International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Evaluating Concurrency Impacts on Open AI Language Models

Shreyam Dutta Gupta *

Palo Alto, California.

International Journal of Science and Research Archive, 2025, 14(03), 378-387

Publication history: Received on 26 January 2025; revised on 04 March 2025; accepted on 06 March 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.14.3.0647

Abstract

While the OpenAI API documentation presents a range of theoretical guidelines and optimization techniques for reducing latency and improving performance in language model applications, it largely focuses on high-level principles rather than providing quantitative, comparative data under realistic load conditions. In this paper, we offer an empirical evaluation of four OpenAI language models—o1-mini, o1-preview, GPT-4o, and GPT-4o-mini; across diverse task categories including explanatory, creative, technical, translation, and coding prompts. By employing asynchronous load testing with varying concurrency levels, we measure key performance metrics such as average response time, throughput, and token efficiency. Our study not only validates the optimization principles discussed in the API documentation but also provides actionable insights and a data-driven framework for model selection in real-world scenarios. This comparative analysis enables practitioners to make informed decisions based on measured performance trade-offs, thereby complementing and extending the theoretical recommendations in the OpenAI guidelines.

Keywords: Generative AI; Language Models; Performance Evaluation; Latency Optimization; Token Efficiency

1. Introduction

The rapid evolution of language models has led to a proliferation of architectures and configurations, each tailored to specific use cases and application domains. While it is widely acknowledged that different models are optimized for distinct tasks—ranging from generating creative narratives to providing precise technical explanations; the selection process often relies on qualitative assessments and theoretical guidelines. The OpenAI API documentation, for instance, offers a set of principles for latency optimization and performance improvement. However, these guidelines primarily focus on high-level recommendations without providing a comparative, quantitative analysis of model performance under realistic load conditions.

This paper presents a systematic empirical evaluation of four OpenAI language models: o1-mini, o1-preview, GPT-4o, and GPT-4o-mini—across a diverse set of prompt categories including explanatory, creative, technical, translation, and coding tasks. Our approach, characterized as "blanket testing," involves the application of asynchronous load testing with varying concurrency levels to generate measurable performance metrics such as average response time, throughput, and token efficiency.

By integrating real-world load conditions into our evaluation framework, we aim to complement the theoretical optimization strategies outlined in the API documentation [1]. The intent is not to claim a universal superiority of one model over another, but rather to provide a robust, data-driven basis for informed decision-making. This enables practitioners to understand the trade-offs between latency, scalability, and resource utilization, thereby guiding the selection of the most appropriate model for their specific operational requirements.

* Corresponding author: Shreyam Dutta Gupta

Copyright © 2025 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution Liscense 4.0.

In summary, while each OpenAI model is designed with a particular set of objectives, our research offers a comparative analysis that bridges the gap between theoretical recommendations and practical performance, supporting a more nuanced and empirical approach to model selection.

2. literature review

The evaluation of language models under real-world conditions has been an area of significant research, with studies examining various performance trade-offs, including latency, throughput, scalability, and robustness. While theoretical guidelines provided by model providers such as OpenAI offer optimization principles, empirical studies provide quantitative insights into how models perform under load, informing model selection for practical applications. This section reviews existing literature related to scalability, inference efficiency, and trade-offs in large language model (LLM) performance, positioning our work within this broader research landscape.

2.1. Scalability and Model Performance Trade-offs

The scalability of LLMs has been studied in various contexts, from inference latency and concurrency handling to robustness against adversarial attacks. Sun and Miceli-Barone (2024) investigate the scaling behaviour of LLMs in machine translation, revealing an inverse scaling phenomenon, where larger models, while more powerful, become increasingly vulnerable to prompt injection attacks in zero-shot settings [2]. Their findings highlight that scaling alone does not always lead to improved performance, reinforcing the importance of efficiency optimizations in model selection. While their research primarily focuses on adversarial robustness, it provides valuable insights into scalability trade-offs, complementing our investigation into how OpenAI models handle increasing concurrency levels in real-world deployments.

Beyond adversarial robustness, Tran et al. (2024) explore multi-dimensional LLM performance analysis in structured evaluation tasks, finding that reducing context length enhances reliability but increases computational cost [3]. Their study underscores the need for task-specific optimizations, a theme that aligns with our findings on how different OpenAI models exhibit varying performance patterns depending on prompt complexity and concurrency load. By highlighting efficiency trade-offs in structured assessments, their research reinforces our work's emphasis on balancing response speed, resource usage, and scalability in production environments.

