal: The objective of this project is to compare the energy efficiency of code solutions generated by different large language models (LLMs) during their execution to quantify the sustainability performance of models.

We can compare the energy efficiency of widely-used models:(ChatGPT, Code Llama, Claude, etc.)

Existing code can be taken from CLBG benchmark or Leetcode, some common algorithms can also be used

Example of Experiment Design:

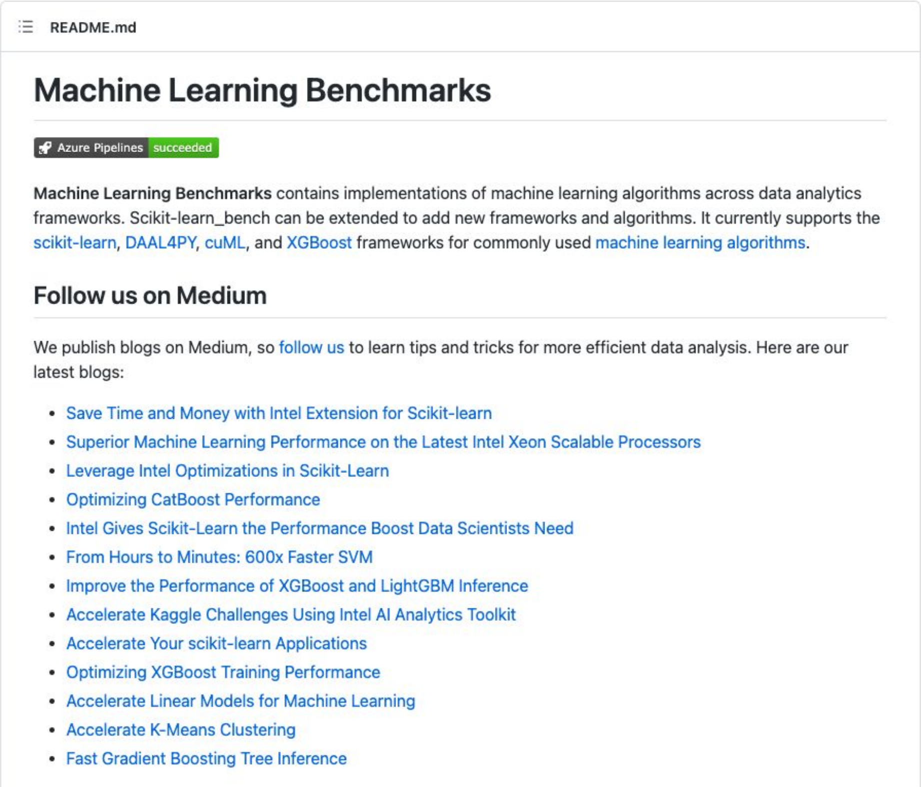
- *Dependent Variable:* Energy Consumption(and metrics like CPU/memory usage, etc.)
- *Independent Variables*: Benchmarks, Programming Language, Large Language Models

17 – Greener ML software

Informal RQ

How to improve ML software from an energy point of view?

https://github.com/IntelPython/scikit-learn_bench



Machine Learning Benchmarks

Machine Learning Benchmarks contains implementations of machine learning algorithms across data analytics frameworks. Scikit-learn_bench can be extended to add new frameworks and algorithms. It currently supports the scikit-learn, DAAL4PY, cuML, and XGBoost frameworks for commonly used machine learning algorithms.

