Brains Without Brawn: Evaluating CPU Performance for Code Generation with Large Language Models

Miren Illarramendi[1] Joseba Andoni Agirre[1] Aitor Picatoste[2] Juan Ignacio Igartua[2]

[1]Software and Systems Engineering Research Group
[2]Circular Economy and Industrial Sustainability
Faculty of Engineering, Mondragon University, Spain

IARIA Conference – 2025

Presenter: Miren Illarramendi | millarramendi@mondragon.edu

Presenter Bio

Miren Illarramendi
Lecturer Researcher, Mondragon University

Researcher in the *Software and Systems Engineering Research Group*.

Research interests:

Sustainable AI and Green Computing

Code generation with LLMs

Energy-aware software development

DevOps integration with environmental monitoring (CodeCarbon, MLflow)

Autonomous systems reliability

Autonomous systems runtime monitoring

Research Group & Current Projects

Software and Systems Engineering Research Group
Faculty of Engineering – Mondragon University

Key Research Areas:

- Green and Sustainable AI
- LLM efficiency in low-resource environments
- AI for industrial digitalization
- ACPSs: Autonomous CPSs
- Energy-aware MLOps/DevOps pipelines

Current Projects:

- **GRECO Elkartek (KK-2024/00090)** – AI for circular economy
- **Ikerketa Taldeak (IT1519-22)** – Basque Gov't–funded software engineering research
- **MSCA-DN** – HE Innoguard Doctoral Network
- **LLM Benchmarking on CPUs** – This paper's study

We welcome collaboration on sustainable AI, code generation, and green software engineering!

Motivation & Problem Statement

- Large Language Models (LLMs) are transforming software development and software engineering to improve efficiency and quality in the final results — but most studies assume **GPU availability**.
- Many developers or industrial environment only have **CPUs** (local machines, edge devices, cost-limited cloud).
- **Gap**: Lack of empirical analysis on CPU-only inference for code generation.
- Need to understand trade-offs: **accuracy vs. speed vs. energy vs. $\mathbf{CO_2}$**.

Goal: Evaluate LLMs for code generation in CPU-constrained, GPU-free environments.

Evaluated Large Language Models

OpenAI Models (via API):

- `gpt-4o`
- `gpt-4-turbo`
- `gpt-3.5-turbo`
- `gpt-4o-mini`

Hugging Face Models (via Novita API):

- `Mistral-7B-Instruct`
- `Llama-3-8B-Instruct`
- `WizardLM-2-8x22B`
- `Qwen3-235B-A22B`

Parameter count ranges from **3.8B** to **235B**.
All inference performed **remotely** from a local CPU (Intel Core i5-1135G7, Windows 11).

Experimental Setup

Hardware: Intel Core i5-1135G7 @ 2.40 GHz, Windows 11 Pro
Tasks: 5 C programming prompts:

- Prime number check:*Prompt: Write a C program that checks if a number is prime. The program validation returns 1 if it is prime and 0 if not. The number to check is 11.*
- Greatest of three integers: *Prompt: Write a C program that defines three integer variables and prints the greatest of them. The numbers to check are 11, 22, and 33*
- Even/odd check
- Absolute difference
- Sum of digits

Methodology:

Each model ran **10 times per task** (50 inferences total)

Code validated with gcc for correctness

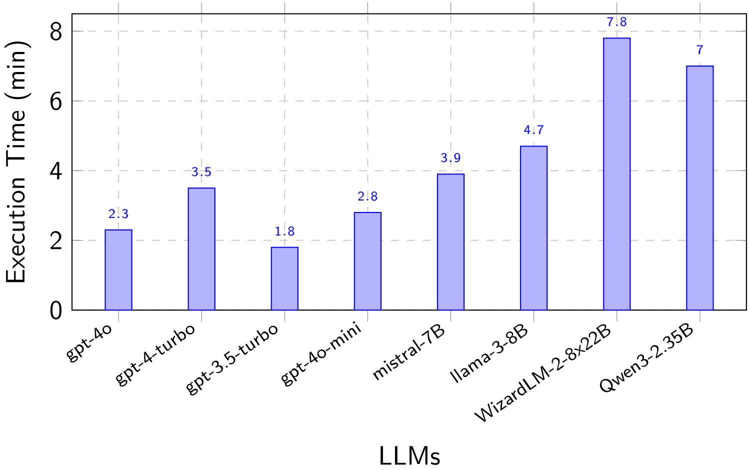
Metrics tracked via **MLflow** (time, accuracy) and **CodeCarbon** (energy, CO_2)

