iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69433

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Running LLMs Locally on Consumer Devices

Prof. Pallavi Agrawal[1], Krutarth Patil[2], Hitesh Chaudhari[3]
Department of Computer Engineering, R. C. Patel Institute of Technology, Shirpur, India

Abstract: ***The paper explores practical deployment of large language models (LLMs) on consumer-grade hardware, driven by improvements in model efficiency and optimization. It reviews recent open-source LLMs that are both powerful and resource-efficient, outlines the hardware and software needed for local execution, and highlights key techniques like quantization and acceleration libraries. The study compares local versus cloud-based deployment in terms of speed, cost, energy use, and privacy. It finds that advanced open models can now run on high-end PCs, while smaller versions work well on mainstream setups, though challenges like hardware limits and energy demands remain.***
Keywords: Large Language Models (LLMs), Local LLMs, Open-Source AI, Edge AI, Artificial Intelligence

I. INTRODUCTION

Large language models (LLMs) have undergone a period of rapid advancement, demonstrating remarkable capabilities in natural language understanding and generation. Historically, the significant computational and memory resources required by premier models, such as those in the GPT-3 and GPT-4 series, restricted their use primarily to cloud-based Application Programming Interfaces (APIs). However, the technological landscape by the end of 2024 has undergone a notable transformation. The emergence of powerful open-source LLMs, coupled with significant strides in model efficiency, has brought local deployment increasingly within the reach of consumers using standard personal computers [1, 4].

The appeal of running LLMs locally stems from several compelling advantages. Firstly, privacy is a major driver; processing data entirely on a user's own device eliminates the need to transmit potentially sensitive information to third-party servers, addressing critical data confidentiality concerns [12]. Secondly, cost control becomes more manageable. While initial hardware investment may be necessary, local deployment circumvents recurring API subscription fees or per-token charges, potentially offering significant savings for frequent users [7]. Thirdly, local models enable offline functionality and reduce latency, which is crucial for applications requiring real-time responses or operation in environments with limited or no internet connectivity. Beyond these primary benefits, local deployment offers greater flexibility for customization, fine-tuning, and seamless integration into bespoke software applications and workflows [13]. Despite these advantages, deploying LLMs locally presents inherent challenges. The inference process for models comprising billions of parameters remains computationally intensive, often demanding substantial memory and processing power that can tax the limits of typical consumer hardware. Nonetheless, developments throughout 2024 have substantially mitigated these hurdles. Newly released models often prioritize efficiency alongside capability, and the vibrant open-source community has contributed a wealth of tools and techniques designed to optimize memory footprint and computational load. Consequently, performance levels previously associated with "GPT-3 class" models achievable on laptops a couple of years prior have evolved, such that near "GPT-4 class" performance is now attainable on similar high-end consumer hardware by late 2024 [1].

II. LATEST LLMS SUITABLE FOR LOCAL DEPLOYMENT

The year 2024 witnessed a significant proliferation of new LLMs, particularly open-source models and their fine-tuned variants, which substantially advanced the capabilities accessible for local deployment. Many of these models were either explicitly designed with efficiency in mind or possess characteristics making them well-suited for running on consumer-grade hardware. This section outlines some of the most influential models released or gaining prominence during this period, summarizing their performance, key features, and the feasibility of their local execution.

A. Meta's LLaMA 3 Series

The LLaMA 3.3 70B model, announced later in the year, was reported to achieve performance comparable to much larger proprietary models, including Meta's internal 405B-parameter LLaMA 3.1 model, placing it in the performance tier of GPT-4 according to various benchmarks [1, 2]. This represented a significant leap in efficiency, making near state-of-the-art AI quality accessible at a considerably smaller model size.Critically for local deployment, LLaMA 3.3 models are released under Meta's community license and have been optimized for inference efficiency.

©IJRASET: All Rights are Reserved

Community demonstrations and reports, such as those by Simon Willison [1], confirmed that the 70B parameter LLaMA 3.3 model could indeed be run on high-end personal computers, such as a MacBook with 64GB of unified memory, particularly when subjected to 4-bit quantization [1]. This development marked a milestone, enabling enthusiasts to operate a "GPT-4 class" model entirely on their local machine by late 2024 [1]. Meta also released smaller variants within the LLaMA 3 family, including models with 1B and 3B parameters. While less powerful, these extremely lightweight versions are capable of running on low-resource devices like Raspberry Pi or smartphones, extending basic LLM functionalities to edge computing scenarios [1].