2.2. Inference Optimization and Latency-Throughput Trade-offs

Efforts to optimize LLM inference efficiency have introduced novel techniques for managing throughput and latency, especially in large-scale AI applications. Agrawal et al. (2024) address this challenge by introducing Sarathi-Serve, a scheduling framework that optimizes request batching for scalable LLM inference [4]. Their study identifies inefficiencies in existing model-serving infrastructures, demonstrating that prioritizing either latency or throughput alone results in suboptimal performance under high concurrency. By implementing Chunked-Prefills and Stall-Free Scheduling, Sarathi-Serve achieves 5.6× improvements in throughput while maintaining low latency. These findings are directly relevant to our work, as we examine how OpenAI models respond to concurrent requests, providing empirical evidence on performance bottlenecks and API scalability trade-offs.

Similarly, Jaiswal et al. (2025) introduce SAGESERVE, a hybrid inference framework designed to dynamically allocate GPU resources based on workload demand [5]. Their research highlights that traditional GPU allocation strategies either over-provision or underutilize compute resources, leading to inefficiencies in large-scale AI applications. By combining reactive and predictive scaling strategies, SAGESERVE improves GPU-hour utilization by 25% while maintaining response-time guarantees. These findings align with our study’s focus on model efficiency under concurrency stress, reinforcing the importance of dynamic resource scheduling in real-world LLM deployments.

2.3. Model Evolution and Evaluation Frameworks

Understanding the historical evolution of LLMs is crucial for contextualizing their current scalability challenges. Zhou et al. (2023) provide a comprehensive survey on Pretrained Foundation Models (PFMs), tracing their development from early transformer architectures (e.g., BERT, GPT-3) to modern large-scale models (e.g., GPT-4, ChatGPT) [6]. Their analysis highlights key scalability and efficiency challenges, particularly in instruction tuning and inference optimizations. While their survey does not specifically examine model concurrency, it offers critical background on why scaling strategies differ across LLM architectures, informing our study on how OpenAI models perform under varying levels of concurrent requests.

A more structured evaluation approach is presented by Liang et al. (2023) in HELM (Holistic Evaluation of Language Models), a benchmarking framework that assesses 30 major LLMs, including OpenAI's GPT-3 and InstructGPT, across multiple performance dimensions [7]. HELM evaluates models on accuracy, robustness, fairness, efficiency, and bias, providing standardized metrics for LLM comparison. Although HELM does not directly assess concurrent API performance, its emphasis on efficiency trade-offs complements our empirical analysis of latency degradation, throughput bottlenecks, and token processing efficiency in OpenAI models.

2.4. Positioning Our Work

While prior research has explored scalability, inference efficiency, and task-specific LLM optimizations, there remains a gap in empirical studies on how OpenAI's models handle concurrent requests in real-world applications. Our study builds on these foundational works by conducting a quantitative analysis of latency, throughput, and token efficiency under varying concurrency levels, bridging the gap between theoretical model optimizations and practical performance trade-offs. By benchmarking o1-mini, o1-preview, GPT-4o, and GPT-4o-mini, we extend prior research on scalability and efficiency, offering actionable insights for model selection in high-load environments.

3. Implementation

This section details the experimental framework developed to empirically evaluate the performance of selected OpenAI language models under real-world load conditions. Our approach systematically measures key performance metrics like average response time, throughput, and token efficiency; across diverse task categories using asynchronous load testing. The following subsections describe our methodology, organized into four parts.

3.1. Prompt Categories

To capture a broad spectrum of use cases, we evaluate each model on five distinct prompt categories. These categories were chosen to represent tasks ranging from explanatory content and creative narrative generation to technical explanations, translation, and coding. Table I summarizes these categories along with descriptions and representative sample prompts.

Table 1 Prompt Categories

Prompt Category	Description	Sample Prompt
Explanatory	Requires clear explanation of complex concepts.	"Explain the principles behind quantum computing in simple terms."
Creative	Demands narrative generation and creative storytelling.	"Write a short story about an AI that saves a city from a cyber-attack, in a suspenseful tone."
Technical	Focuses on precise technical explanations and comparisons.	"What are the key differences between supervised and unsupervised machine learning? Provide examples."
Translation	Involves converting text between languages with an emphasis on brevity.	"Translate the following sentence into French: 'The future of technology is bright and full of innovation.'"
Coding	Requires generating functional code or algorithmic solutions.	"Write a Python code for binary search."

3.2. Load Testing Setup

The performance evaluation was conducted using an asynchronous load testing framework implemented in Python. This framework leverages the *asyncio* [8] library to simulate real-world, concurrent usage of language models. For each prompt category, we executed tests at multiple concurrency levels, specifically:

$$Concurrency\ Levels = \{\ 1, 2, 5, 10, 20, 30, 40\ \}$$

At each level, 5 requests per model were issued. This setup enables us to observe how the models perform as the number of simultaneous requests increases, thereby providing insights into their scalability, latency, and overall efficiency.

