Brains, Brawn, and Silicon: Evaluating LLM Performance for Code Generation across Intel and Apple M3 Max Architectures

Miren Illarramendi[1], Joseba Andoni Agirre[1], Aitor Picatoste[2], Juan Ignacio Igartua[2]
[1]Software and Systems Engineering Research Group, Mondragon University, Spain
Email: {millarramendi, jagirre}@mondragon.edu
[2]Circular Economy and Industrial Sustainability, Mondragon University, Spain
Email: {apicatoste, jigartua}@mondragon.edu

***Abstract*—This research presents a comparative analysis of various Large Language Models (LLMs) for code generation tasks executed without dedicated Graphics Processing Units (GPUs), focusing exclusively on CPU-based environments. While GPUs are typical for large-scale inference, many users rely on CPUs due to resource constraints. However, high-performance chips like the Apple M3 Max introduce a new paradigm for local execution. This study addresses the gap in understanding how LLMs perform across different portable hardware tiers, focusing on the trade-offs between model logic and hardware power. The study evaluates key performance metrics including inference time, code generation accuracy, CPU and memory usage, and energy consumption. To assess the influence of hardware architecture on model efficiency, experiments were conducted across two distinct hardware tiers: a resource-constrained Intel x86 CPU platform and a high-performance ARM-based system powered by the Apple M3 Max. By performing repeated experiments under controlled conditions, we analyze the impact of model size, architectural optimization, and hardware characteristics on computational efficiency. Energy consumption is measured using tools such as CodeCarbon, enabling a detailed examination of the environmental footprint associated with CPU-based LLM execution. The comparison between traditional x86 architectures and modern unified memory ARM systems provides insights into the trade-offs between model precision, computational resource usage, and energy efficiency. The findings contribute practical guidance for developers and researchers seeking to balance performance, sustainability, and hardware constraints in low-resource or portable computing environments, highlighting how architectural differences influence both operational efficiency and environmental impact.**

Keywords-LLMs; GenIA; GreenComputing; Code Generation; Energy Consumption; Sustainability; Apple M3 Max.

I. Introduction

The rapid advancement of LLMs has revolutionized various fields, including Natural Language Processing (NLP), code generation, and even machine translation. Models like GPT-3, DeepSeek, and Llama have shown remarkable capabilities in tasks ranging from text generation to understanding and generating code. However, these models are computationally expensive and require significant resources, particularly during the training and inference stages [2] [3].

While GPUs are typically the hardware of choice for running large-scale machine learning models due to their high parallel processing capabilities, not all environments have access to dedicated GPUs. Many users, particularly those in resource-constrained settings or utilizing cloud computing, must rely on local CPUs for model inference. CPUs, though less powerful than GPUs in terms of parallel processing, are widely available and more energy-efficient in certain use cases, especially for smaller models or lightweight tasks [4].

Despite the growing use of LLMs in production environments, there is a lack of comprehensive analysis comparing the performance and sustainability of these models when executed on CPUs versus GPUs. The existing literature focuses mainly on GPU-based performance, leaving a gap in understanding how LLMs perform in real-world scenarios where only CPU resources are available. Some research has pointed out that the energy consumption of LLMs is often underestimated in most studies, with the environmental impact becoming a significant concern when deploying models at scale.

This study aims to address this gap by conducting a comparative analysis of the performance of various LLMs for code generation tasks when executed in GPU-based environments remotely. Specifically, we will focus on several key metrics, including inference time, code generation accuracy, energy consumption, and computational cost. The initial phase will involve measuring the cost of running inference from our local CPU to understand the energy and computational efficiency of remote execution. The next step will be to extend this analysis by deploying the LLMs directly on our CPU for inference, allowing us to compare performance and resource usage when running these models in resource-constrained environments. Models to be evaluated include "gpt-4o", "gpt-4-turbo", "gpt-3.5-turbo", "gpt-4o-mini", "meta-llama/Meta-Llama-3-8B-Instruct", "alpindale/WizardLM-2-8x22B", and "Qwen/Qwen3-235B-A22B-Instruct-25072". These models have been selected due to their variety in size and diverse platforms (OpenAI, HuggingFace), providing a comprehensive comparison of different model architectures and inference performance across various levels of complexity and computational demands.

The primary contribution of this research is the systematic evaluation of large language models (LLMs) for code generation under real inference conditions, with particular emphasis on accuracy, execution efficiency, energy consumption, and associated CO_2 emissions. Unlike studies that focus solely on benchmark accuracy, this work integrates performance and environmental sustainability metrics, providing a holistic perspective on LLM deployment trade-offs.

Specifically, we measure and monitor code generation tasks executed across different LLM families—both proprietary cloud-based models and open-weight models—on heteroge-

neous hardware platforms. By analyzing inference behavior in resource-constrained environments, we aim to quantify how model scale, hardware architecture, and deployment configuration jointly influence the accuracy–efficiency–sustainability balance. This integrated analysis provides practical guidance for researchers, developers, and organizations seeking cost-effective and environmentally responsible AI deployment strategies.

It is important to clarify that this study focuses exclusively on *inference-time evaluation*. Model training and fine-tuning phases are not considered. Furthermore, part of the experimental evaluation relies on remote execution (API-based inference), where energy measurements may be influenced by external factors such as network latency, backend infrastructure optimizations, and limited transparency into proprietary data-center energy usage. These constraints are acknowledged as inherent limitations of current cloud-based evaluation methodologies.

A. The Hardware Democratization Challenge

Although state-of-the-art (SotA) LLMs have demonstrated remarkable capabilities, their widespread adoption is constrained by substantial hardware requirements. Traditionally, efficient execution of frontier models has required high-end data-center infrastructures equipped with specialized accelerators such as NVIDIA H100 or A100 GPUs. While suitable for centralized cloud services, this paradigm raises concerns regarding data privacy, operational costs, vendor dependency, and inference latency.

As a result, there is a growing interest in *Local AI*, where models are executed directly on user devices. However, democratizing access to LLMs requires addressing the fundamental hardware limitations of consumer-grade systems.

B. The CPU Inference Gap

For most users, local hardware consists primarily of laptops and workstations relying on Central Processing Units (CPUs), rather than dedicated AI accelerators. Transformer-based architectures are highly parallel and computationally intensive, typically benefiting from GPUs equipped with tensor cores and high-bandwidth memory (HBM). CPUs, by contrast, are traditionally perceived as suboptimal for large-scale matrix operations.

Despite this limitation, CPUs remain the most ubiquitous computing resource worldwide. Enabling efficient LLM inference on CPUs would significantly expand accessibility, reducing dependence on specialized infrastructure. This study investigates whether modern CPU architectures can narrow the so-called "CPU inference gap," particularly when evaluating energy efficiency alongside performance.

C. The Rise of ARM Architectures and Unified Memory

The emergence of Apple Silicon (M1, M2, M3 series) represents a paradigm shift in personal computing. These ARM-based architectures introduce a unified memory model in which RAM is dynamically shared between CPU and GPU components. This design reduces data transfer overheads associated with PCI-Express bottlenecks typical in traditional x86 systems.

This research evaluates whether such architectural innovations translate into measurable improvements in inference efficiency for LLM workloads. By comparing Intel x86 systems with Apple ARM-based systems, we aim to determine whether modern consumer hardware can deliver near-accelerator efficiency for inference tasks without relying on discrete GPUs.

D. Research Objectives

To address these challenges, this paper conducts a structured comparative evaluation focusing on four main dimensions:

1) **Performance Profiling:** Measuring inference time and throughput across different models and hardware configurations to understand latency scaling behavior.
2) **Architectural Comparison:** Quantifying performance and efficiency differences between traditional Intel x86 platforms and Apple ARM-based architectures.
3) **Sustainability Analysis:** Estimating energy consumption (kWh per inference) and CO_2 emissions to promote Green AI practices and environmentally responsible deployment strategies.
4) **Quality Evaluation:** Assessing functional correctness of generated code using Pass@1 metrics to ensure that efficiency gains do not compromise output quality.

By integrating these dimensions, this study contributes to bridging the gap between high-performance AI research and sustainable, hardware-aware deployment strategies.