Code

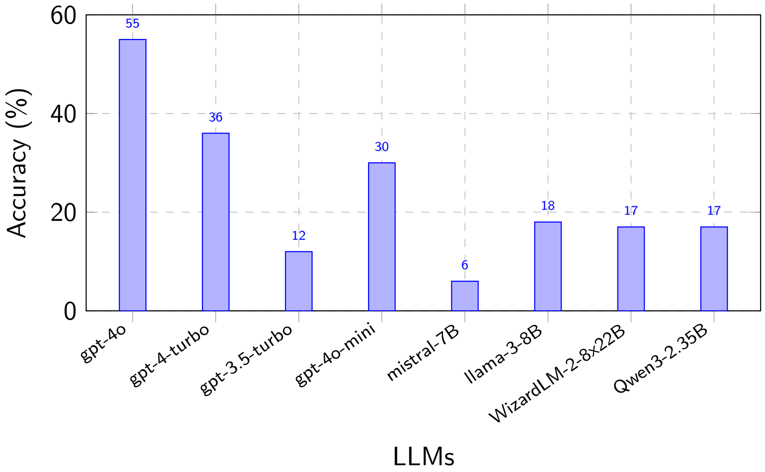
LLMCodeGen_Green_HF.py > ...

```python
def main():
            for j in range(len(eval_user_messages)):
                eval_user_message = eval_user_messages[j]
                i = 0
                test_evaluator = TestEvaluatorLlama(model, eval_system_message, eval_user_message)
                for i in range(total_code_count):
                    try:
                        generated_code = test_evaluator.evaluateLLMOuput()
                        print(generated_code)
                        if validate_code(generated_code,j):
                            correct_code_count += 1
                        i += 1
                    except ValueError as e:
                        print(f"Error during code generation: {e}")
                # Calcular la precisión (accuracy)
            accuracy = correct_code_count / (total_code_count*j)
            # Medir el tiempo de ejecución
            execution_time = time.time() - start_time
            # Detener el seguimiento de energía con CodeCarbon
            tracker.stop()

            # Log Carbon footprint to MLflow
            emissions_data = tracker._prepare_emissions_data()
            emissions_data_delta = tracker._compute_emissions_delta(emissions_data)


            tracker._persist_data(
```