3.3. Metrics Extraction and Equations

Key performance metrics were derived using the following definitions and equations:

Average Response Time ($\bar{T}$): The mean time required for a model to generate a response: $\bar{T} = \frac{1}{N}\sum_{i=1}^{N} T_i$

where T_i represents the response time for the *i*th request and N is the total number of requests.

Throughput (η): The number of requests processed per second: $\eta = \frac{N}{T_{total}}$

where T_{total} is the total elapsed time for processing N requests.

Token Efficiency ($\bar{U}$): The average number of tokens generated per response: $\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U_i$

with U_i representing the token count for the *i*th response.

These equations underpin our quantitative analysis and support the visual and statistical comparison of model performance.

3.4. Implementation Details

The overall architecture of our implementation is composed of several modules designed to ensure reproducibility and consistency:

- **API Call Wrappers**: Custom functions were developed to interface with each OpenAI model. These wrappers are responsible for sending prompts, recording the response time, and extracting token usage where available.
- **Asynchronous Load Testing Engine**: Using Python's asyncio and semaphore-based concurrency control, the engine orchestrates parallel API calls. This design ensures that the number of concurrent requests adheres to predefined levels, mimicking real-world load scenarios.
- **Data Aggregation and Analysis**: Response data is aggregated into a structured, nested dictionary organized by model, prompt category, and concurrency level. This dataset is then used to compute average response times, throughput, and token efficiency using the equations described above.
- **Visualization**: Finally, the aggregated metrics are visualized through a series of plots, with separate graphs for average response time, throughput, and token efficiency. These visualizations facilitate a comprehensive, side-by-side comparison of model performance under varying load conditions.

This implementation framework not only complements the theoretical guidelines provided by OpenAI but also extends them by offering empirical, data-driven insights. By systematically evaluating these models under controlled conditions, our study enables practitioners to make informed decisions based on measurable performance trade-offs.

4. Results

In this section, we present a detailed analysis of the experimental findings obtained from our load testing framework. The performance metrics—average response time, throughput, and token efficiency—were computed across five prompt categories: Explanatory, Creative, Technical, Translation, and Coding. The following subsections describe the observations for each metric, supported by quantitative data and summarized in Table II.

4.1. Average Response Time

The average response time is defined as the mean duration required for a model to generate a response. Our experiments, conducted over seven concurrency levels (1, 2, 5, 10, 20, 30, 40) with 5 requests per level, revealed distinct latency profiles for each model:

- Explanatory Prompts: GPT-4o consistently produced the lowest response times, whereas o1-preview exhibited significantly higher latencies.
- Creative Prompts: Although GPT-4o maintained stable and low response times, GPT-4o-mini showed increased variability at higher concurrency levels. In this category, o1-preview again lagged behind.

- Technical Prompts: GPT-4o outperformed the other models with response times typically between 7 and 9 seconds. In contrast, o1-preview's response times fluctuated between 18 and 25 seconds, while o1-mini and GPT-4o-mini showed intermediate performance.
- Translation Prompts: All models achieved lower response times in this category; notably, GPT-4o managed sub-second responses, with o1-preview trailing at 6–7 seconds.
- Coding Prompts: The fastest responses were observed with GPT-4o (approximately 5–6 seconds), whereas o1-preview peaked near 23 seconds. GPT-4o-mini and o1-mini maintained moderate response time