E. Next Steps: Local Execution Benchmarking

As a continuation of this work, the next experimental phase will involve running selected open-weight models entirely locally on both CPU and GPU configurations. This will enable direct comparison between remote API-based execution and on-device inference, further clarifying trade-offs in latency, energy consumption, carbon footprint, and operational control.

Such an extension will allow the identification of optimal model–hardware pairings for resource-constrained environments and contribute to the development of reproducible and sustainable AI evaluation methodologies.

The remainder of the paper is organized as follows: Section II and Section III cover the background and related work, respectively, providing the foundational context for this study. In Section IV, we describe the experimental setup and methodologies employed. Section V presents the experimental results, focusing on performance, throughput, code generation accuracy, and energy efficiency. Section VI offers an in-depth discussion of the evaluation, interpreting the results and their implications. Finally, Section VII concludes the paper and proposes directions for future work.

II. Background

The rapid evolution of Large Language Models (LLMs) has been driven by advances in deep learning architectures, large-scale pretraining, and the availability of massive datasets. Foundational models such as GPT-3 [5] and BERT [6] demonstrated that scaling parameter counts and training data significantly improves performance across a broad range of natural language

processing (NLP) tasks, including text generation, translation, and question answering. Subsequent models, such as Codex [7], extended these capabilities to software engineering tasks, enabling natural language–to–code generation.

Despite their impressive performance, LLMs require substantial computational resources. Training large-scale models involves distributed GPU clusters and significant energy expenditure. Although inference is less demanding than training, it remains computationally intensive, particularly for real-time or interactive applications. GPUs are typically used to accelerate inference through parallel matrix operations; however, many deployment environments—such as edge devices, educational settings, or small-scale development infrastructures—rely primarily on CPUs.

As model sizes continue to grow, concerns regarding computational cost and environmental impact have intensified. For instance, GPT-3, with 175 billion parameters [5], illustrates the scaling trend that improves performance but also increases energy demand. Prior research has highlighted the environmental footprint associated with large-scale models, emphasizing the importance of energy-aware AI practices [2], [3]. Consequently, optimizing inference efficiency—particularly in CPU-based or resource-constrained settings—has become a critical research direction.

At the same time, the demand for automated code generation has increased significantly. LLM-based coding assistants can accelerate development by generating boilerplate code, suggesting completions, automating refactoring, and assisting debugging. Models trained on large code repositories, such as Codex [7], demonstrate that LLMs can produce syntactically valid and semantically meaningful programs from natural language prompts. This capability has practical implications for productivity and software quality.

However, deploying LLMs for code generation introduces unique constraints. Unlike natural language, source code must be syntactically precise and logically consistent; even minor token-level errors can invalidate the output. Furthermore, interactive development workflows demand low latency. In integrated development environments (IDEs), delays of several seconds can disrupt developer productivity, regardless of the underlying model's accuracy.

A. Evolution of Large Language Models

The progression from Recurrent Neural Networks (RNNs) to the Transformer architecture introduced by Vaswani et al. marked a turning point in language modeling. The Transformer's attention mechanism enabled parallelization and efficient scaling, paving the way for large models such as BERT [6] and GPT-3 [5].

Empirical scaling laws revealed that increasing model size often leads to emergent capabilities, including few-shot reasoning and improved generalization. However, this scaling also introduces substantial hardware requirements. Models with hundreds of billions of parameters require significant GPU memory (VRAM), effectively limiting local execution to specialized infrastructure.

In response, recent research has explored smaller yet efficient alternatives—often referred to as Small Language Models (SLMs)—such as Llama-3-8B or Mistral-7B. These models aim to preserve strong performance while reducing memory footprint and enabling deployment on consumer-grade hardware.

B. LLMs for Code Generation

Code generation represents a specialized application domain with distinct evaluation requirements. Benchmarks such as HumanEval and MBPP assess functional correctness of generated programs, typically through automated test suites. While these benchmarks provide standardized accuracy metrics, they often overlook computational cost and environmental considerations.

Moreover, existing literature largely focuses on benchmark accuracy without systematically analyzing inference efficiency across different hardware configurations. There is limited empirical work comparing execution time, energy consumption, and carbon footprint of LLM-based code generation in CPU-centric environments.

An additional gap concerns remote inference. Many proprietary models operate via cloud-based APIs, where computational resources are abstracted from the end user. In such scenarios, developers query remote systems from local CPU environments, but the energy cost and environmental impact of this interaction remain partially opaque. Understanding these trade-offs—between local execution and remote cloud-based inference—is essential for informed and sustainable deployment decisions.

These gaps motivate the need for comprehensive evaluations that integrate accuracy, latency, and sustainability metrics across both local and remote execution contexts. Such analyses are critical for determining the practical feasibility of LLM-assisted software development in resource-constrained and environmentally conscious settings.

C. Selected Models

The selection of models for these experiments provides a comprehensive comparison across different architectures and computational demands. The evaluation includes proprietary models from OpenAI (`gpt-4-turbo`, `gpt-3.5-turbo`, `gpt-4o-mini`) and open-source models from Hugging Face (`Meta-Llama-3-8B`, `WizardLM-2-8x22B`, `Qwen-2`).

1) Proprietary Models: These models are accessed via API and represent the current state-of-the-art in closed-source LLMs.

- GPT-4 Turbo: The immediate predecessor to GPT-4o, this model features a 128k context window and was trained with a cutoff date up to December 2023. It utilizes a Mixture-of-Experts (MoE) architecture to balance high performance with inference efficiency.
- GPT-3.5 Turbo: A highly optimized, efficient model that served as the industry standard for speed and cost-effectiveness prior to the release of the GPT-4o mini series. It is a dense model known for its reliability in lower-complexity tasks.

- GPT-4o mini: OpenAI's most cost-efficient small model, designed to replace GPT-3.5 Turbo. It supports multimodal inputs and offers reasoning capabilities superior to GPT-3.5, leveraging the architectural advancements of the GPT-4o family.

2) Open-Source Models: The selected open-weight models cover a range of sizes, from efficient 7B parameter models to large-scale Mixture-of-Experts systems.

- Meta-Llama-3-8B: The 8-billion parameter iteration of the Llama 3 family. It is a standard dense decoder-only transformer trained on a massive dataset of over 15 trillion tokens. It employs a tokenizer with a vocabulary size of 128k, significantly improving performance on multilingual tasks compared to Llama 2.
- WizardLM-2-8x22B: A large-scale Mixture-of-Experts (MoE) model based on the Mixtral 8x22B architecture. It is fine-tuned specifically for complex instruction following and reasoning. Despite having a total of 141B parameters, it only uses 39B active parameters per token, allowing for efficient inference relative to its total size.
- Qwen-2: A dense transformer architecture employing SwiGLU activation, Rotary Positional Embeddings (RoPE), and QKV bias. The model is noted for its strong performance in coding and mathematics benchmarks and its support for a context window of up to 32k tokens in standard configurations.

TABLE I: Summary of Selected Models

Model	Type	Arch.	Access
GPT-4 Turbo	Prop.	MoE	API
GPT-3.5 Turbo	Prop.	Dense (est.)	API
GPT-4o mini	Prop.	–	API
Llama-3-8B	Open	Dense (8B)	Local/HF
WizardLM-2-8x22B	Open	MoE (141B)	Local/HF
Qwen-2	Open	Dense	Local/HF

D. Justification of Model Selection and Theoretical Baseline

The selected models have been chosen to represent three different capacity regimes: large-scale foundation models, medium-sized open models, and compact lightweight models. The objective is to theoretically analyze how model dimension and training knowledge influence expected accuracy and computational efficiency.

From a theoretical standpoint, model size (number of parameters) correlates with representational capacity. Larger models typically capture more complex patterns and exhibit stronger reasoning and generalization capabilities due to broader training corpora. However, this increased capacity results in higher computational cost, increased inference latency, and greater energy consumption.

Conversely, smaller models are expected to be computationally efficient and suitable for edge or energy-constrained environments, though potentially at the expense of reduced reasoning accuracy in complex tasks.

