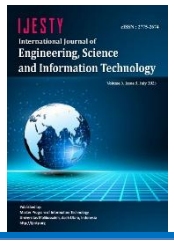
International Journal of Engineering, Science and Information Technology
Volume 5, No. 3 (2025) pp. 363-372
eISSN: 2775-2674
Website: http://ijesty.org/index.php/ijesty
DOI: https://doi.org/10.52088/ijesty.v5i3.955
Research Paper, Short Communication, Review, Technical Paper



Benchmarking Techniques for Real-Time Evaluation of LLMs In Production Systems

Reena Chandra[1]*, Rishab Bansal[2], Karan Lulla[1]

[1]*Independent Researcher, San Francisco, California, United States of America*
[2]*Independent Researcher, Fremont, California, United States of America*

**Corresponding author E-mail: reenachandra11@gmail.com*

The manuscript was received on 1 February 2025, revised on 10 March 2025, and accepted on 15 June 2025, date of publication 26 June 2025

Abstract

Large language models (LLMs) should perform reliably and work efficiently in today's applications that use AI chatbots, copilots, and search systems. Usually, the traditional type of benchmarking deals mainly with understanding linguistics and accurate task performance, while important factors like latency, how much memory is used, and optimisation are ignored. A benchmarking framework is proposed in this study that reviews LLMs using four critical factors: number of tokens processed per second, accuracy, peak memory usage, and Efficiency. Using the Open LLM Performance dataset, 350 open-source models were examined with standardised tools and methods across various families and sizes of parameters. According to the studies, the TinyStories-33M and OPT-19M middle-scale models are ideal for practical use because they handle many words per second without taking up much memory. ONNX Runtime uses less memory than PyTorch, and applying LLM.fp4 quantisation greatly increases throughput without a significant loss in accuracy. Visualisations and ranks are presented to help choose a production model. By following the framework, AI engineers, MLOps teams, and system architects can spot innovative models that can be built, deployed, expanded, and managed within budget. It improves LLM assessment by relating technical measures to practical limitations in real systems so that smarter choices can be made for systems used in actual operations.

Keywords: *Large Language Models, Real-Time Inference, Benchmarking, Throughput, Production Deployment.*

1. Introduction

The rapid evolution of Large Language Models (LLMs) has significantly transformed various industries, necessitating more sophisticated evaluation frameworks that extend beyond traditional accuracy metrics. According to Hendrycks et al. [1], while benchmarks like MMLU have been instrumental in assessing language understanding, they often fail to capture the real-time performance and efficiency required for production environments. Recent developments underscore this necessity; for instance, NVIDIA's DGX B200 system achieved a record-breaking 1,038 tokens per second per user by leveraging optimisations such as TensorRT and speculative decoding. Agrawal et al. [2] emphasised that introducing frameworks like Etalon emphasises the need for holistic performance metrics that consider user-facing experiences in real-time applications. These advancements highlight a paradigm shift towards comprehensive benchmarking approaches that assess LLMs' suitability for deployment in dynamic, resource-constrained environments.

With large language models (LLMs), it is now possible to build accurate systems for asking questions, summarising documents, automating translation, and activating dialogue. Ragsdale and Boppana [3] highlighted that chat support for searching information, writing programming code and helping assistants provide answers, LLMs are becoming important in many live-usage apps. Now, these deployments are used in businesses and scalable systems, as they need to work well, respond quickly and be efficient with costs. At the same time, using LLMs in real-time settings is a serious engineering problem. A fundamental conflict exists between latency, required memory, and the model's accuracy. Menghani [4] observed that accurate models are usually demanding to execute and assess, so they take time to respond even to simple inputs in cases with latency issues. In contrast, optimised models for performance and size usually lack meaning and sound reasoning. It is essential in production settings like customer support or mobile apps, since users rely on quick, steady and dependable service that causes little trouble for the server.

Chang et al. [5] noted that LLMs are becoming necessary based on their production results; most existing benchmarks concentrate on simple accuracy in language or type of tasks. Zhang et al. [6] observed that MMLU, Stanford HELM and Hugging Face's Open LLM Leaderboard focus closely on accuracy in multiple-choice questions, matching questions exactly or BLEU scores, respectively [6]. Even though they assess cognitive abilities well, these tools still omit results for aspects like per-second throughput, memory usage during running and backend differences. Paleyes et al. [7] pointed out that as a result of this difference in methodology, it is more challenging to

make informed choices when deploying models in practice, when costs, computing power or infrastructure are part of the challenge. A practical benchmarking framework is presented in this study to help fill the critical gap in real-time analysis of LLMs. Ahmed et al. [8] introduced the Open LLM Performance dataset, which offers standard throughput, memory and performance scores across numerous open models. This allows this research to analyse their operational features in detail. Considering other factors and accuracy, this model method will enable stakeholders to choose those that are suitable for their production environment.

This study aims to assess LLMs using various standards that represent how they are used in practice. Efficiency through score-memory ratio and the number of required table cells for deployment are explicitly used in our benchmarks. The evaluation is enhanced by running backend systems like PyTorch and ONNX and trying various approaches, such as BetterTransformer and different quantisation types (FP4 and int8), which strongly affect real-time performance. Significant results come from the work done in this benchmarking initiative. Originally, the table charts help illustrate how different models' throughput, memory, and Score relate to one another in models spanning from TinyStories to LLaMA and FreeWilly2. Such plots quickly provide the most efficient and budget-friendly combinations of model settings for digital use. Additionally, this research identifies two leading models, Salesforce/codegen-16B-nl for the best accuracy-to-memory ratio and TinyStories-33M for the quickest throughput while needing less memory.