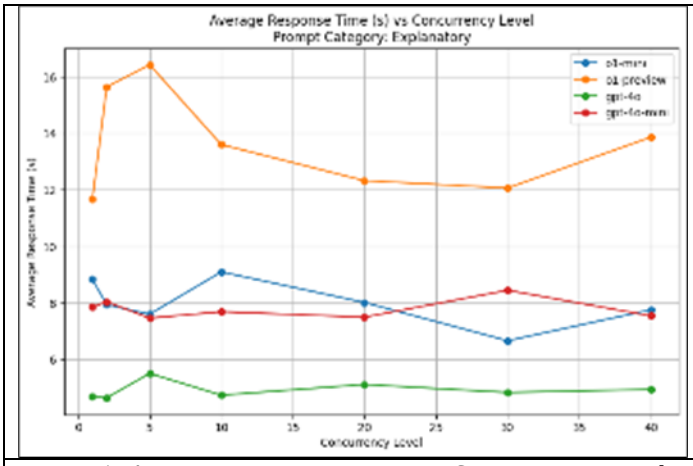

Figure 1: Average Response Time vs. Concurrency Level – Explanatory

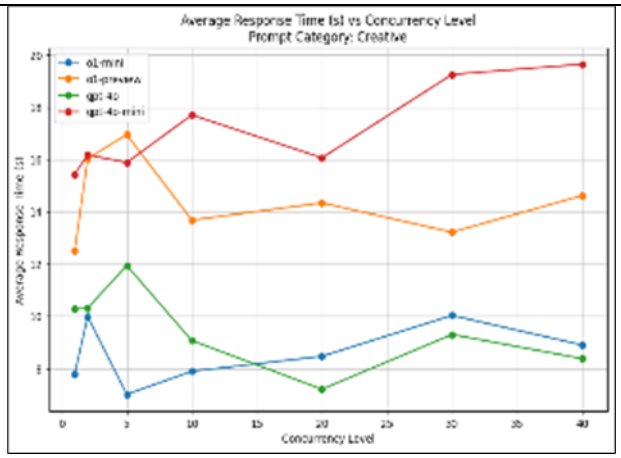

Figure 2: Average Response Time vs. Concurrency Level - Creative

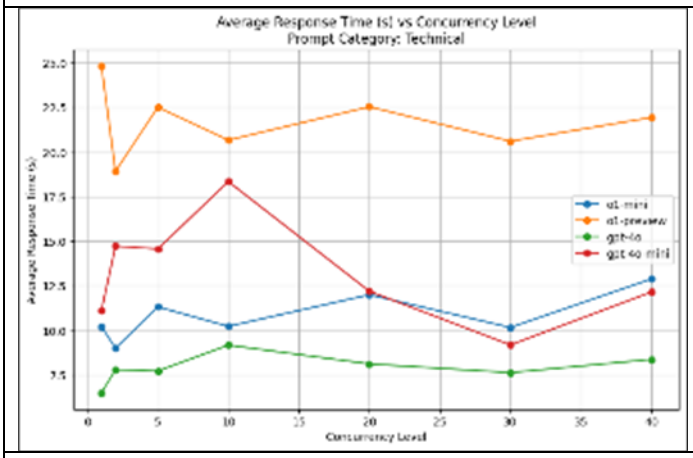

Figure 3 Average Response Time vs. Concurrency Level – Technical

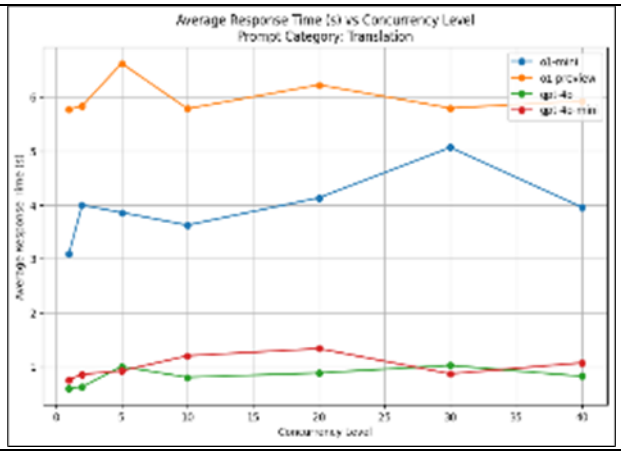

Figure 4 Average Response Time vs. Concurrency Level - Translation

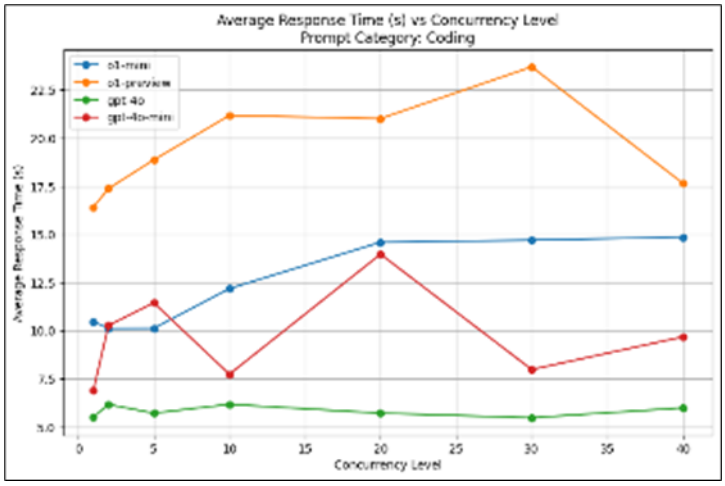

Figure 5 Average Response Time vs. Concurrency Level – Coding

4.2. Throughput

Throughput is measured as the number of requests processed per second. It is calculated by dividing the total number of requests by the elapsed time for the batch. Our findings include:

- Explanatory Prompts: GPT-4o achieved the highest throughput, peaking near 1.0 req/s, while o1-preview maintained the lowest throughput.
- Creative Prompts: Both GPT-4o and o1-mini delivered high throughput at lower concurrency levels; however, GPT-4o sustained its performance as concurrency increased.
- Technical Prompts: GPT-4o again led the performance, reaching around 0.6 req/s, whereas o1-preview remained near 0.2 req/s.
- Translation Prompts: This category showed exceptional throughput for GPT-4o and GPT-4o-mini, with peak values as high as 5.5 req/s. The other models lagged significantly.
- Coding Prompts: GPT-4o demonstrated superior throughput (~0.75 req/s), while o1-preview consistently showed lower throughput (~0.2 req/s).

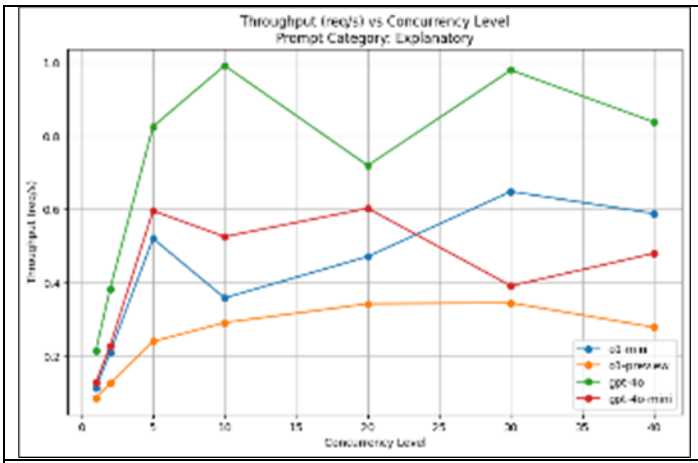

Figure 6 Throughput vs. Concurrency Level – Explanatory

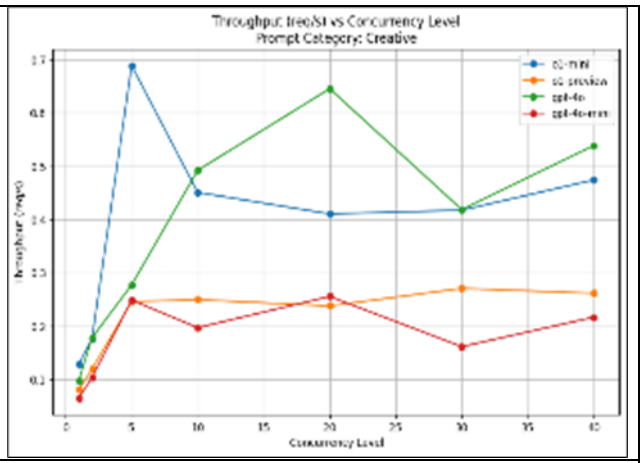

Figure 7 Throughput vs. Concurrency Level - Creative

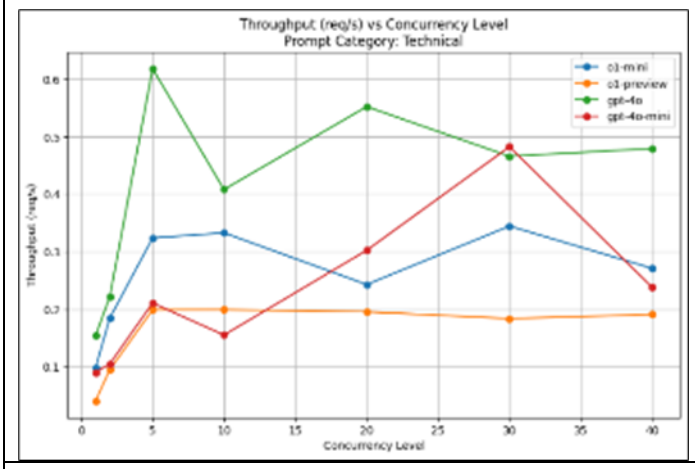

Figure 8 Throughput vs. Concurrency Level – Technical

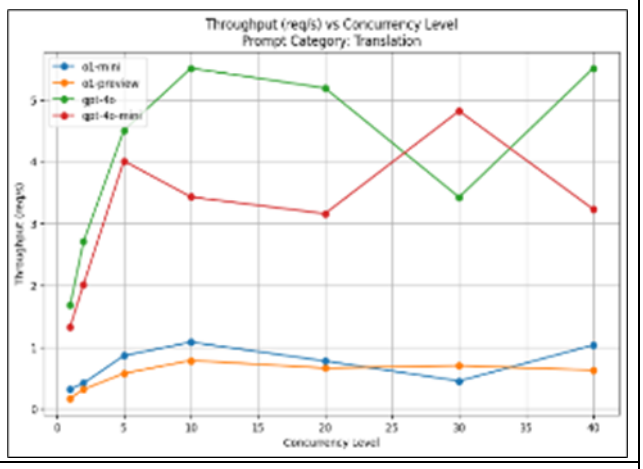

Figure 9 Throughput vs. Concurrency Level - Translation

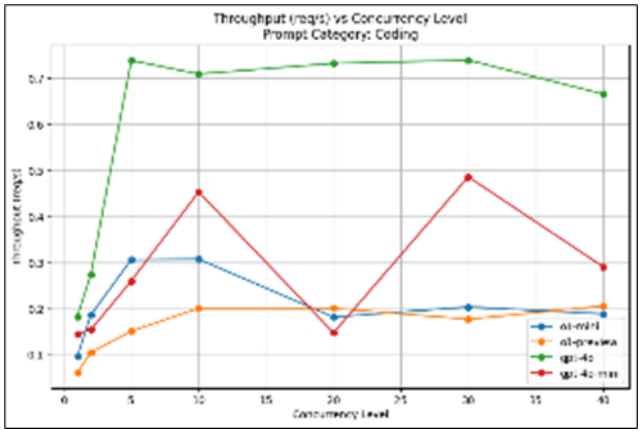

Figure 10 Throughput vs. Concurrency Level – Coding

4.3. Token Efficiency

Token efficiency reflects the average number of tokens generated per response. Lower token usage indicates more concise outputs without sacrificing quality:

- Explanatory Prompts: o1-preview consumed up to ~1600 tokens per response, with o1-mini slightly lower (approximately 900–1100 tokens). In contrast, GPT-4o and GPT-4o-mini maintained a token count around 400–500.
- Creative Prompts: GPT-4o led in efficiency by averaging between 700 and 900 tokens, while o1-preview peaked around 1800 tokens, and o1-mini showed moderate verbosity (1000–1300 tokens).
- Technical Prompts: o1-preview and o1-mini produced more verbose responses (up to ~2500 and ~1400–1500 tokens respectively), while GPT-4o and GPT-4o-mini stayed under ~700 tokens.
- Translation Prompts: In tasks requiring brevity, GPT-4o and GPT-4o-mini were the most efficient (50–100 tokens), compared to o1-mini (~300–400 tokens) and o1-preview (~450–550 tokens).
- Coding Prompts: GPT-4o and GPT-4o-mini averaged around 500 tokens per response, whereas o1-preview and o1-mini generated outputs ranging from 1200 to over 2200 tokens.