This categorization defines a theoretical baseline that will later be validated through empirical experimentation.

a) Theoretical Hypotheses:

- H1: The large-scale model is expected to achieve the highest accuracy due to its larger parameter space and broader training knowledge.
- H2: The small-scale model is expected to provide the highest energy efficiency and lowest latency.
- H3: The medium-scale model is expected to offer the best trade-off between accuracy and efficiency.

These theoretical expectations define the reference framework for subsequent experimental validation, where accuracy, inference time, and energy consumption will be quantitatively measured.

III. Related work

The growing reliance on LLMs for tasks, such as code generation, text generation, and question answering has significantly advanced the field of artificial intelligence. However, these models, especially large-scale ones like GPT-3 [5], Codex [7], and BERT [6], have raised concerns regarding their environmental impact due to their substantial energy consumption and carbon footprint. As these models become larger, the need for energy-efficient deployment methods becomes critical, particularly when leveraging resources such as CPUs instead of GPUs, which are commonly used in research environments.

A. Energy Consumption and Sustainability in AI

The environmental impact of LLMs has been a topic of growing concern in recent research. Strubell et al. [2] highlighted the significant energy consumption required to train and run models like GPT-3, estimating that the carbon emissions of training such models can rival those of several cars over their lifetimes. This study emphasizes the need for developing models that are not only accurate but also energy-efficient, promoting the idea of Green AI. However, this research focuses primarily on the training phase and the larger-scale infrastructures typically used for training these models, rather than on their inference phase or CPU-based execution.

Schwartz et al. [3] further expanded on the concept of sustainable AI, advocating for a shift toward models that prioritize resource efficiency. They call for reducing the carbon footprint of deep learning models and propose that energy-efficient algorithms should be a focus in model design. However, their work lacks a focus on real-world inference scenarios, particularly in environments where GPUs are unavailable or impractical for deployment.

Xu et al. [8] provided a comprehensive survey on strategies to improve energy efficiency in deep learning models, addressing the growing need to reduce the environmental impact of AI systems. They reviewed a variety of techniques, including model compression, pruning, quantization, and efficient data usage, all aimed at optimizing the energy consumption of machine learning models. These methods can significantly

TABLE II: Theoretical Comparison of Selected Models

Category	Example Model	Params	Expected Accuracy	Latency	Energy Efficiency
Large-scale	GPT-4 class (>50B)	Very High	Very High	High	Low
Medium-scale	DeepSeek / LLaMA-13B	High	High	Medium	Medium
Small-scale	TinyLlama / 7B models	Moderate	Medium	Low	High

reduce the computational load during inference, particularly for large-scale models. While this work offers valuable insights into improving the energy efficiency of deep learning systems, it does not specifically focus on the trade-offs involved in running LLMs, such as GPT-3 or Codex, on CPUs for code generation tasks—an area central to our research.

B. CPU-Based Inference Optimization and Resource-Aware Deployment

Making LLM inference viable on CPU-centric systems has become an active area of research, particularly as organizations seek to reduce hardware dependency and environmental impact. Several optimization techniques have emerged to improve execution efficiency on commodity hardware.

One of the most widely adopted strategies is **quantization**, which reduces numerical precision from 16-bit floating point (FP16) to lower-bit representations such as 8-bit, 4-bit, or even 2-bit integers. This significantly decreases memory footprint and computational overhead, allowing larger models to run on systems with limited RAM. Frameworks such as `llama.cpp` have demonstrated that even 70B-parameter models can operate on machines with approximately 32GB of RAM through aggressive quantization techniques.

Another important approach is **distillation**, where smaller "student" models are trained to replicate the behavior of larger "teacher" models. Distillation reduces model size and inference cost while attempting to preserve task performance. This technique is particularly relevant in CPU-based environments, where memory and computation resources are constrained.

More recently, **speculative decoding** has been proposed as a latency-reduction strategy. In this method, a smaller model generates draft tokens that are subsequently verified or corrected by a larger model. This hybrid mechanism reduces average inference time while maintaining output quality. Although promising, the energy implications of speculative decoding across different hardware architectures remain underexplored.

While these optimization strategies improve feasibility on CPUs, there is limited literature systematically comparing their energy efficiency across x86 and ARM-based architectures under controlled experimental conditions. Most existing evaluations focus primarily on latency and memory reduction rather than carbon footprint or energy-per-token metrics.

Beyond algorithmic optimizations, inference efficiency is also closely linked to resource allocation strategies. Patterson et al. [9] examined the energy and carbon implications of large-scale deep learning systems, highlighting trade-offs between CPU and GPU utilization. Their work emphasized that hardware selection significantly influences environmental impact. However, the analysis was primarily centered on training workloads and did not explicitly address LLM inference or code generation tasks.

Subsequent work [4] proposed carbon-aware machine learning practices, including scheduling and hardware-aware deployment strategies for energy-efficient inference. While these studies provide important conceptual frameworks for sustainable AI, they do not specifically evaluate large language models in CPU-based settings or compare architectural differences such as x86 versus ARM processors.

In parallel, the integration of energy monitoring tools into DevOps pipelines has gained attention. Tools such as CodeCarbon [10] enable real-time estimation of energy consumption and CO_2 emissions during training and inference. When embedded into continuous integration workflows, these tools allow practitioners to track environmental impact alongside performance metrics. However, such monitoring is typically applied to smaller models or research experiments and is rarely systematically integrated into large-scale LLM inference pipelines, particularly in CPU-centric deployments.

Similarly, MLFlow has been explored for experiment tracking and resource management in production systems [11]. By logging performance metrics and managing model versions, MLFlow facilitates adaptive deployment strategies and cost optimization. Nevertheless, current literature does not sufficiently combine performance tracking with energy-aware metrics in the context of LLM-based code generation tasks.

Overall, while substantial progress has been made in inference optimization techniques and resource management frameworks, a comprehensive evaluation integrating CPU-based optimization, cross-architecture comparison (x86 vs. ARM), and energy monitoring remains largely absent. Addressing this gap is essential for developing sustainable and hardware-aware deployment strategies for LLM-driven code generation systems.

C. Code Generation with LLMs

The application of LLMs for code generation has gained significant attention, particularly with models such as Codex [7], which are specifically designed to generate programming code from natural language prompts. Codex has shown great potential in automating code completion, bug fixing, and refactoring tasks, but there is a lack of research on how these models perform when executed on CPU-based systems as opposed to GPUs.

A recent study by Arora et al. [12] introduced SetupBench, a benchmark designed to evaluate the ability of LLM agents to bootstrap development environments autonomously. The benchmark comprises 93 tasks spanning various programming

languages, database engines, and multi-service orchestration scenarios. The evaluation of OpenHands, a state-of-the-art coding agent, revealed low success rates across task categories, particularly in repository setup and local database configuration. The study identified substantial inefficiencies in agent exploration strategies, with a significant percentage of actions being unnecessary compared to optimal human behavior. These findings highlight gaps in current agents' practical environment-bootstrap capabilities.

However, the research does not investigate the inference efficiency of these models when deployed in resource-constrained environments or on CPUs, which is a critical gap in the current body of literature. Furthermore, their focus was mainly on cloud-based models and did not consider the potential environmental impact of running these models in cloud infrastructures, where energy consumption and carbon footprint can vary significantly depending on the hardware used.

D. Gap in Literature

While the literature provides a strong foundation for understanding the energy consumption and performance of deep learning models, particularly in large-scale environments using GPUs, there is limited research specifically addressing the trade-offs and performance of LLMs for code generation when executed in CPU-only environments. Most of the existing studies focus on training and GPU-based inference, overlooking the operational efficiency and sustainability of running LLMs in low-resource environments where only CPUs are available.

Energy-efficient deployment strategies using tools like MLFlow and CodeCarbon remain underexplored for LLMs, particularly in real-time inference tasks like code generation. This paper addresses this gap by comparing the CPU-based performance and energy efficiency of various LLMs, focusing on code generation and incorporating energy monitoring through MLFlow and CodeCarbon to evaluate inference performance and environmental impact in resource-constrained environments.