Furthermore, results demonstrate that ONNX Runtime uses less memory than PyTorch and can be chosen for websites that must work efficiently despite having limited resources. This work offers practical knowledge about LLMs' performance differences in real-world scenarios to AI practitioners, MLOps engineers and decision-makers. Because it includes latency, memory, runtime analysis, and accuracy, the new framework improves how real-time LLMs are assessed and aids in making better, more efficient decisions about their use.

2. Literature Review

2.1. Review of Academic Benchmarks

Bommasani, Liang, and Lee [9] emphasised that benchmarking LLMs is now essential as their abilities and uses expand rapidly. Among the vital research programmes is Stanford's HELM (Holistic Evaluation of Language Models), which assesses the models and their features for accuracy, fairness, strength, reliability and Efficiency. Since HELM examines models in multiple ways, it gives a more complete picture of behaviour, yet measures "efficiency" mostly by looking at model size or FLOPs, instead of how fast and/or much memory is used in the real application. Similarly, MMLU (Massive Multitask Language Understanding) is commonly used to measure how much an LLM knows and can reason in 57 different topic areas [10]. Although MMLU can assess learning and general thinking, it is not meant to check model operation in real situations. EleutherAI develops the LM Harness and provides a flexible system for evaluating many NLP tasks (Bommasani et al., 2023). While neural networks are commonly used in education and development, they do not yet offer built-in tools for measuring latency, memory use or throughput in real production settings. McIntosh *et al.* [11] argued that academic benchmarks are excellent for assessing model intelligence and language skills but fail to prepare models for practical use. They usually analyse models in perfect, fixed environments, paying little attention to issues such as GPU restrictions, how much time users want their systems to react and the real number of resources used. Consequently, these standards aren't enough for ML teams hoping to roll out models quickly, efficiently and as needed.

2.2. Limitations in Real-Time Evaluation

Hodak *et al.* [12] pointed out that Current LLM benchmarking systems emphasise accuracy, but do not pay much attention to metrics needed for actual deployment. For any app where users take action, latency measured as tokens per second is significant. Few systems test or report the performance of throughput under real situations. In addition, how much memory a model requires at peak is very important for choosing where to deploy it, but this information is frequently left out of the assessment process. There are also not enough diverse experts behind the evaluation of these programmes. Many academic measures analyse models using PyTorch with its default parameters. Zhou and Yang [13] demonstrated that, to be practical, having ONNX Runtime, TensorRT or TorchScript as an inference engine can significantly improve how a model operates, making it faster and more efficient with memory. Benchmarks do not analyse how models operate across different runtimes, which can cause a model that does well in PyTorch to act differently when applied by a lightweight runtime such as ONNX. Benchmarking without considering optimisation is currently a big problem. Chitty-Venkata *et al.* [14] noted that Real-time performance of models can be strongly affected by practices such as quantising weights, speeding up transformers and performing optimisations at the graph level. However, such factors are not included in typical academic measures. Consequently, production engineers lack the data to pick the best runtime configurations or improvement strategies.

2.3. Need for Real-Time Benchmarking Framework

With today's basic benchmarks, companies need a method that measures their cognitive strength and how well they run operations simultaneously. Saxena *et al.* [15] emphasised that peak memory used, the accuracy of a model's predictions, and a combined metric to express performance against resource use must all be part of the framework. It must check models with different deployment situations and hardware, using tests that simulate the real-world use of the model. Lightweight LLMs have lately been applied in specific industry settings with limited resources. Tang *et al.* [16] demonstrated that the TinyStories family of models has achieved excellent throughput, sending up to 400 tokens per second and requiring no more than 3 GB of memory. While their accuracy is poor in benchmark tests (around 30%), their efficient operation draws interest for applications like summarising and answering questions on mobile or embedded systems. Traditional assessments are more likely to pass over those models because of their weak academic standing, despite their practical benefit. Two other systems, OPT-19M-ChatSalad and Salesforce/codegen-16B-nl, have also shown great potential for real-time tasks. Jeon [17] highlighted that it is the most efficient model in this case and requires only 128 MB of memory, so it's a good choice for inference in edge or serverless settings. The results demonstrate that comparative frameworks are more meaningful when they value high-efficiency models just as much as classic accuracy criteria.

2.4. Research Gap

The Open LLM Performance dataset bridges this gap by compiling standard and in-depth metrics for a broad group of open-source models. Ahmed *et al.* [8] stated that it measures throughput, uses the maximum amount of memory, looks at accuracy and evaluates backend performance to give a complete picture of LLM use in real production. Alsaqer *et al.* [18] and Saha *et al.* [19] noted that researchers and engineers use the dataset to decide which approach gives the best accuracy while being fast and suited for hardware. This study uses the

strengths of the dataset to offer a framework that clearly shows and explains the compromises between the assessed metrics. With Score compared to throughput on a scatter plot, it's evident that TinyStories excels in speed but is less accurate, whereas FreeWilly2 has high accuracy but takes longer to process data and eats up a large amount of memory. Similarly, this chart summarises how efficient small models are with memory use versus how much overhead the large ones have. With these visualisations, practitioners can find the best option for their constraints during deployment.

Additionally, using ONNX Runtime shows that it typically uses far less memory than PyTorch, sometimes less than half as much. Durán *et al.* [20] highlight that by using ONNX efficiently, the study can switch to edge deployment and offer multiple users the ability to use their inferred models. Apart from this, the research evaluates different approaches, noting that improvements such as BetterTransformer and LLM.fp4 increase system speed, but approaches like LLM.int8 can decrease performance in specific types of workloads. As a result, benchmarking frameworks should be used to evaluate the real effects of optimisation. In conclusion, without immediate evaluation, these traditional benchmarks cannot be used well in today's AI applications, which need to be speedy and take up less memory. The study requires measuring accuracy, throughput, memory, Efficiency, and runtime optimisations in a unified efficiency evaluation. This work fills this need by introducing and validating the Open LLM Performance dataset framework, offering helpful advice to people in artificial intelligence, infrastructure engineering, and decision-making roles who must apply LLMs in practice. This framework helps close the gap between academic evaluations and their use in practice.