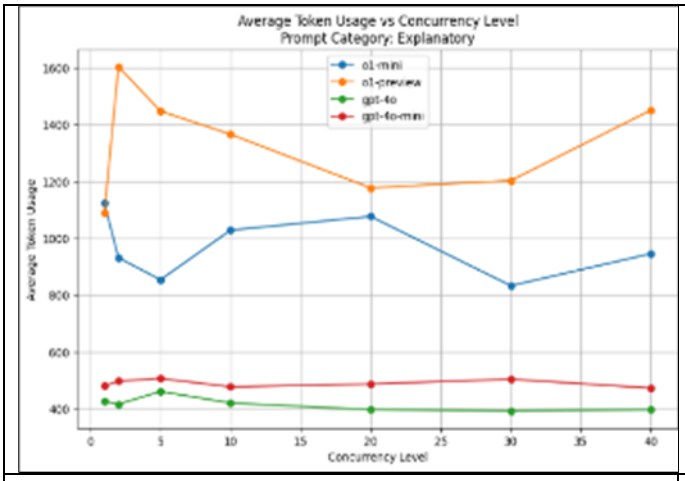

Figure 11 Average Token Usage vs. Concurrency Level – Explanatory

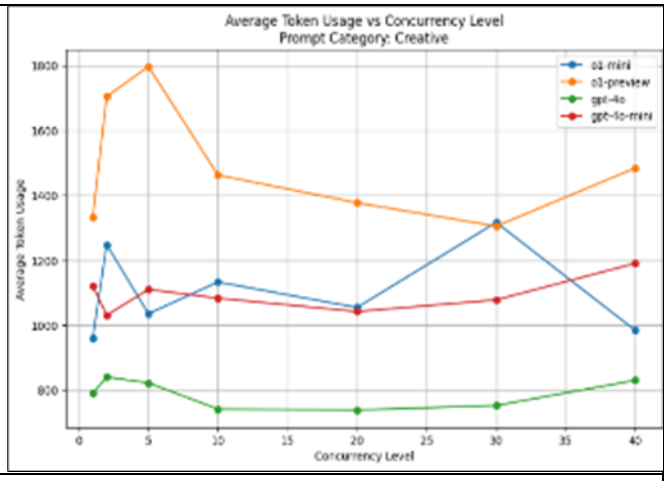

Figure 12 Average Token Usage vs. Concurrency Level - Creative

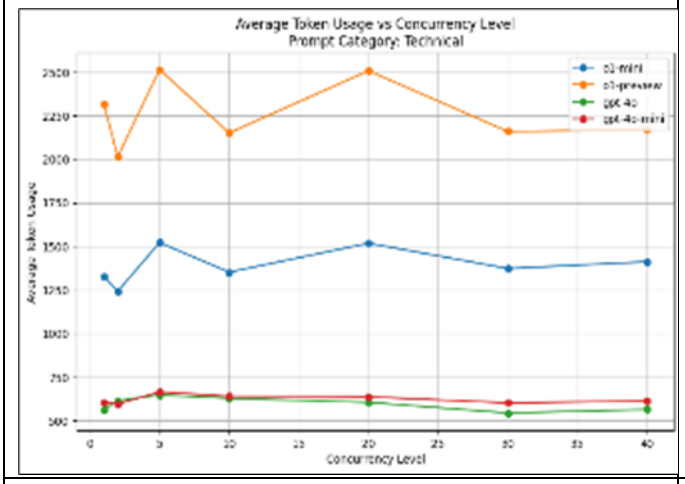

Figure 13 Average Token Usage vs. Concurrency Level – Technical

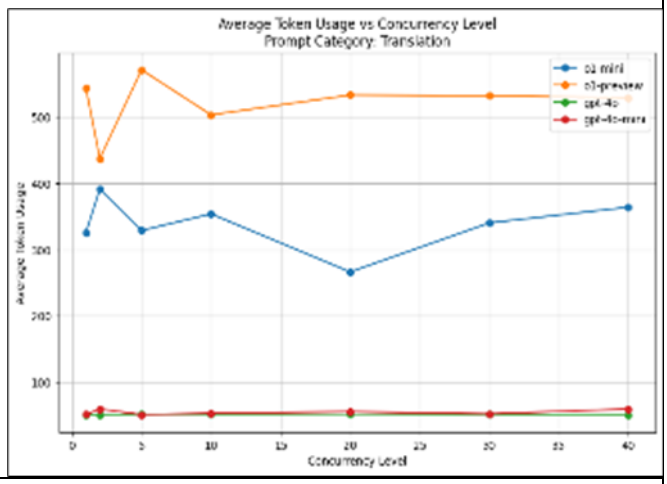

Figure 14 Average Token Usage vs. Concurrency Level - Translation

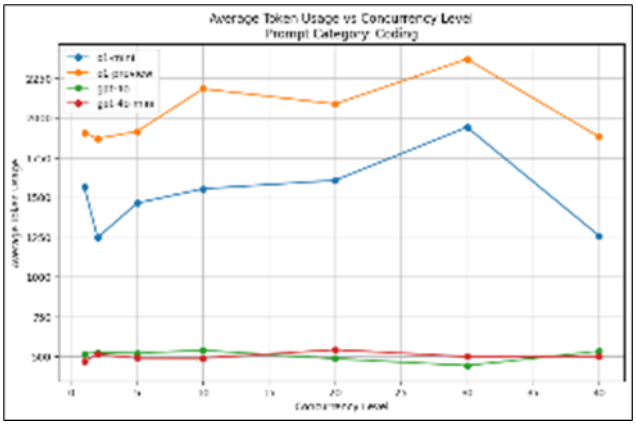

Figure 15 Average Token Usage vs. Concurrency Level – Coding

4.4. Combined Performance Summary

The following table (Table II) summarizes the overall performance characteristics of each model across the three key metrics.

Table 2 Combined Performance Summary

Model	Average Response Time	Throughput (req/s)	Token Efficiency
GPT 4o	Lowest response times across categories	Highest throughput; excels in translation tasks	Most efficient; lowest token usage
GPT 4o-mini	Moderate response times; some variability at high load	High throughput, slightly below GPT-4o	Highly efficient, slightly more verbose than GPT-4o
o1-mini	Moderate performance with increased latency at high concurrency	Moderate throughput	More verbose compared to GPT-4 series
o1-preview	Consistently highest response times across categories	Lowest throughput; poor scalability	Most verbose; highest token consumption

5. Discussion

The experimental findings reveal critical trade-offs in model performance under load. The superior performance of GPT-4o, evidenced by its minimal response times, high throughput, and efficient token usage, makes it a favorable choice for applications that demand rapid and scalable performance. The data indicate that GPT-4o's advantages are particularly pronounced in translation and technical tasks, where latency and resource efficiency are paramount.