IV. Experimental Setup

This section describes the experimental setup for evaluating the performance, energy efficiency, and environmental impact of various LLMs in code generation tasks. The selected models include proprietary LLMs accessed via remote APIs as well as open-weight models available through Hugging Face repositories. Performance monitoring is conducted using MLFlow, while energy consumption and estimated CO_2 emissions are tracked using CodeCarbon to ensure reproducible and standardized measurement of sustainability metrics.

Given that the models are accessed remotely through API calls, the evaluation focuses on the interaction between the client-side hardware platform (Intel x86 vs. Apple ARM architectures) and the observed latency, local energy usage, and output correctness. The goal is to understand whether differences in hardware architecture influence inference behavior and sustainability when invoking cloud-hosted LLMs.

A. Research Questions

To structure the experimental analysis, we formulate the following Research Questions (RQs):

- **RQ1 (Performance):** How does inference latency for remote API-based LLM calls vary when executed from different client systems (Intel x86 vs. Apple ARM architectures)?
- **RQ2 (Sustainability):** What is the local energy consumption and estimated CO_2 footprint associated with remote inference requests when initiated from x86 versus ARM-based systems?
- **RQ3 (Accuracy):** Does model accuracy remain consistent across remote API calls when accessed from different hardware platforms, or are there observable differences in output correctness due to system-level factors?

B. Hardware and Operating System

The experimental evaluation has been conducted on two alternative computing platforms in order to analyze potential variations in performance and energy behavior across different hardware architectures and operating systems.

- System 1 (Legacy x86):
  - **CPU:** Intel Core i5-1135G7 (4 cores, 8 threads, up to 4.2 GHz).
  - **RAM:** 16 GB DDR4.
  - **OS:** Windows 11 Pro.
  - **Context:** Represents a resource-constrained environment common in general office settings.
- System 2 (Modern ARM):
  - **CPU:** Apple M3 Pro (12-core CPU, 18-core GPU - GPU not used for main inference logic).
  - **RAM:** 36 GB Unified Memory.
  - **OS:** macOS Sonoma.
  - **Context:** Represents a high-performance portable workstation capable of loading larger models into memory.

The inclusion of both a resource-constrained x86-based architecture (Intel i5, Windows) and a high-performance ARM-based architecture (Apple M3 Max, macOS) enables a comprehensive comparative analysis of inference latency and energy efficiency across heterogeneous computing environments. This distinction is particularly relevant for addressing the "Hardware Democratization Challenge," as it allows for an empirical evaluation of the "CPU inference gap" by comparing ubiquitous office hardware against modern systems featuring unified memory architectures. By utilizing this multi-tier setup, the study transitions from a Phase 1 remote API-based baseline to Phase 2 local execution, facilitating a deeper assessment of how specific architectural designs influence both the performance-per-watt efficiency and the environmental sustainability of large language model workloads.

From a Green AI perspective, evaluating the same models under different processor architectures allows us to examine the trade-off between accuracy and energy consumption in realistic deployment scenarios. This comparison contributes to understanding whether architectural differences (CISC vs.

ARM-based designs) significantly impact performance-per-watt efficiency during inference.

C. LLMs Selected for the Experiments

The models shown in Table III will be evaluated for code generation in C programming tasks from OpenAI and HuggingFace (via Novita as inference provider):

These models represent a range of architectures, including large-scale models like gpt-4-turbo and optimized models like gpt-4o-mini.

D. Performance Monitoring and Energy Consumption Measurement

To evaluate the energy consumption and carbon emissions, the following tools will be employed:

- MLFlow will be integrated to monitor and track the inference time, accuracy, and computational cost of each model.
- CodeCarbon will be used to track CO_2 emissions and energy consumption for each inference task, helping assess the environmental impact of running LLMs.

E. Inference Tasks for the LLMs

The models will generate C source code for a set of problems, which are as follows:

- 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.*
- Finding the 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: *Prompt: Write a C program that returns 1 if the input number is even, and 0 if it is odd. The input number for testing will be 122.*
- Absolute Difference Calculation: *Prompt: Write a C program that calculates the absolute difference between two numbers. The input numbers are -122 and 11.*
- Sum of Digits: *Prompt: Write a C program that calculates the sum of the digits of the input number. The input number is 123.*

Each task was repeated 10 times for each model to ensure that the results are statistically significant and to account for potential variations in model performance across multiple runs. The generated code will be validated using gcc to ensure correctness, and the inference time, accuracy, energy consumption, and computational cost will be tracked.

F. Evaluation Framework and Experimental Design

This study evaluates the performance, sustainability, and practical feasibility of large language models (LLMs) for code generation tasks under remote inference conditions. The experimental design is structured around clearly defined evaluation criteria, measurement tools, and phased experimentation to ensure methodological consistency and reproducibility. Figure 1 summarizes the overall evaluation workflow, from prompt execution to metric collection and cross-platform comparison.

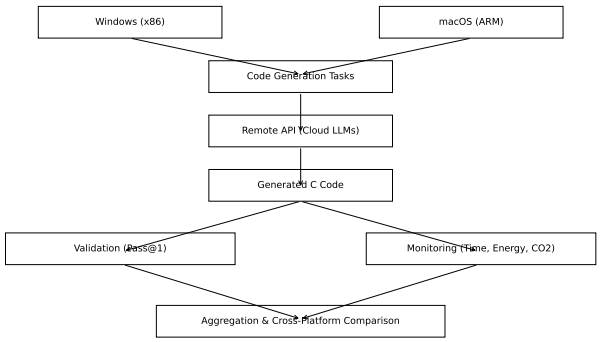

Figure 1: Evaluation pipeline for remote API-based inference from two client systems (Windows x86 and Apple Silicon ARM).

1) Evaluation Criteria: The assessment focuses on four main dimensions:

- **Inference Time:** The total time required by the model to generate the requested C code.
- **Code Generation Accuracy:** The functional correctness of the generated code, verified through compilation and execution against predefined unit tests.
- **Energy Consumption:** The client-side energy usage associated with performing remote inference requests.
- **Computational Cost:** The monetary cost of executing inference through remote APIs (e.g., GPT-4o and related cloud-based models).

2) Experimental Phases: The experimental workflow is structured in two phases:

- **Phase 1 – Remote API-Based Inference:** The primary phase of this study evaluates models accessed through remote APIs. Inference latency, accuracy, energy consumption, and cost are measured when invoking the models from two different hardware platforms (Intel x86 and Apple ARM architectures).
- **Phase 2 – Local CPU Deployment:** As a continuation of this research, future experiments will involve executing open-weight LLMs locally on CPU-based systems. This phase will enable direct comparison between remote and local execution, assessing trade-offs in latency, energy efficiency, and operational cost.

The current study focuses on Phase 1, establishing a baseline for remote inference analysis before extending the evaluation to local CPU deployment scenarios and it is the extension of a previous work presented in [1].

3) Metrics and Measurement Tools: A structured set of quantitative metrics is employed to ensure rigorous evaluation.

a) Performance Metrics:

- **Inference Time (T_{inf}):** The total wall-clock time measured from sending the prompt to receiving the final generated token.
- **Throughput (TPS):** Tokens per second, calculated as:

TABLE III: Selected LLMs for the Experiments

From OpenAI	From Hugging Face
gpt-4-turbo	meta-llama/Meta-Llama-3-8B-Instruct
gpt-3.5-turbo	alpindale/WizardLM-2-8x22B
gpt-4o-mini	Qwen/Qwen3-235B-A22B-Instruct-2507

$$TPS = \frac{N_{tokens}}{T_{inf}} \tag{1}$$

b) Energy Metrics: Energy consumption is measured using the **CodeCarbon** Python package. On Intel systems, measurements rely on the RAPL (Running Average Power Limit) interface, while on macOS systems the `powermetrics` tool is used.

Total energy consumption is computed as:

$$E_{total} = \int_{t_{start}}^{t_{end}} P(t)\, dt \tag{2}$$

where $P(t)$ represents instantaneous power draw in Watts during inference.

c) Correctness Metric: Functional correctness is evaluated using the **Pass@1** metric, validated against a suite of predefined unit tests for each C programming task:

$$\text{Pass@1} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}(s_i \text{ passes all tests}) \tag{3}$$

where k is the number of evaluated samples and $\mathbf{1}(\cdot)$ is the indicator function.