3. Methods

3.1. Benchmarking Framework

The proposed framework aims to evaluate LLMs by determining their outputs' accuracy and whether they are ready to work in real-time production settings. Frequently, the main goal of these benchmarks is to ensure that a model is correct and fails to consider deployment challenges. The metrics grouped in this framework show a model's performance, how quickly it responds, its ability to go live, and its Efficiency in real production environments (see Figure 1).

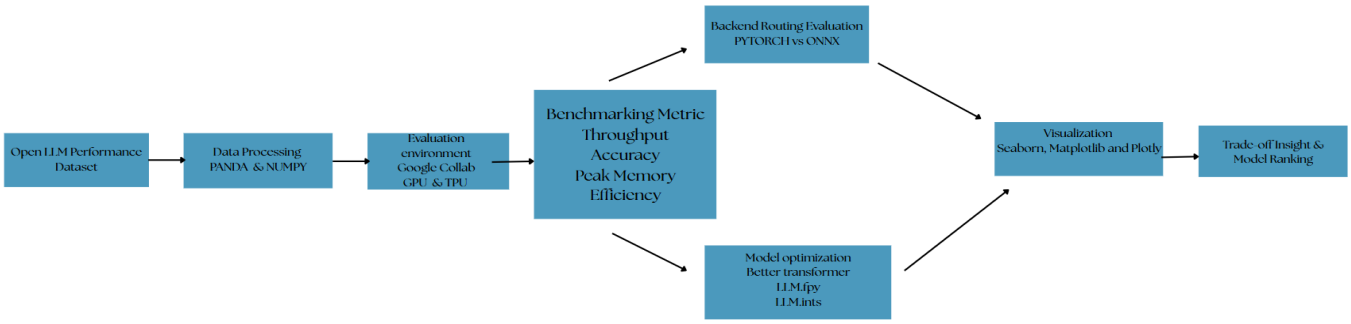


Fig. 1: Proposed Framework

When you trade, the main thing to watch is your throughput rate, measured as tokens per second. Because inference speed and throughput are directly tied, this is the most critical metric for apps that rely on low latency, such as chatbots, real-time summarisers and voice assistants. Higher throughput mostly means responses happen more quickly, which is noticeable and enjoyable for the user. Rather than measuring latency, we use throughput to represent it by looking at batch-inference times of multiple models. The second measure is accuracy, which is based on the percentage in benchmark test data found in the Open LLM Performance dataset. This indicates how well the model understands the text, can reason, and generally completes different NLP tasks. Still, accuracy needs attention when matching the throughput and amount of memory required for the applications. Models designed for outstanding accuracy but are slow or require a great deal of memory may not be suitable for user-facing tasks or places where costs are essential. The third metric is how much memory is taken up at the highest point and is given in megabytes. The result shows if a model can be used in resource-limited areas, including mobile phones, embedded systems or cloud microservices with limited RAM. High memory requirements in a process can cause infrastructure expenses. They may result in problems with the app's ability to operate steadily or respond more slowly after being restarted in serverless solutions. A composite indicator called Efficiency is added to the framework, which calculates the ratio of Score to the amount of memory used (score ÷ memory). It shows how much you can get out of your model for each amount of resource you use. These highly efficient models strike an outstanding balance between usefulness and being producible, so they are given priority for implementation. Thanks to this, the stakeholders can detect AI models that provide the most significant performance boost for the lowest overhead, which is vital for using AI economically and for expandability.

3.2. Dataset

This benchmarking framework depends on the Open LLM Performance Dataset, a carefully selected and thorough place where over 350 open-source LLMs are evaluated and where data is stored. Several of these models have ultra-light parameter scales, while LLaMA-65B and FreeWilly2 are the largest, using billions of parameters. All empirical results in the dataset come from tests on standard hardware, guaranteeing that all models show consistent and comparable results. The models are sorted into groups by size: Tiny for models under 100 million, Small for models between 100 million and 1 billion, Medium for models with 1 billion to 10 billion parameters and Giant for models with more than 10 billion parameters. Comparisons in one scale cohort are balanced because of this system, and it's easier to notice if the firm's performance matches the threshold. Different architectures, pretraining styles and arrangements of parameters are shown through LLaMA, MPT, OPT, TinyStories and FreeWilly.

Most importantly, the dataset contains results from PyTorch and ONNX Runtime inference backends. PyTorch is the leading tech used for research and learning, although it's not designed for quick execution during deployment. In contrast, ONNX Runtime is lighter and faster in inference. It features operator fusion, optimises graphs, and supports quantised computation. The study measures how much flexibility and efficiency change depending on the cloud backend.

Several optimisation methods are also included in the dataset. These include:

1. BetterTransformer is a runtime transformer that optimises performance by parallelising the process.
2. LLM.fp4 is a version of a small, strong language model that performs faster and still gives accurate predictions.
3. LLM.int8 uses 8-bit data to shrink memory use, though how it influences throughput depends on the model design and the hardware used.

Using backend and optimisation features allows a thorough evaluation of the model's work based on its architecture, the tools used and how it will be deployed.

3.3. Evaluation Setup

The performance measurements were made in Google Colab Pro+, using NVIDIA T4 GPUs and TPU v3 to simulate real production settings. These platforms represent a balance between advanced research servers and edge devices, enabling us to assess the practicality of deployment under different situations. The data was evaluated using a standard Python pipeline with essential data science and visualisation libraries. The preprocessing tasks and metric computations depended on Pandas and NumPy, and to show the results, the study turned to Seaborn, Matplotlib and Plotly. Among the visual outputs are:

1. Boxplots are used to visualise the distribution of throughput by model class.
2. Comparing accuracy with throughput and memory usage using scatter plots.
3. Bar charts show how each model performs based on Efficiency, Score, and memory.
4. Maps that clearly show where the product does best in its trade-off performance.