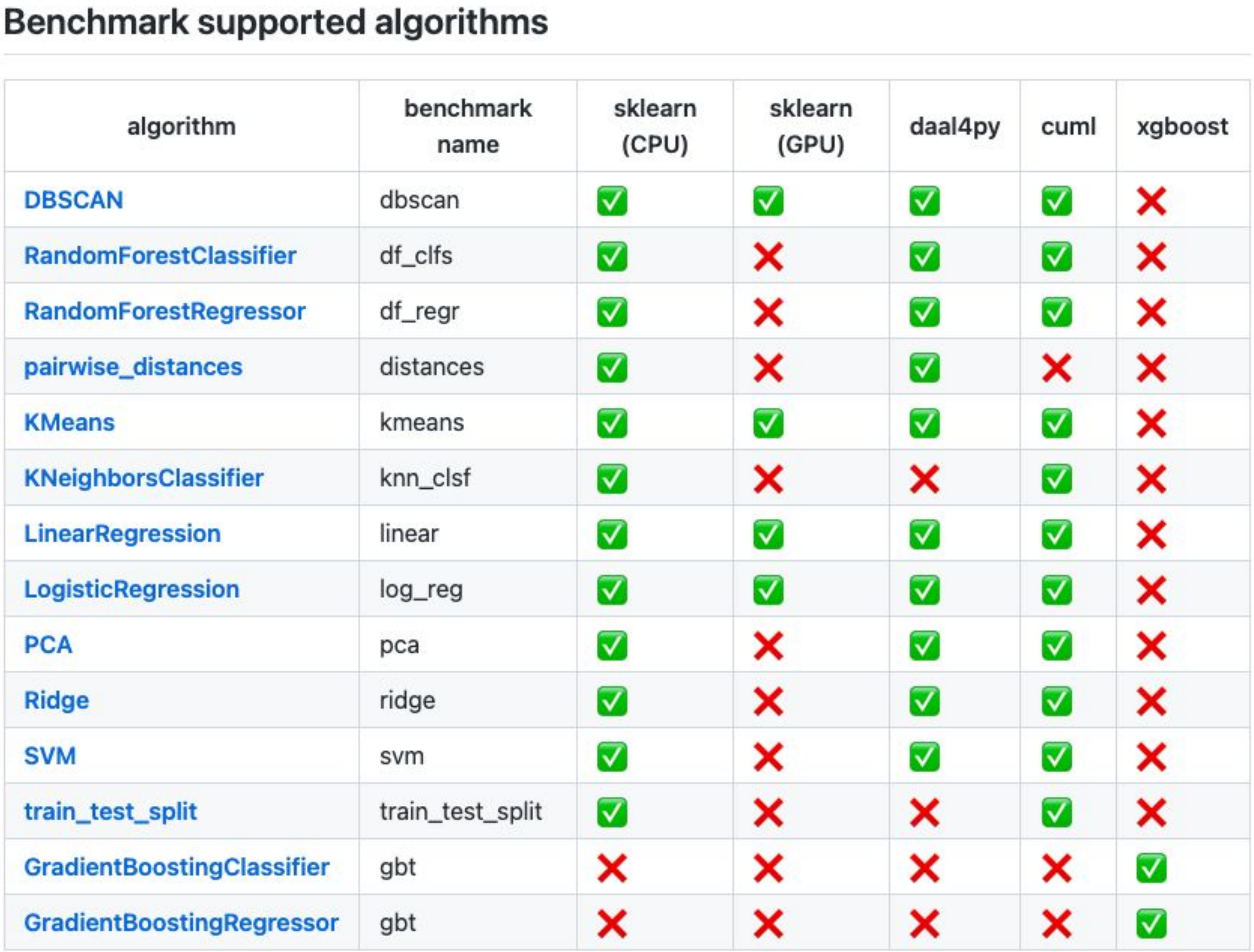
Follow us on Medium

We publish blogs on Medium, so follow us to learn tips and tricks for more efficient data analysis. Here are our latest blogs:

- Save Time and Money with Intel Extension for Scikit-learn
- Superior Machine Learning Performance on the Latest Intel Xeon Scalable Processors
- Leverage Intel Optimizations in Scikit-Learn
- Optimizing CatBoost Performance
- Intel Gives Scikit-Learn the Performance Boost Data Scientists Need
- From Hours to Minutes: 600x Faster SVM
- Improve the Performance of XGBoost and LightGBM Inference
- Accelerate Kaggle Challenges Using Intel AI Analytics Toolkit
- Accelerate Your scikit-learn Applications
- Optimizing XGBoost Training Performance
- Accelerate Linear Models for Machine Learning
- Accelerate K-Means Clustering
- Fast Gradient Boosting Tree Inference

17 – Greener ML software

https://github.com/IntelPython/scikit-learn_bench



Benchmark supported algorithms

algorithm	benchmark name	sklearn (CPU)	sklearn (GPU)	daal4py	cuml	xgboost
DBSCAN	dbscan	✅	✅	✅	✅	❌
RandomForestClassifier	df_clfs	✅	❌	✅	✅	❌
RandomForestRegressor	df_regr	✅	❌	✅	✅	❌
pairwise_distances	distances	✅	❌	✅	❌	❌
KMeans	kmeans	✅	✅	✅	✅	❌
KNeighborsClassifier	knn_clsf	✅	❌	❌	✅	❌
LinearRegression	linear	✅	✅	✅	✅	❌
LogisticRegression	log_reg	✅	✅	✅	✅	❌
PCA	pca	✅	❌	✅	✅	❌
Ridge	ridge	✅	❌	✅	✅	❌
SVM	svm	✅	❌	✅	✅	❌
train_test_split	train_test_split	✅	❌	❌	✅	❌
GradientBoostingClassifier	gbt	❌	❌	❌	❌	✅
GradientBoostingRegressor	gbt	❌	❌	❌	❌	✅