By combining performance, energy, cost, and correctness metrics within a phased experimental design, this framework enables a comprehensive analysis of the trade-offs involved in deploying LLMs for code generation tasks in resource-constrained environments.

V. Results

The experimental results are presented considering two different hardware platforms: a Windows-based x86 system and an Apple Silicon (M3 Pro) system. Since hardware architecture may significantly influence latency and energy behavior, the evaluation is structured in three stages: (i) results on the Windows system, (ii) results on the macOS system, and (iii) a cross-platform comparative analysis.

The evaluation focuses on three key dimensions: Performance and Throughput, Code Generation Accuracy, and Energy Efficiency.

The analyzed models include proprietary and open-weight large language models obtained from OpenAI and Hugging Face, with parameter sizes ranging from 3.8B to 235B parameters. All models were executed under the same experimental protocol on both systems.

A. Results on Windows (x86 Architecture)

This subsection presents the performance metrics obtained on the Windows 11 Pro platform equipped with an Intel i5-1135G7 processor. The models are first compared within this controlled environment to analyze differences in execution time, throughput, accuracy, energy consumption, and estimated CO_2 emissions.

The objective is to evaluate how model scale influences the accuracy–efficiency trade-off under an x86-based architecture.

1) Performance & Throughput: The execution time of each model varied significantly, primarily due to the differences in model size and complexity. GPT-4-turbo, the largest models in the study, had execution times of 3.5 minutes. Despite optimizations in GPT-4-turbo, it required more time to complete the task, suggesting trade-offs between speed and accuracy.

In contrast, GPT-3.5-turbo was the fastest, taking only 1.9 minutes to generate code. However, this speed came at the cost of accuracy, as shown in the next section. The smaller model, GPT-4o-mini, took 2.8 minutes, still significant for its reduced size.

The smaller models from Hugging Face, including Meta-Llama-3-8B, took between 3.9 and 4.7 minutes for the task, which is relatively long compared to their smaller size. Finally, the largest models like WizardLM-2-8x22B and Qwen/Qwen3-235B took 7.8 minutes and 1.2 hours, respectively, showing the strong correlation between model size and execution time.

2) Code Generation Accuracy: The accuracy of the models in generating correct C code varied significantly, with the larger models performing better in generating valid code. GPT-4-turbo achieved the highest accuracy of 36% demonstrating its effectiveness in generating correct code for the given tasks..

GPT-3.5-turbo, with its smaller size, performed poorly, with an accuracy of 12%, reflecting the limitations of smaller models for such complex tasks. The smaller models, such as Meta-Llama-3-8B had accuracy rates of 6% and 18%, respectively, indicating that smaller parameter models struggle with generating accurate code. Larger models like WizardLM-2-8x22B and Qwen/Qwen3-235B both showed 17% accuracy, suggesting that despite their massive size, they also faced challenges in code generation.

3) Energy Efficiency: The energy consumption per inference varied based on model size, with larger models generally consuming more energy. GPT-4-turbo consumed between 0.1–0.5 kWh, which is typical for models of their size and complexity. Interestingly, GPT-4o-mini, despite being smaller, consumed 0.003 kWh, likely due to specific optimizations and the inherent inefficiency of smaller models for complex tasks.

In contrast, smaller models like Meta-Llama-3-8B consumed 0.0015 kWh and 0.0028 kWh, respectively, indicating their efficiency relative to their size. However, larger models like WizardLM-2-8x22B and Qwen/Qwen3-235B consumed significantly more energy, with values of 0.0047 kWh and 0.0071 kWh, respectively, consistent with their massive size and computational demands.

CO2 emissions follow the same trend as energy consumption. GPT-4-turbo emitted 0.000337222 kg while GPT-3.5-turbo produced 0.00044873 kg, showing the inefficiency of smaller models in terms of their carbon footprint. Meta-Llama-3-8B had lower CO2 emissions of 0.000270392 kg and 0.000498069 kg, respectively, reflecting their lower energy consumption.

The largest models had the highest emissions: WizardLM-2-8x22B generated 0.001331077 kg and Qwen/Qwen3-235B generated 0.005493181 kg.

B. Results on macOS (Apple Silicon Architecture)

This subsection reports the experimental results obtained on the Apple M3 Pro platform running macOS. As ARM-based processors are designed with different performance-per-watt characteristics, this analysis allows us to assess whether architectural efficiency improves inference latency and reduces energy consumption while maintaining comparable accuracy levels.

The same evaluation metrics used in the Windows experiments are applied here to ensure methodological consistency.

1) Performance & Throughput (Apple M3 Pro): The execution time measured on the Apple M3 Pro platform shows a different performance distribution compared to the x86 system, highlighting the influence of ARM-based architecture on inference behavior.

GPT-3.5-turbo was the fastest model, completing the task in 1.24 minutes. GPT-4o-mini followed with 1.64 minutes, while GPT-4-turbo required 2.66 minutes. Among the open-weight models, Meta-Llama-3-8B completed the task in 2.31 minutes, WizardLM-2-8x22B in 2.8 minutes, and the largest evaluated model, Qwen/Qwen3-235B-A22B-Instruct-2507, required 4.27 minutes.

Although larger parameter models generally exhibited longer execution times, the differences were less extreme than typically expected, suggesting that the Apple Silicon architecture may mitigate some latency overhead through improved performance-per-watt efficiency.

2) Code Generation Accuracy (Apple M3 Pro): The accuracy results on the Apple platform reveal notable variations between proprietary and open-weight models.

GPT-4o-mini achieved the highest accuracy, reaching 100%, followed by GPT-4-turbo and Meta-Llama-3-8B, both achieving 90%. WizardLM-2-8x22B reached 80% accuracy, demonstrating strong performance despite its larger size.

In contrast, GPT-3.5-turbo and Qwen/Qwen3-235B-A22B-Instruct-2507 both obtained 0% accuracy in the evaluated task, indicating that model size alone does not guarantee correctness and that architecture specialization and training quality play a critical role.

These results suggest that mid-scale and optimized proprietary models may provide superior reliability in code generation tasks under the tested conditions.

3) Energy Efficiency (Apple M3 Pro): Energy consumption per inference on the Apple M3 Pro platform remained relatively low across all models, reflecting the efficiency of the ARM-based architecture.

GPT-4-turbo consumed 0.000497 kWh, while GPT-3.5-turbo required 0.000634 kWh despite its shorter execution time. GPT-4o-mini consumed 0.000843 kWh, slightly higher due to its full task completion and higher computational workload.

Among open-weight models, Meta-Llama-3-8B was the most energy-efficient, with 0.000109 kWh consumption. WizardLM-2-8x22B consumed 0.000984 kWh, and Qwen/Qwen3-235B-A22B-Instruct-2507 required the highest energy consumption at 0.001467 kWh.

CO_2 emissions followed the same trend as energy consumption. GPT-4-turbo emitted 8.64×10^{-5} kg of CO_2, while GPT-3.5-turbo emitted 1.10×10^{-4} kg. Meta-Llama-3-8B showed relatively low emissions (1.09×10^{-4} kg), whereas WizardLM-2-8x22B and Qwen/Qwen3-235B-A22B-Instruct-2507 generated 1.71×10^{-4} kg and 2.55×10^{-4} kg, respectively.

Overall, the Apple Silicon platform demonstrates strong performance-per-watt characteristics, particularly benefiting medium-scale and optimized models.

VI. Discussion and Evaluation

The experimental evaluation conducted on both the Windows (x86) and Apple Silicon (M3 Pro, ARM) platforms provides insights into the interaction between model scale, hardware architecture, accuracy, and energy efficiency in code generation tasks.

Rather than presenting isolated per-system results, this section focuses on cross-platform comparisons to highlight architectural influences on performance and sustainability metrics.