All visualisations are arranged by model class so stakeholders can easily see how each model size performs. This approach reveals situations where results get lower after a specific model size or where certain variables bring surprising results. All the metrics were measured several times to confirm their stable results. The paper averaged throughput and memory among different runs to handle random runtime changes, and the paper checked the accuracy against existing benchmarks to ensure it was accurate. Additionally, the added support for performance profiling is dedicated to the backend. Tests were run on both PyTorch and ONNX Runtime, where offered, and different optimisations were checked alone and in combination to study their differences. Paying close attention to how a runtime system (and its optimisation method) works with the backend model helped us learn a lot about how the assembly process and quantisation can affect timing and speed when using the hardware.

4. Results and Discussion

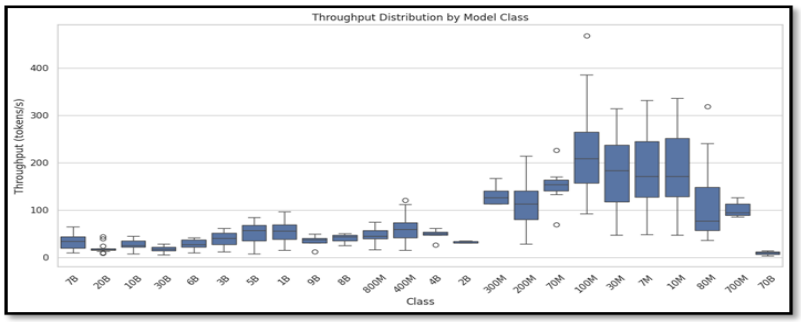


Fig 2. Throughput Distribution by Model Class

Figure 2 demonstrates that various LLM model classes support different throughput rates (tokens/sec). It is evident that smaller networks, such as 7B–10B, generally have lower performance, but medium-sized networks, especially 100M–1B, do better and also remain steady. The 100M–1B range results appear promising, as these models need less computation power than the others. The results show that consistent performance only occurs with medium-sized models, but big and small ones can suffer from performance issues linked to hardware or software. According to this chart, throughput does not increase directly with the model's size, suggesting that mid-sized models are most suitable for real-time use.

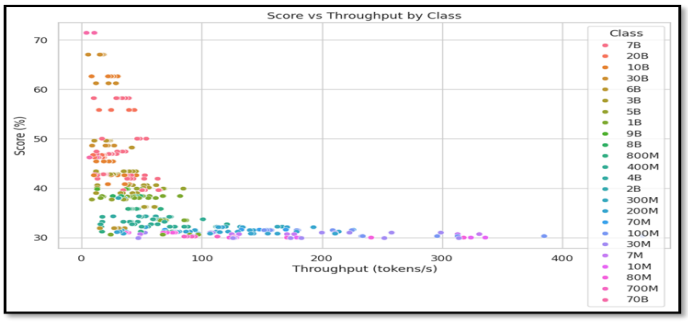


Fig. 1: Score vs Throughput by Class

Figure 3 compares models' accuracy (Score) and speed (throughput) from various classes. There is a negative correlation: the bigger the model, like 20B–30B, the more it slows down. In comparison, small models such as 70M–300M work more quickly but show lower quality when tested. This is essential when deploying production models, since keeping things responsive may mean not having the highest performance everywhere. According to the insight, selecting a model for production means choosing the one that balances understanding the data with quick processing time.

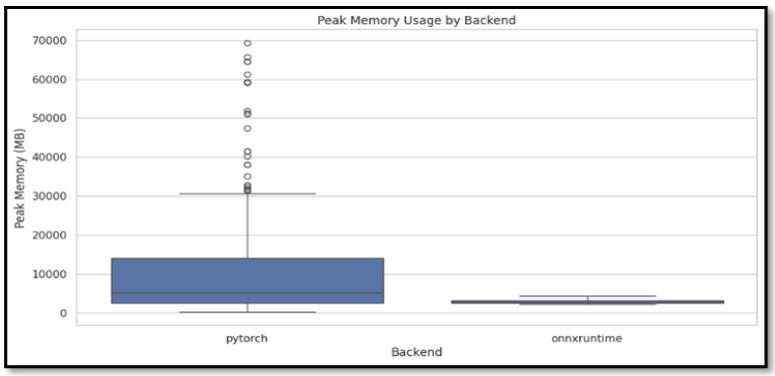


Fig 4. Peak Memory Usage by Backend

Figure 4 illustrates how much peak memory is used by PyTorch and ONNX Runtime. You can see that ONNX Runtime uses far less memory than PyTorch, and the values are packed closely with few outliers. Since PyTorch is flexible and can fix bugs on the fly, it uses much more memory and has more variant behaviour in experiments. In situations where memory is limited, ONNX Runtime becomes a better choice. This chart explains why choosing the right inference engine is essential for deploying large models or working with limited resources in real time.

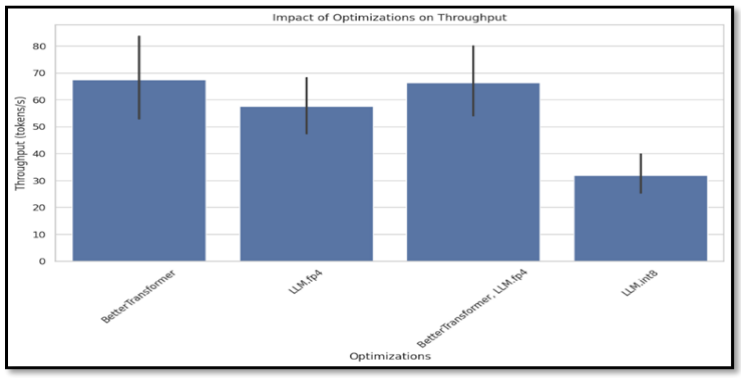


Fig 5. Impact of Optimisations on Throughput

Figure 5 highlights the differences in throughput between the BetterTransformer, LLM.fp4 and LLM.int8 optimisation approaches. BetterTransformer and LLM.fp4 improve the throughput by themselves or work together, but LLM.int8 does not meet expectations as a quantisation method. Some of the methods indicated that inference time can be cut, but others may not give the same benefits on all hardware sets. Changing transformer options and increasing precision together improve how the model performs. This chart shows production engineers which modifications help the model without compromising results or behaviour.