17 – Greener ML software

Main steps before running the experiment:

1) Be able to execute all ML algorithms across all ML frameworks (with timestamps!)
   - ○ Tool: https://github.com/powerapi-ng/pyJoules
2) Define the plan of interventions

   - aka the settings/configurations that you want to compare

   - see here for some examples on scikit-learn (but there are more – get creative!)

NOTE: this experiment can be run on either:

- a **board** (--> IoT flavour to your research) or

- your **laptop**

It is up to you to choose which flavour you want to give to your project

Relevant reading:

- Video: https://www.dropbox.com/s/ak4bl3f9mf0h65q/video1065367371.mp4?dl=0
- Paper: https://stefanos1316.github.io/my_curriculum_vitae/GKSSZ22.pdf

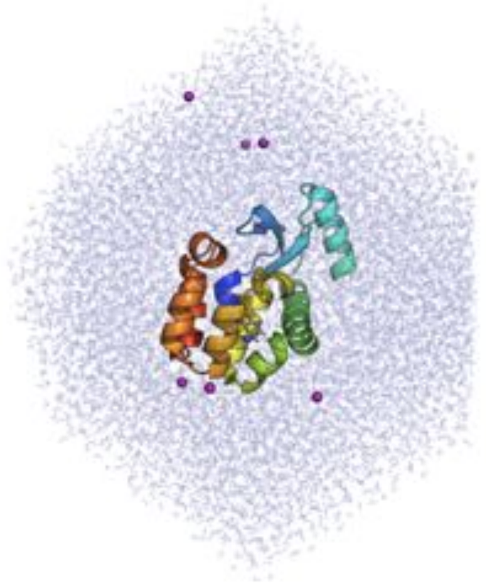
18 – Green Molecular Dynamics Simulations

Molecular Dynamic (MD) is a computational technique to simulate atomic and molecular movements in three-dimension and motion, governed by physical laws and particle interactions. This method has different applications, in bioinformatics is commonly used for proteins and protein ensembles simulation.

Motivation: due to the large amount of atoms taken into account in a MD simulation, this technique is extremely computationally expensive. MD algorithms have several steps which have to be repeated for every atom in the protein. While a simulation of a moderately sized protein (around 100 000 atoms) might be run locally, larger systems use several thousands of CPUs and need to be explored on HPCs.

18 – Green Molecular Dynamics Simulations

Potential research question: What strategies can be employed to reduce the computational expense of Molecular Dynamics simulations and how do these methods affect the accuracy and efficiency of the simulations?



Examples of approaches to explore:

- Use of different MD softwares (e.g. GROMACS, NAMD, AMBER)
- All-atoms vs coarse- grained simulations
- Parallelization of CPUs and GPUs

19 – bioScience

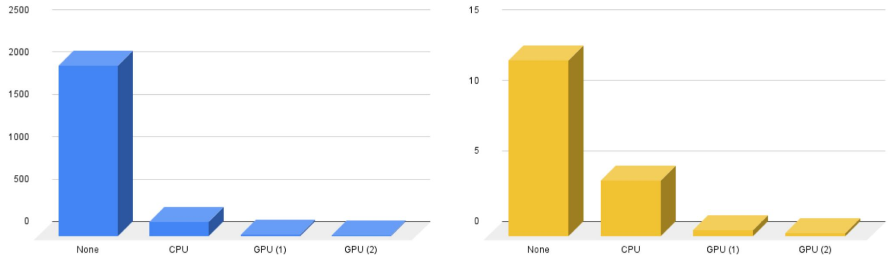
Motivation: Increasing need to adapt data analysis methodologies and computational approaches to handle the vast amount, diversity, speed, and authenticity of the data being currently produced, transmitted, and analyzed.

Biclustering can detect intrinsic similarity in data, identifying instances exhibiting similar behavior according to a subset of their characteristics (e.g.: only some samples share similarities on some features). It is becoming an increasingly significant technique in the analysis of complex data, particularly genetic data. It is an NP-hard computational problem with heuristic-based workarounds to enhance efficiency.