A. Performance and Throughput

Figure 2 shows the execution time for all evaluated models on both hardware platforms. A clear relationship between model size and inference latency can be observed. Larger models generally require more execution time; however, the Apple Silicon platform demonstrates consistently lower latency across most models.

This indicates that ARM-based architecture provides improved performance-per-watt characteristics, particularly benefiting medium and large-scale models. The performance gap becomes more noticeable as model size increases, suggesting that hardware efficiency plays a critical role in scaling behavior.

B. Code Generation Accuracy

As illustrated in Figure 3, in general accuracy on the Apple M3 hardware is better but accuracy remains largely consistent across both platforms. This suggests that code generation correctness is primarily model-dependent.

Optimized proprietary models outperform several large open-weight models, indicating that parameter count alone does

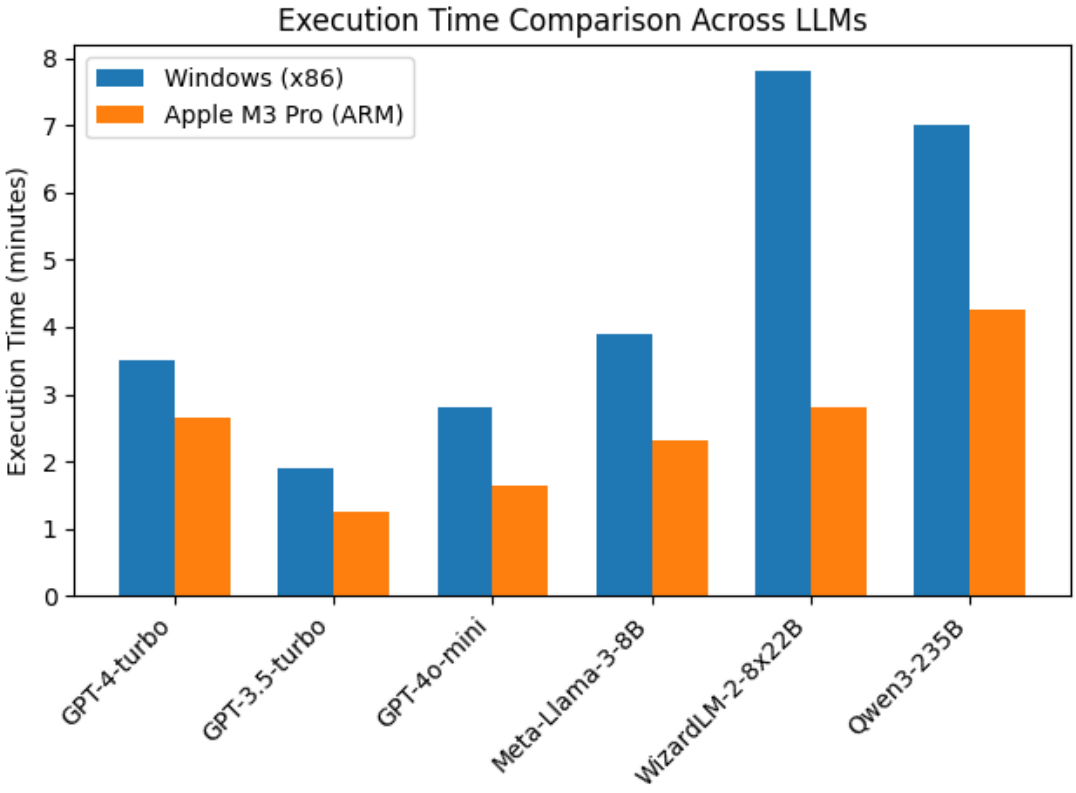

Figure 2: Execution Time Comparison Across LLMs (Windows vs Apple Silicon).

not guarantee higher accuracy. The results reinforce that model architecture and training quality are dominant factors in correctness performance.

C. Energy Efficiency and Environmental Impact

Figures 4 and 5 show that energy consumption and carbon emissions are strongly influenced by both model size and hardware architecture.

While larger models generally consume more energy per inference, the Apple Silicon platform consistently exhibits lower consumption compared to the Windows x86 system. This confirms that processor architecture significantly impacts sustainability metrics.

Interestingly, certain mid-scale models provide favorable accuracy–energy trade-offs, suggesting that optimal deployment decisions should consider both model dimension and hardware platform.

D. Overall Trade-Off Analysis

The cross-platform comparison reveals three key observations:

- Model size strongly affects execution time and energy consumption.
- Hardware architecture significantly influences latency and energy efficiency.
- Accuracy remains largely model-dependent and relatively stable across systems.

From a Green AI perspective, the results demonstrate that combining moderately sized models with energy-efficient hardware architectures can substantially reduce environmental impact without sacrificing accuracy.

Therefore, sustainable AI deployment requires joint consideration of model selection and hardware platform rather than focusing solely on parameter scale.

This study provides a comprehensive evaluation of various LLMs for code generation tasks across two different hardware platforms: a Windows-based x86 system and an Apple Silicon (M3 Pro) ARM-based system. The analysis considered performance, accuracy, energy efficiency, and environmental impact, enabling a cross-platform assessment of the accuracy–efficiency trade-off.

The results reveal several key insights:

1) Model size significantly influences execution time and energy consumption on both systems. Larger models generally require longer inference time and exhibit higher energy consumption, leading to increased CO_2 emissions. However, the magnitude of this impact depends on the underlying hardware architecture. The Apple Silicon platform demonstrated improved performance-per-watt characteristics, reducing energy consumption while maintaining comparable accuracy levels.
2) In general, accuracy on the Apple M3 hardware is better, although on both platforms, performance remains heavily dependent on the specific model used. Optimized proprietary

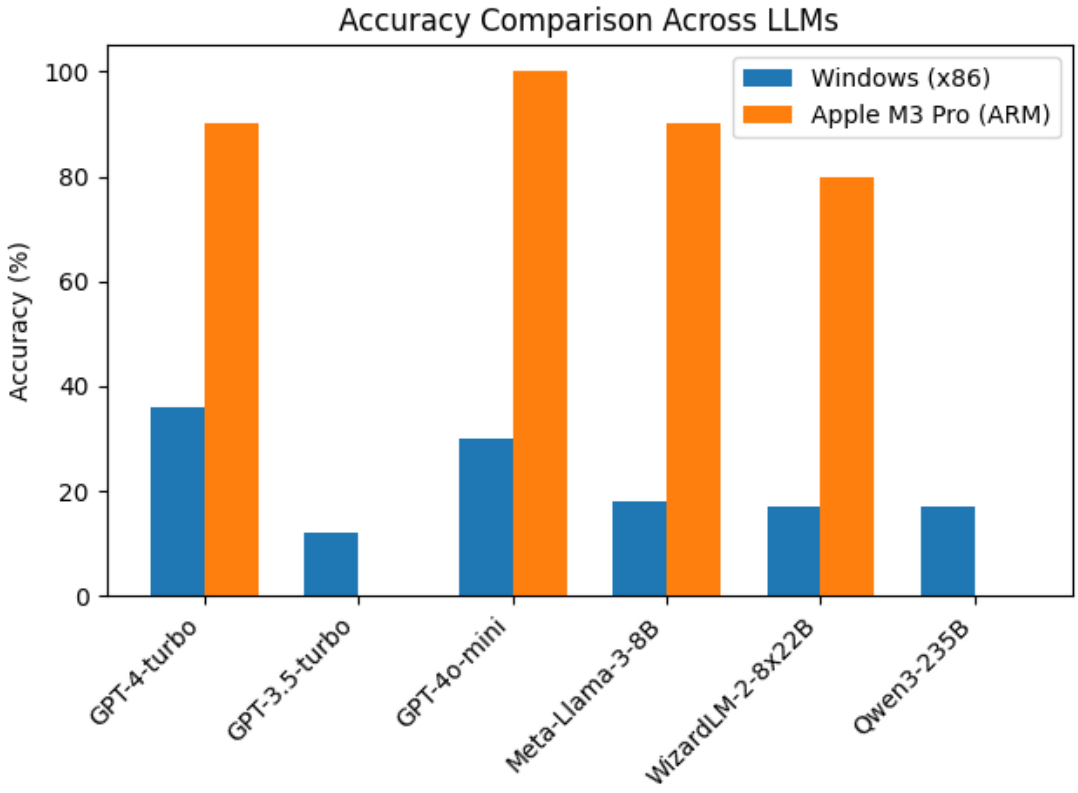

Figure 3: Code Generation Accuracy Comparison Across Platforms.

models consistently achieved higher correctness rates across both platforms, whereas some large open-weight models did not demonstrate proportional gains in accuracy despite their scale .