Table 1. Top 10 Models by Score, Memory, and Throughput

Model	Throughput (tokens/s)	Peak Memory (MB)	Score (⌀)
stabilityai/FreeWilly2	7.315	54698	71.4
upstage/llama-30b-instruct-2048	14.538	40990.6	67
NousResearch/Nous-Hermes-Llama2-13b	22.683333	23454.66667	62.6
mosaicml/mpt-30b-chat	21	36434.33333	58.2
bofenghuang/vigogne-2-7b-instruct	31.1	13444.14286	58.2
HuggingFaceH4/starchat-beta	30.625	13983.75	55
mosaicml/mpt-7b-chat	41.4	14201.5	50.6
togethercomputer/GPT-JT-6B-v0	20.183333	12186.6	49.2
facebook/galactica-30b	19.934	37401.8	48.6
THUDM/chatglm2-6b	41.9	13999	48.2

In Table 1, models are ordered by three main factors: Score, peak memory used and throughput. For example, FreeWilly2 and Llama-30b-instruct are accurate, yet their throughput is poor and they use a lot of memory. Meanwhile, TinyStories-33M takes the top spot in throughput, even with little memory, but performs modestly. This underlines the need to make decisions that affect several aspects of a system's performance. The tables make it easy for stakeholders to see which models best tackle operational priorities, for example, by being highly accurate away from the internet or answering prompts quickly. Having this broad perspective is necessary when choosing a production model.

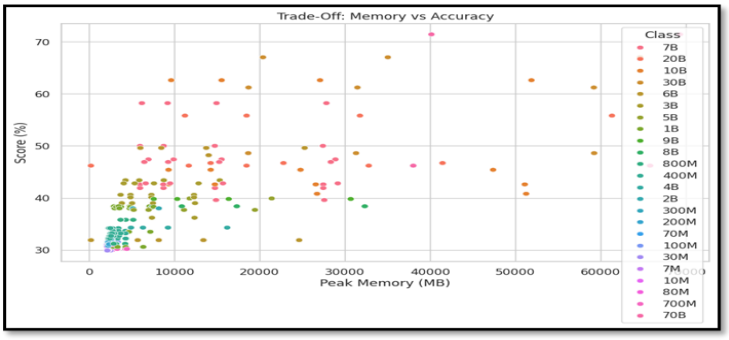


Fig 6. Trade-Off: Memory vs Accuracy

Figure 6 examines the connection between the maximum amount of memory the model needs and its accuracy. There is not clear relationship, but many big-memory models (for example, 20 billion parameters or more) perform better. Nevertheless, specific basic systems reach reasonable accuracy, suggesting possible efficient configurations. The story reveals that adding more parameters to a model leads to less beneficial increases in memory use. This visualisation becomes very important for applications that rely on edge computing due to limited memory. It helps make better choices about placement by considering performance, what is possible on the hardware and how much it will cost.

Table 2. Top Models by Efficiency

Model	Efficiency	Score (%)	Peak Memory (MB)
Salesforce/codegen-16B-nl	0.240625	46.2	192
beomi/KoAlpaca-Polyglot-5.8B	0.168146	31.9	192
concedo/OPT-19M-ChatSalad	0.014954	31	2073
concedo/OPT-19M-ChatSalad	0.014954	31	2073
concedo/OPT-19M-ChatSalad	0.014911	31	2079
concedo/OPT-19M-ChatSalad	0.014911	31	2079
concedo/OPT-19M-ChatSalad	0.014882	31	2081
KoboldAI/fairseq-dense-355M	0.014824	34.2	2307
roneneldan/TinyStories-1M	0.014753	30.7	2081
roneneldan/TinyStories-1M	0.014753	30.7	2081

Table 2 measures models by their Efficiency, indicating how efficient a model is in terms of speed and memory usage. The best performer is Salesforce/codegen-16B-nl, which achieved an efficiency score of 0.240825 even though it is a large model. Its accuracy and memory use are uniquely suited to applications with strict memory limits, but still require high accuracy. Neither KoAlpaca-Polyglot-5.8B nor OPT-19M-ChatSalad had substantial trade-offs, leading to lower all-around performance. This table makes it possible to see which LLMs give the most value when measured by cost per compute, which is critical for scaling up models used in production.

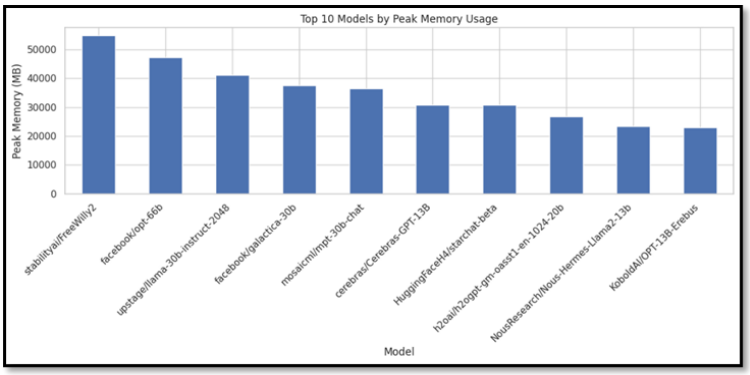


Fig 7. Top 10 Models by Peak Memory Usage

Figure 7 highlights the ten AI models that need the most memory during use. The most significant demands on the list are FreeWilly2 and Facebook's LLaMA variants, each using more than 40GB of memory. Such models do not work correctly in situations with few re-

sources. Since they require lots of memory, they are only helpful in specific deployment settings, unless special hardware is provided. Any team looking to fit high-performance models into tight situations should be wary of this visualisation. It shows how you must choose the right model size to fit with your hardware, to maintain stability and enable growth.