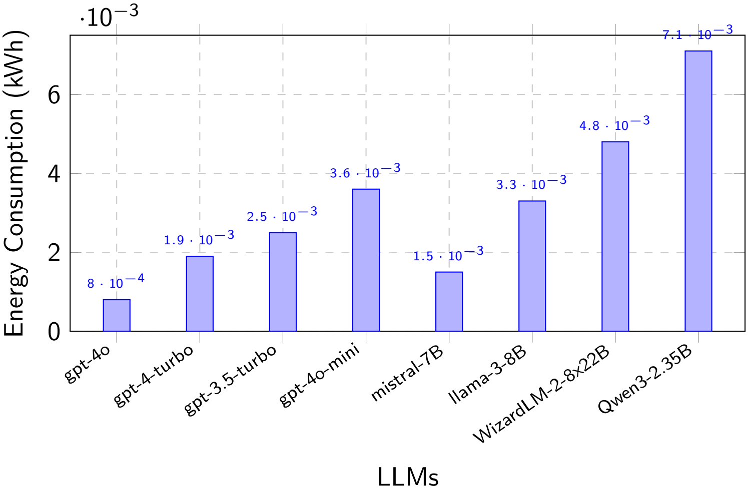
Results: Execution Time vs LLMs



Results: Accuracy vs LLMs



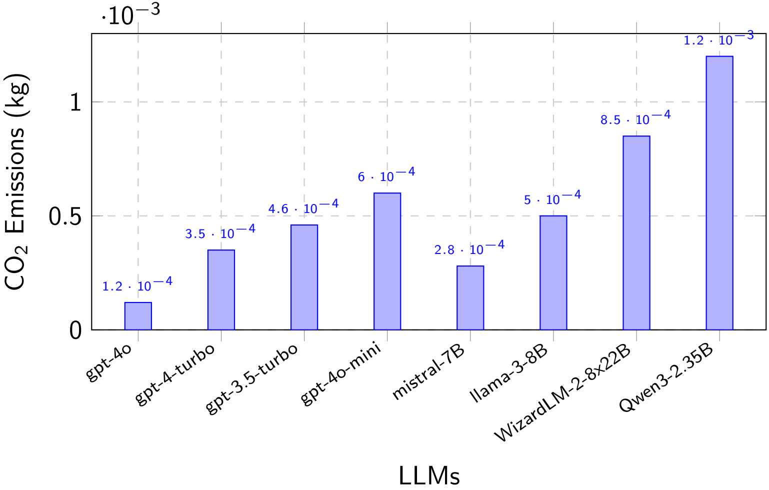
Results: Energy Consumption vs LLMs



- GPT-4o achieves high accuracy with relatively low energy use.
- Qwen3-235B consumes **9× more energy** than GPT-4o.

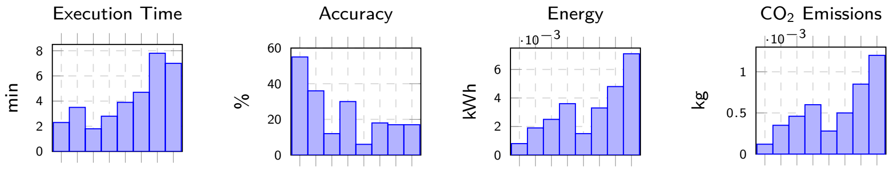
Results: CO_2 Emissions per Inference vs LLMs

Emissions (kg CO_2) tracked via CodeCarbon:



- Qwen3-235B emits **42× more CO_2** than GPT-4o.
- Smaller models are not always greener—GPT-4o offers best accuracy-per-emission ratio.

Comparison of LLM Performance Metrics



Models: GPT-4o, GPT-4-turbo, GPT-3.5-turbo, GPT-4o-mini, Mistral-7B, Llama-3-8B, WizardLM-2-8x22B, Qwen3-2.35B

Practical Trade-offs for Developers

Model	Accuracy	Time	CO_2
GPT-4o	**54%**	2.3 min	**0.00013 kg**
GPT-4o-mini	30%	2.8 min	0.00030 kg
Mistral-7B	6%	3.9 min	0.00027 kg
Qwen3-235B	17%	72 min	0.00549 kg

Key insight: **GPT-4o** dominates in accuracy and sustainability. **GPT-4o-mini** is the best compromise for resource-constrained settings.

Limitations

- All inference was performed **remotely** via APIs (OpenAI, Novita).
- Measurements include **network latency** and backend black-box effects.
- Energy estimates for proprietary models (e.g., GPT-4o) are **approximate** (CodeCarbon uses regional grid data).
- Tasks limited to **simple C programs** — may not generalize to complex software.
- No local CPU deployment yet (planned for future work).

Conclusion

- LLMs exhibit significant trade-offs in CPU-initiated code generation.
- **GPT-4o** delivers the best balance of accuracy, speed, and low emissions.
- **GPT-4o-mini** is a strong alternative when cost or moderate performance is acceptable.
- Massive models like `Qwen3-235B` are **inefficient** in this context—slow, inaccurate, and high-emission.
- Sustainable AI requires evaluating models in **realistic, GPU-free environments**.

"Efficiency isn't optional—it's ethical."

Future Work

Building on this study, we plan to:

- **Expand to other LLMs**: Evaluate newer or alternative models (e.g., Llama-3.1, Claude Sonnet) to assess evolving efficiency trends.
- **Broaden task scope**: Extend beyond C code generation to include **debugging**, **refactoring**, **code optimization**, and multi-language tasks (Python, JavaScript).
- **Local CPU/GPU deployment**: Move from remote API inference to **local execution** on CPU and GPU to measure true hardware-specific performance and energy use—eliminating network and backend black-box effects.
- **Model optimization**: Apply **quantization**, **pruning**, and **distillation** to reduce resource demands while preserving code generation quality.
- **Cost-emission-accuracy modeling**: Develop a unified metric to guide model selection based on accuracy, energy, CO_2, and API cost.

Thank You!

For your attention and interest.

`millarramendi@mondragon.edu`

Mondragon Unibertsitatea Goi Eskola Politeknikoa Faculty of Engineering

Mondragon Unibertsitatea

IARIA

Project or Initiative Logo