3) Mid-scale or optimized models (e.g., compact frontier variants) often provide the most favorable trade-off between accuracy and energy consumption. These models benefit particularly from energy-efficient hardware architectures, making them strong candidates for sustainable deployment scenarios.

Overall, the findings indicate that the performance of LLMs in code generation tasks is determined by the interaction between model architecture and hardware platform. While larger models remain preferable when maximum accuracy is required, their environmental footprint must be carefully considered. Conversely, combining moderately sized models with energy-efficient processors can substantially reduce carbon impact without severely compromising performance.

E. Threats to Validity

- **Prompt Sensitivity:** Smaller models are notoriously more sensitive to prompt phrasing. We used a standardized prompt, which may favor larger models with better instruction-following robustness.
- **Measurement Overhead:** CodeCarbon introduces a slight overhead, though negligible compared to the model inference load.

VII. Conclusions and Future Work

This study presented a comprehensive cross-platform evaluation of several large language models (LLMs) for code generation tasks, analyzing their performance on two distinct hardware architectures: a Windows-based x86 system and an Apple Silicon (M3 Pro) ARM-based system. The evaluation considered execution time, code generation accuracy, energy consumption, and CO_2 emissions in order to understand the multi-dimensional trade-offs between model scale, computational efficiency, and environmental impact.

The dual-system experimentation provides a more realistic perspective on deployment scenarios, since model performance is not only dependent on parameter size and training data, but also on the underlying hardware architecture and its performance-per-watt characteristics.

The results reveal several key insights:

1) **Impact of Model Scale on Accuracy and Resource Consumption.** Larger proprietary models, such as GPT-4o and GPT-4-turbo, consistently demonstrated superior code generation accuracy across both platforms. Their advanced architectures and extensive training corpora enable improved reasoning and syntactic correctness in complex C programming tasks. However, this performance advantage comes at the cost of increased computational demand. On both systems, larger models required longer execution times and exhibited higher energy consumption and CO_2 emissions.

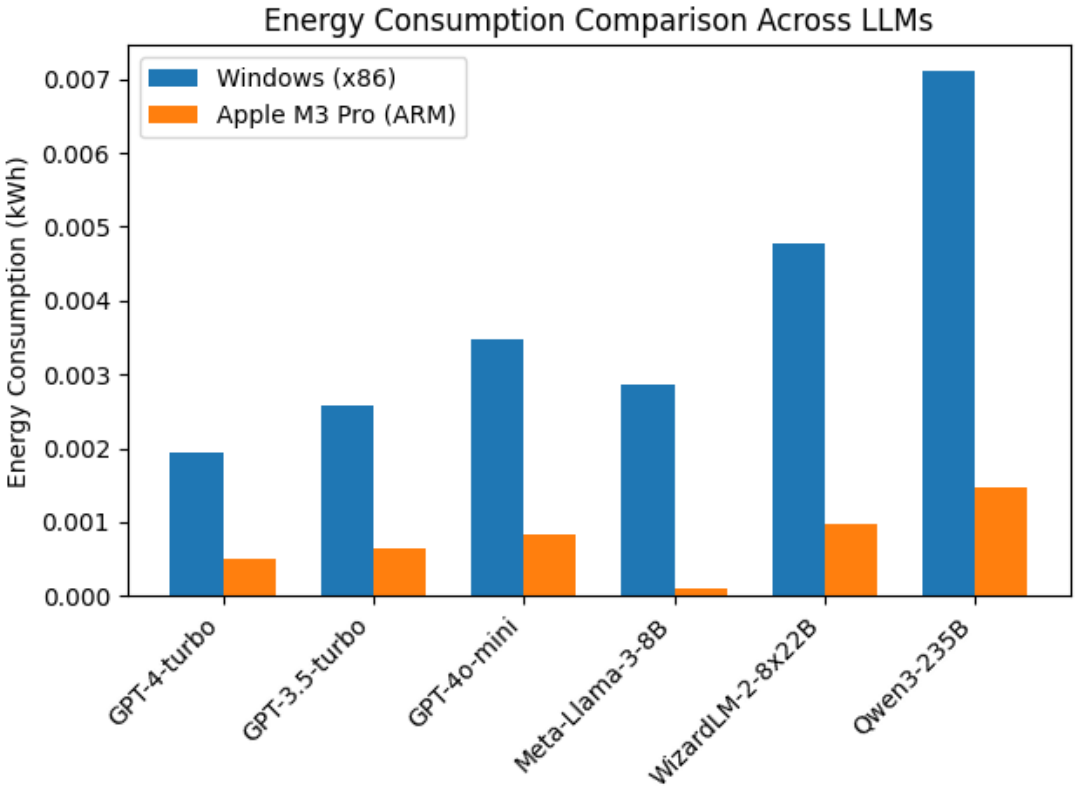

Figure 4: Energy Consumption per Inference Across Platforms.

Although the Apple Silicon platform mitigated some of these overheads, the environmental footprint of large-scale models remains significant, particularly when deployed in high-frequency or large-scale inference scenarios.

2) **Role of Hardware Architecture in Efficiency.** The comparison between Windows (x86) and Apple Silicon (ARM) platforms highlights the substantial influence of processor architecture on performance-per-watt efficiency. While accuracy remained largely stable across systems—confirming that correctness is primarily model-dependent— execution time and energy consumption were consistently lower on the Apple Silicon platform. This suggests that ARM-based architectures offer improved energy efficiency for inference workloads, making them particularly suitable for sustainable AI deployment strategies.
3) **Trade-Offs in Medium and Smaller Models.** Medium-scale and smaller open-weight models, such as Meta-Llama-3-8B, demonstrated improved energy efficiency and lower carbon emissions, especially when executed on the ARM-based platform. However, their accuracy varied depending on task complexity and optimization level. These models may represent viable alternatives in scenarios where moderate performance is acceptable and sustainability is a priority. In particular, optimized intermediate models (e.g., GPT-4o-mini) appear to offer a balanced compromise between accuracy and energy consumption, especially when combined with energy-efficient hardware.
4) **Environmental Implications for Sustainable AI.** The joint evaluation of energy consumption and CO_2 emissions reinforces the importance of considering environmental impact in AI system design. Larger parameter models substantially increase per-inference carbon footprint, particularly under less energy-efficient hardware configurations. Therefore, sustainable deployment decisions must consider not only model size but also the interaction between software architecture and hardware platform.

Overall, the results demonstrate that LLM deployment decisions should not be based solely on accuracy benchmarks. Instead, a holistic approach is required, balancing correctness, latency, computational cost, and environmental sustainability. The findings emphasize that combining appropriately sized models with energy-efficient hardware can significantly reduce environmental impact while maintaining acceptable performance levels.