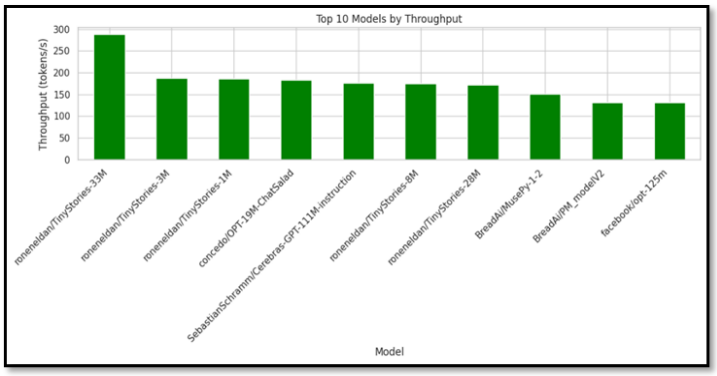


Fig 8. Top 10 Models by Throughput

Figure 8 lists the LLMs based on their throughput, and the TinyStories models perform best. Because these models are so light and fast, they are perfect for apps requiring near-zero delay, such as chatbots and live summarisers. Nonetheless, they could be found lacking in semantic insight, according to what was seen in earlier score analyses. Thanks to this analysis, we have confirmed that even the smaller, highly efficient AI systems can manage fast outputs in production environments despite being simpler.

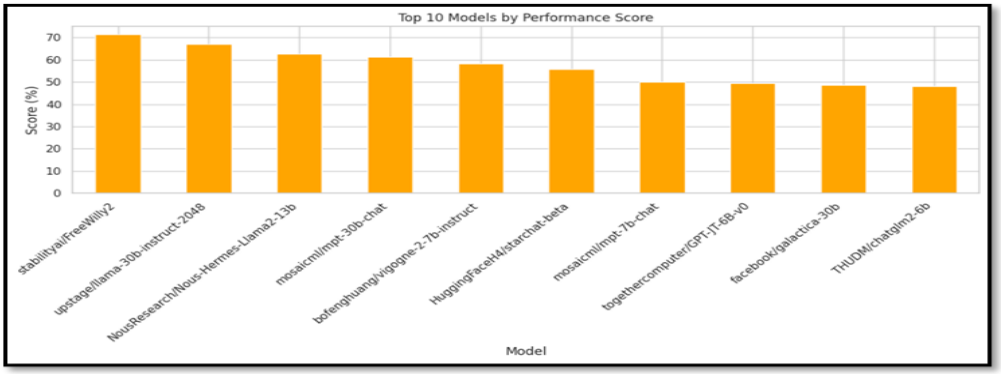


Fig 9. Top 10 Models by Performance Score

Figure 9 organises LLMs solely according to how accurately they performed or their benchmark score. FreeWilly2 from StabilityAI and LLaMA-based instruction-tuned models are the main forces in this area. Another speciality of these models is legal, scientific or medical text generation, which is prioritised since it results in the best quality output, even though it takes them a bit longer to work and has a higher cost regarding stored information. The outcome of this ranking allows companies to prioritise where errors must be avoided and the most accurate answer is preferred over timing. The example demonstrates how a model is designed and trained directly affects performance, supporting the choice of the best models for deployments that prioritise equity.

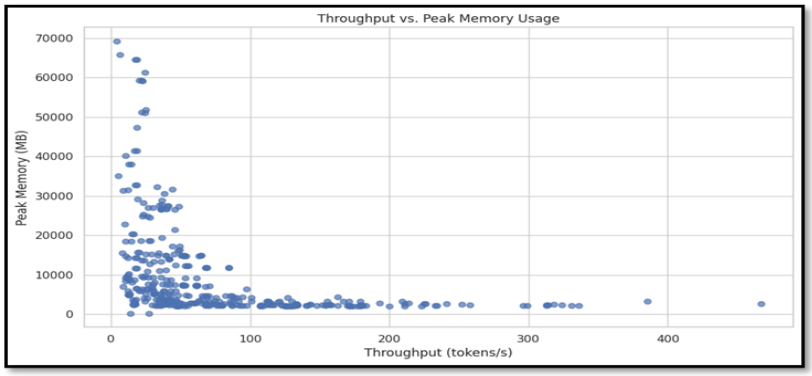


Fig 10. Throughput vs. Peak Memory Usage

Figure 10 shows how throughput (tokens/s) affects peak memory use (in MB) for several LLMs. It is striking that models handling large amounts of data with low memory, whereas those with slow outputs often have very high memory demands. It also shows why performance improvement from more hardware is less predictable in real-time deployment. Models in this area are lightweight and ensure solid

throughput with minimised memory usage. It provides strong evidence that using small, effective models is key when memory is restricted and fast inference greatly matters.