Conversely, while GPT-4o-mini offers a viable alternative, its performance variability under high concurrency suggests potential limitations in stability that may affect user experience in peak load scenarios. The o1 series, particularly o1-preview, consistently underperformed across all metrics, indicating that their design may favor other aspects (such as creativity or domain-specific optimizations) over pure latency and efficiency.

These results extend the theoretical guidelines provided in the OpenAI API documentation by grounding them in empirical data. While the documentation outlines optimization principles such as reducing token count and parallelizing requests, our study quantifies the impact of these strategies under real-world conditions. The comprehensive comparative analysis presented here enables practitioners to make informed decisions by aligning model selection with the specific operational requirements of their applications.

6. Conclusion

In conclusion, our empirical evaluation under varying concurrency levels revealed distinct performance profiles for the four models tested. GPT-4o consistently achieved the lowest response times and highest throughput, maintaining its advantage even as concurrency increased, while also proving the most token-efficient by using the fewest tokens per response. GPT-4o-mini came close to GPT-4o's performance but exhibited slight latency increases and higher variability at peak loads, indicating some instability under extreme concurrency. In contrast, o1-mini delivered only moderate performance, with latency rising significantly at higher concurrency levels. At the lower end of the spectrum, o1-preview lagged in all metrics, suffering the longest response times, lowest throughput, and highest token consumption. This severe performance degradation under concurrent load reflects poor scalability for o1-preview.

These concurrency-induced performance disparities and degradations have direct implications for stability and scalability in real-world deployments. Notably, latency directly affects end-user experience, throughput determines how well a service scales with parallel requests, and token efficiency influences resource consumption and cost. Models that degrade significantly under heavy concurrency (such as o1-preview and, to a lesser extent, GPT-4o-mini) can produce inconsistent response times and limit a system's ability to scale to high user volumes. Conversely, GPT-4o's robust performance at scale indicates greater reliability for serving many simultaneous users. From a practical perspective, our findings guide practitioners in aligning model choice with operational demands. For applications that must sustain high throughput with minimal latency, GPT-4o stands out as the optimal choice. If smaller models (e.g., GPT-4o-mini or o1-mini) are preferred due to cost or other constraints, the performance trade-offs identified here should inform capacity planning and system design. By quantifying latency, throughput, and token usage under realistic conditions, this study provides a rigorous, data-driven foundation that complements theoretical guidelines and informs model deployment decisions in production environments.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] OpenAI, "Latency Optimization," OpenAI Documentation. [Online]. Accessed: 12 December 2024. Available: https://platform.openai.com/docs/guides/latency-optimization.

[2] Sun Z, Miceli-Barone AV. Scaling behavior of machine translation with large language models under prompt injection attacks. arXiv preprint arXiv:2403.09832. 2024 Mar 14.

[3] Tran N, Pierce B, Litman D, Correnti R, Matsumura LC. Multi-Dimensional Performance Analysis of Large Language Models for Classroom Discussion Assessment. Journal of Educational Data Mining. 2024 Dec 23;16(2):304-35.

[4] Agrawal A, Kedia N, Panwar A, Mohan J, Kwatra N, Gulavani B, Tumanov A, Ramjee R. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24) 2024 (pp. 117-134).

[5] Jaiswal S, Jain K, Simmhan Y, Parayil A, Mallick A, Wang R, Amant RS, Bansal C, Rühle V, Kulkarni A, Kofsky S. Serving Models, Fast and Slow: Optimizing Heterogeneous LLM Inferencing Workloads at Scale. arXiv preprint arXiv:2502.14617. 2025 Feb 20.

[6] Zhou C, Li Q, Li C, Yu J, Liu Y, Wang G, Zhang K, Ji C, Yan Q, He L, Peng H. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. International Journal of Machine Learning and Cybernetics. 2024 Nov 24:1-65.

[7] Bommasani R, Liang P, Lee T. Holistic evaluation of language models. Annals of the New York Academy of Sciences. 2023 Jul;1525(1):140-6.

[8] Python Software Foundation, "asyncio — Asynchronous I/O," Python Documentation. [Online]. Accessed: 19th November 2024. Available: https://docs.python.org/3/library/asyncio.html.