Across the diverse set of evaluation tasks—including prime number checks, finding the greatest of three integers, and absolute difference calculations—the results indicate that both platforms behaved similarly, exhibiting consistent performance trends across the entire experimental set. While the Apple M3 Pro system demonstrated superior efficiency and reduced latency due to its ARM-based architecture, the relative behavior of the models remained stable between the two environments. Specifically, execution time and energy consumption scaled consistently with model complexity on both systems. Con-

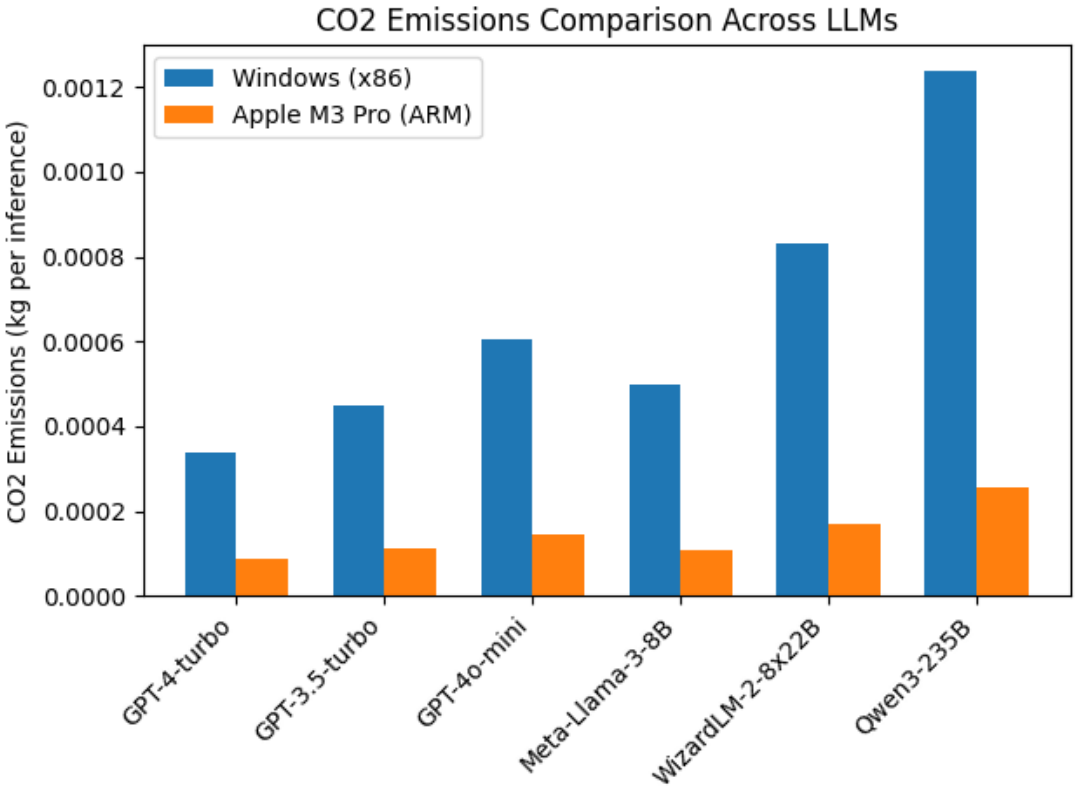

Figure 5: CO_2 Emissions per Inference Across Platforms.

sequently, no single task caused a significant divergence in how the platforms managed the workloads, supporting the conclusion that both systems respond uniformly to the varying demands of the selected C programming problems.

Additionally, it is important to recognize that the energy measurements reported in the remote inference phase may not fully reflect the actual computational energy consumed by the models themselves. When utilizing cloud-based APIs, measurements taken on the local laptop primarily capture the hardware's idle state and the overhead of maintaining a network connection while awaiting a response, rather than the raw energy required for model processing. To provide a more accurate assessment of the model's environmental footprint, a definitive energy analysis should be centered on Phase 2, which involves local execution and direct monitoring of on-device resource usage.

Furthermore, the comparison between the two test machines is constrained by numerous confounding variables, including significant differences in CPU cores, RAM capacity, operating systems, and the underlying measurement tools used (RAPL for Intel versus powermetrics for macOS). These systemic variations make it challenging to isolate and attribute performance differences specifically to the ARM versus x86 architectures. Acknowledging these limitations is crucial for a transparent interpretation of the architectural efficiency and the sustainable deployment of LLMs.

Future Work

Building upon these findings, several research directions are planned to extend and deepen the current analysis:

- **Local Model Deployment and Hardware Acceleration.** The next step of this research will involve executing the evaluated models locally rather than exclusively through cloud-based APIs. This will include deploying open-weight models on local CPUs and GPUs to analyze inference latency, energy consumption, and CO_2 emissions under controlled on-premise environments. By comparing local CPU and GPU execution against API-based inference, it will be possible to quantify the trade-offs between cloud efficiency and local resource utilization.
- **Cross-Architecture Local Comparison.** The local experiments will be conducted on both x86 and ARM-based systems to determine how architectural characteristics influence performance and sustainability metrics in fully local execution scenarios. This will allow the identification of optimal hardware–model pairings for resource-constrained environments.
- **Extended Task Evaluation.** While the present study focused on C code generation, future work will evaluate additional software engineering tasks such as debugging, code optimization, automated refactoring, and test generation. This broader evaluation will help determine whether the observed accuracy–efficiency trade-offs generalize across

diverse development workflows.

- **Inference Optimization Techniques.** Further research will investigate model optimization strategies including quantization, pruning, distillation, and low-rank adaptation. These techniques may reduce computational demand and energy consumption while preserving acceptable accuracy levels.
- **Composite Efficiency Metrics.** Future analysis will incorporate normalized indicators such as accuracy-per-kWh and performance-per-watt metrics to provide a unified sustainability score. This will support more systematic evaluation of Green AI strategies in real-world deployment contexts.

By integrating local execution experiments and advanced optimization techniques, the ongoing research aims to develop a comprehensive framework for sustainable AI deployment. This framework will support evidence-based decision-making for selecting models and hardware configurations that balance performance, cost, and environmental impact in practical software engineering applications.

ACKNOWLEDGEMENTS

The authors are part of the Software and Systems Engineering research group of Mondragon Unibertsitatea (IT1519-22), supported by the Department of Education, Universities and Research of the Basque Country. This research was supported by the Ikerketa Taldeak funding (IT1519-22) and the GRECO Elkartek project (KK-2024/00090), both funded by Eusko Jaurlaritza.

REFERENCES

[1] M. Illarramendi, J. A. Agirre, A. Picatoste, and J. I. Igartua, "Brains without brawn: Evaluating CPU performance for code generation with large language models", in *GREEN 2025, The Tenth International Conference on Green Communications, Computing and Technologies*, 2026, pp. 8–15, ISBN: 978-1-68558-311-8.

[2] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in NLP*, (visited on 09/03/2025), 2019. arXiv: 1906.02243 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1906.02243 (visited on 09/03/2025).

[3] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, *Green AI*, (visited on 09/03/2025), 2019. arXiv: 1907.10597 [cs.CY]. [Online]. Available: https://arxiv.org/abs/1907.10597 (visited on 09/03/2025).

[4] D. Patterson *et al.*, *The carbon footprint of machine learning training will plateau, then shrink*, (visited on 09/03/2025), 2022. arXiv: 2204.05149 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2204.05149 (visited on 09/03/2025).

[5] T. B. Brown *et al.*, *Language models are few-shot learners*, (visited on 09/03/2025), 2020. arXiv: 2005.14165 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.14165 (visited on 09/03/2025).

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, (visited on 09/03/2025), 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805 (visited on 09/03/2025).

[7] M. Chen *et al.*, *Evaluating large language models trained on code*, (visited on 09/03/2025), 2021. arXiv: 2107.03374 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2107.03374 (visited on 09/03/2025).

[8] J. Xu, W. Zhou, Z. Fu, H. Zhou, and L. Li, *A survey on green deep learning*, (visited on 09/03/2025), 2021. arXiv: 2111.05193 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.05193 (visited on 09/03/2025).

[9] D. Patterson et al., *Carbon emissions and large neural network training*, (visited on 09/03/2025), 2021. arXiv: 2104.10350 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2104.10350 (visited on 09/03/2025).

[10] MLCO2, *Codecarbon*, (visited on 09/03/2025), 2025. [Online]. Available: https://github.com/mlco2/codecarbon (visited on 09/03/2025).

[11] Y. Rangineeni and J. Pub, "End-to-end mlops: Automating model training, deployment, and monitoring", *Journal of Recent Trends in Computer Science and Engineering (JRTCSE)*, vol. 7, pp. 60–76, Sep. 2019.

[12] A. Arora, J. Jang, and R. Zilouchian Moghaddam, *Setupbench: Assessing software engineering agents' ability to bootstrap development environments*, (visited on 09/03/2025), 2025. arXiv: 2507.09063 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2507.09063 (visited on 09/03/2025).