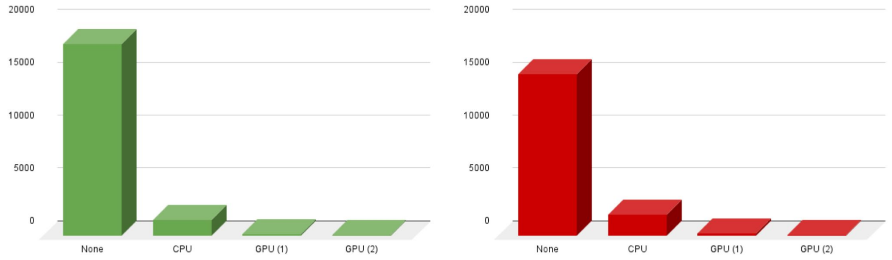
HPC approaches: parallel or distributed computing, GPU usage.

19 – bioScience

Goal: measure the performance of biclustering algorithms and the impact of enhancement heuristics. Consider the effect of the number of rows and columns, and properties of the data sets such as sparsity, nonnegativity and variance. Determine whether different preprocessing techniques have a significant impact on the use of energy and other resources.



(a) Synthetic (b) Microarray



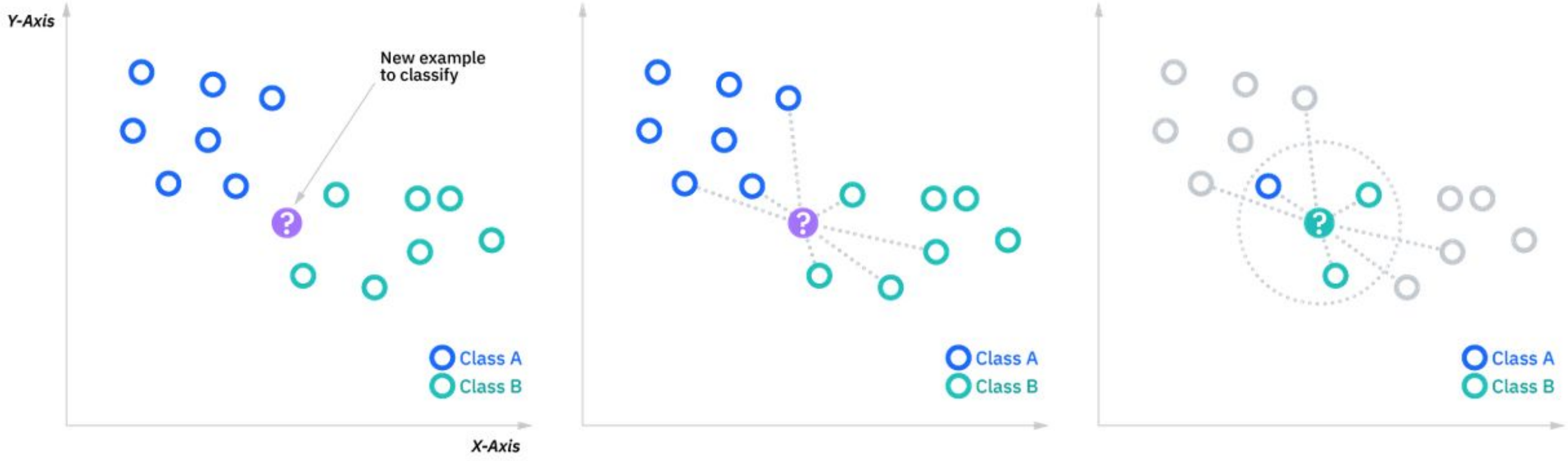
(c) RNA-Seq (d) single-cell

Visual representation of the time of execution for the experimentation performed for each dataset
Source: https://doi.org/10.1016/j.softx.2024.101666

20 – ML Algos

Motivation: AI algorithms have been used in industry long before the advent of generative AI. Majority of AI algorithms deployed on industrial scale are classical machine learning algorithms.

Efficient implementation becomes key factor when considering power consumption of the said algorithms. Algorithm to consider in this project is **k-NN classifier**.



20 – ML Algos

Goal: Measure performance and resource consumption for 2 different implementations of the same algorithm - naive and optimized one.

With the profiling tools presented in the laboratories, students will collect data from running training and inference sessions. A research hypothesis will be stated and tested with experimental data analysis in R.

k-NN is quite computation intensive when considering large data sets, so efficient implementation can heavily influence its performance.

GO!