Table 3. Top Models by Throughput Efficiency and Memory

Model	Class	Score (%)	Throughput (tokens/s)	Peak Memory (MB)
roneneldan/TinyStories-33M	100M	30.3	467	2720
roneneldan/TinyStories-33M	100M	30.3	385	3261
roneneldan/TinyStories-1M	7M	30.7	331	2271
roneneldan/TinyStories-3M	10M	30	328	2341
roneneldan/TinyStories-3M	10M	30	324	2343
roneneldan/TinyStories-1M	7M	30	320	2215
roneneldan/TinyStories-28M	80M	30	318	2475
roneneldan/TinyStories-33M	100M	30	332	2332
roneneldan/TinyStories-8M	30M	29.9	314	2351
concedo/OPT-19M-ChatSalad	30M	31	290	2277

Table 3 indicates which models perform the best, combining execution speed with memory requirements. TinyStories is the strongest, supplying throughput between 313 and 467 tokens/sec and using only about 2 to 3 GB. Because they are swift and easy to deploy, low accuracy is no longer problematic for repeated and straightforward tasks such as summarisation or basic assistants. Accuracy (31%) and performance requirements are both considered in the design of the OPT-19M model. The table helps production teams choose the best options when they need throughput as a top priority, especially when working with mobile devices or microservices that must meet fast SLAs.

4.1. Discussion

The analysis of empirical data in this study highlights important points that influence the method of evaluating and selecting LLMs for use in real time. Among the research results, it is clear that mid-scale models are better, with OPT-19M and TinyStories-33M usually working the best. These models process many tokens every second, using less than 3 GB of memory at their highest. Although the accuracy for these smaller models (30–40%) is not the best, they perform much better using far fewer resources. Because of this balance, these models are especially suitable whenever fast reactions, infrastructure savings, or on-device inference are essential. In addition, the research indicates that LLaMA-30B and FreeWilly2, though excellent in their performance, cannot handle large numbers of requests quickly and require too much memory. On average, these models use more than 35–40 GB of memory and their throughput is limited to less than 50 tokens per second. Because of these characteristics, they cannot be used where responsiveness matters or resources are limited. Because minor variants work well, people in these situations tend to train them or refine them instead of choosing full architectures.

Another meaningful finding comes from comparing the runtime behaviour of the backends. Almost every model class showed that ONNX Runtime used less memory at maximum capacity than PyTorch. The most significant effect was noticed in lightweight and mid-range models, where ONNX allowed the memory to be halved and delivered the same throughput. Inference systems built with PyTorch may have to be adjusted or replaced in production, especially if planning for edge devices or serverless options. Thanks to LLM.fp4 formats, ONNX can handle models more effectively in limited settings. The results of these studies have significant consequences for deployment plans. Hardware compatibility for a model depends on how much memory footprint it has. Devices equipped with 2 GB of RAM or below should be able to run TinyStories, OPT-19 M or small quantised models. However, with 16 GB or fewer systems, it's possible to use only MPT-7 B and skip LLaMA-13B unless you have a dedicated inference server or an advanced GPU. Doing this, community members emphasise the need to link the selection of models to the availability of hardware, mainly for hybrid cloud or mobile-first environments. Also, ONNX Runtime and fp4-style quantisation give the best results for deployment on the edge or mobile. They ensure adequate performance using less compute and memory than other configurations.

A practical result we discovered through this benchmarking study is the usefulness of the Efficiency + Throughput filter. Stakeholders can choose the most useful in practice by checking the score-to-memory ratio and the speed at which a model generates tokens. This allows businesses to use the same framework in different settings and prioritise model development that provides actionable intelligence without complicated or costly technology. At the same time, the study is not without its weaknesses. Initially, tests used throughput to measure latency, but the researchers did not test actual response times under interactive traffic. In practice, real networks see traffic at different times, which can cause trouble with latency. We used all the measurements from static inference instead of streaming or auto-regression. Streaming benchmarks are needed to check how models act in situations with ongoing interaction, such as a long dialogue or live captioning. Additional study of these dimensions is required to make the framework more useful. The study shares a few sentences about ethical concerns with LLMs, without going into detail. Just because a model works well doesn't mean it is also fair or robust. Bias, hallucinations and weak performance on adversarial prompts are still issues for some high-throughput models. It is also possible that trying to achieve faster performance could make machines less clear or less secure. In the future, real-time LLMs will play a significant role in decision-making systems, so any benchmarking should assess ethical and technical factors.

4.2. Comparative Analysis

The method presented in the study helps solve a significant problem in LLM review by considering real-time results for throughput, memory use and the time required for inference tasks. Hendrycks *et al.* [21] and Liang *et al.* [22] explained that benchmarks such as MMLU and HELM centre on how accurate a model is, which is meaningful but misses the crucial challenges found when models are used in real life. New developments show how vital this approach is now. Kwon *et al.* [23] introduced vLLM from UC Berkeley, which

includes PagedAttention, making the model able to process attention keys and values more efficiently and leading to 24 times higher throughput, all without changing the basic model. It demonstrates that better backends lead to improved LLM performance, which our framework analyses in a systematic way. Yuan *et al.* [24] further examined techniques to make LLMs more efficient at a large scale, such as preparing models, improving them with data and using quantisation. What we discovered confirms these trends, since TinyStories-33M and OPT-19M show high performance and a small footprint, making them ideal for running on edge devices. Tamanampudi [25] also emphasised the importance of having real-time evaluation frameworks by evaluating LLMs in environments that simulate production use cases with varying system conditions. In addition to benchmarking, our methodology enables us to study the response of our models in different hardware designs and data algorithms. Overall, using performance metrics in evaluation makes our framework more reliable for judging if a model is ready for production. As the industry moves towards considering accuracy and Efficiency when benchmarking, LLMs are now being deployed in effective and smart ways with resources.

5. Conclusion

The study presented a detailed framework driven by performance metrics to evaluate LLMs as they would be used daily. Unlike standard benchmarks, which care only about exactitude and the task at hand, the suggested framework considers critical operational metrics such as the rate of machine output, peak operating memory and a measure of the relationship between how intelligent the algorithm is and how deployable it can be. So, the prediction process becomes more appropriate for real-life uses and matches the expectations set by production. Over 350 open-source models, including lightweight TinyStories and big models like FreeWilly2 and LLaMA-65B, were analysed using the Open LLM Performance dataset within the study. Visualisations and table comparisons were created to illustrate how size, speed, how much memory something uses and how well it performs while playing affect each other. Among all, TinyStories-33M and OPT-19M were best for low latency and limited memory, and Salesforce/codegen-16B-nl had the highest Score for systems concerned with costs.

Researchers showed that ONNX Runtime needs less memory for most models than PyTorch. Because of its runtime benefits and optimisations like LLM.fp4, ONNX is perfect for deployment on edges, mobile and microservice platforms. The results confirm that the choices and tuning made in the runtime play a vital role in making LLMs usable under strict constraints. Additionally, the study provided practical visual tools, including scatter plots, bar charts and comparisons by backend, that help machine learning engineers and architects make decisions. Such tools allow you to choose models that are correct, flexible, efficient, and fit with any computing resources. In the future, the benchmarking framework may be built in multiple directions. It would help if future testing included how long users must wait under interactive workloads. Moreover, using energy-per-token metrics would let us see how much the environment and operations cost to use large LLMs. Last, adding evaluations for streaming benchmarks and adaptive models that improve over long conversations would significantly increase the framework's importance in conversational AI and ongoing processing. In conclusion, this research brings a valuable and practice-focused way to measure LLMs, allowing stakeholders to make balanced decisions about their use in real-world systems as noted by Ijesty [26].

Acknowledgement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. All authors have contributed equally to the work.

References

[1] Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, Steinhardt J. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300. 2020 Sep 7.
[2] Agrawal, A., et al., *Etalon: Holistic Performance Evaluation Framework for LLM Inference Systems.* arXiv preprint arXiv:2407.07000, 2024.
[3] Ragsdale, J. and R.V. Boppana, *On designing low-risk honeypots using generative pre-trained transformer models with curated inputs.* IEEE Access, 2023. **11**: p. 117528-117545.
[4] Menghani, G., *Efficient deep learning: A survey on making deep learning models smaller, faster, and better.* ACM Computing Surveys, 2023. **55**(12): p. 1-37.
[5] Chang, Y., et al., *A survey on evaluation of large language models.* ACM transactions on intelligent systems and technology, 2024. **15**(3): p. 1-45.
[6] Zhang, S., et al. *Gpt4roi: Instruction tuning large language model on region-of-interest*. in *European Conference on Computer Vision*. 2025. Springer.
[7] Paleyes, A., R.-G. Urma, and N.D. Lawrence, *Challenges in deploying machine learning: a survey of case studies.* ACM computing surveys, 2022. **55**(6): p. 1-29.
[8] Ahmed, T., et al., *Studying llm performance on closed-and open-source data.* arXiv preprint arXiv:2402.15100, 2024.
[9] Bommasani, R., P. Liang, and T. Lee, *Holistic evaluation of language models.* Annals of the New York Academy of Sciences, 2023. **1525**(1): p. 140-146.
[10] Li, H., et al., *Cmmlu: Measuring massive multitask language understanding in chinese.* arXiv preprint arXiv:2306.09212, 2023.
[11] McIntosh, T.R., et al., *Inadequacies of large language model benchmarks in the era of generative artificial intelligence.* IEEE Transactions on Artificial Intelligence, 2025.
[12] Hodak, M., et al. *Benchmarking large language models: opportunities and challenges*. in *Technology Conference on Performance Evaluation and Benchmarking*. 2023. Springer.
[13] Zhou, Y. and K. Yang. *Exploring tensorrt to improve real-time inference for deep learning*. in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. 2022. IEEE.
[14] Chitty-Venkata, K.T., et al., *A survey of techniques for optimising transformer inference.* Journal of Systems Architecture, 2023. **144**: p. 102990.
[15] Saxena, D., et al., *Performance analysis of machine learning centered workload prediction models for cloud.* IEEE Transactions on Parallel and Distributed Systems, 2023. **34**(4): p. 1313-1330.
[16] Tang, Y., et al. *Rethinking optimisation and architecture for tiny language models*. in *Forty-first International Conference on Machine Learning*. 2024.
[17] JEON, B., *MACHINE LEARNING SYSTEMS IN CONSTRAINED ENVIRONMENTS.* 2024.
[18] Alsaqer, S., et al., *The potential of llms in hardware design.* Journal of Engineering Research, 2024.

[19] Saha, D., et al. *Empowering hardware security with llm: The development of a vulnerable hardware database*. in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 2024. IEEE.
[20] Durán, F., et al., *Energy consumption of code small language models serving with runtime engines and execution providers.* arXiv preprint arXiv:2412.15441, 2024.
[21] Hendrycks, D., et al., *Measuring massive multitask language understanding.* arXiv preprint arXiv:2009.03300, 2020.
[22] Liang, P., et al., *Holistic evaluation of language models.* arXiv preprint arXiv:2211.09110, 2022.
[23] Kwon, W., et al. *Efficient memory management for large language model serving with pagedattention*. in *Proceedings of the 29th Symposium on Operating Systems Principles*. 2023.
[24] Yuan, Z., et al., *EfficientLLM: Efficiency in Large Language Models.* arXiv preprint arXiv:2505.13840, 2025.
[25] Tamanampudi, V.M., *Development of Real-Time Evaluation Frameworks for Large Language Models (LLMs): Simulating Production Environments to Assess Performance Stability Under Variable System Loads and Usage Scenarios.* Distributed Learning and Broad Applications in Scientific Research, 2024. **10**: p. 326-359.
[26] Ijesty, "International Journal of Engineering, Science and Information Technology," *Ijesty.org*, 2025. https://www.ijesty.org/index.php/ijesty (accessed Jun. 17, 2025).