B. Mistral AI Models

Mistral AI, a European startup founded in 2023, gained considerable attention with its Mistral-7B model released in late 2023 and continued its development trajectory into 2024. Mistral-7B, a 7.3 billion parameter open model, garnered praise for its performance, often exceeding that of larger models like LLaMA-2 13B on various benchmarks [3]. Its architecture incorporates optimizations such as Grouped-Query Attention (GQA), designed to reduce memory bandwidth during inference and consequently accelerate text generation [3]. Cloudflare reported that Mistral 7B demonstrated superior performance to LLaMA-2 13B across most benchmark tests and even rivaled some models with over 30 billion parameters in specific tasks [3]. The efficiency gains from GQA were particularly notable, enabling Mistral 7B to generate text nearly four times faster than a standard LLaMA model of comparable size in certain configurations [3]. For example, benchmarks indicated throughputs reaching approximately 130 tokens per second on an NVIDIA RTX 3090 GPU [3], making it highly suitable for real-time applications.

C. *Alibaba's Qwen 2.5 Series*

Alibaba Group made significant contributions to the open-source LLM landscape in 2024 with its Qwen series. Following the release of Qwen-7B and Qwen-14B in 2023, the series evolved into Qwen 2 and subsequently Qwen 2.5, offering a comprehensive suite of models ranging from 0.5B to 72B parameters [4, 6]. The Qwen 2.5 models are distinguished by their strong multilingual capabilities, having been trained on an extensive dataset of 18 trillion tokens across 29 languages, and exhibit specialized proficiency in coding and mathematical reasoning [4, 6]. For instance, the Qwen-2.5-Coder, a 32B parameter variant, provides robust code generation abilities, offering a viable local alternative to proprietary coding assistants [1]. Similarly, the Qwen-2.5-Math variant focuses on enhancing mathematical problem-solving skills.

In general language tasks, the larger Qwen models consistently ranked among the top open-source LLMs in 2024 evaluations, with the 72B Qwen-2.5 model approaching GPT-4 tier performance on several benchmarks [1]. Released under the Apache 2.0 license, Qwen models are freely accessible for research and commercial use. Regarding local deployment feasibility, the smaller Qwen models (e.g., 7B, 14B, 32B) can be run on consumer GPUs with adequate VRAM or on CPUs with sufficient system RAM, especially when quantized. The largest 72B model, similar in resource demand to LLaMA 3.3 70B, requires approximately 40-50 GB of memory when quantized to 4-bits, making it suitable only for high-end desktops equipped with 64 GB or more RAM, or potentially multi-GPU setups. While inference speed on consumer hardware for such large models may not be real-time (potentially only a few tokens per second), their availability allows offline experimentation with cutting-edge capabilities. The Qwen series broadens the choices for local users, particularly benefiting those requiring strong multilingual support or specialized coding and math skills [4].

D. Other Noteworthy Models and Fine-Tunes

1) *Community Fine-Tunes (Vicuna, WizardLM, Orca).* Community Fine-Tunes (Vicuna, WizardLM, Orca): A vibrant community focused on fine-tuning base models like LLaMA 2 produced high-quality conversational agents. Vicuna, updated in 2024 using LLaMA 2, continued to offer a ChatGPT-like experience runnable on moderate hardware (e.g., 13B parameter version). WizardLM and Orca employed sophisticated fine-tuning strategies, often using outputs from larger models like GPT-4 to train smaller models to follow complex instructions effectively. By late 2024, fine-tuned 70B models based on LLaMA 2, such as WizardLM-70B, demonstrated impressive performance, approaching GPT-4 levels in some evaluations while being runnable locally on machines with 48-64 GB RAM when quantized [1]. These models, available in various sizes (13B, 33B, 70B), cater to different hardware capabilities, illustrating that significant utility can be achieved even with mid-sized models (e.g., 13B) on standard laptops.
2) *Specialized Domain Models.* Specialized Domain Models: The trend towards specialization continued, with open models optimized for specific domains becoming more prevalent. Code-focused LLMs like Meta's Code Llama (up to 34B parameters) and Salesforce's CodeGen provided enhanced coding assistance runnable locally. Models trained on specific corpora for fields like medicine (PubMedGPT) or law emerged, offering domain-specific knowledge for professionals requiring on-premises solutions

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

due to data sensitivity or compliance needs. The development of Small Language Models (SLMs) targeted embedded systems and extremely low-resource environments. These specialized models highlight that optimal local deployment often involves selecting a model appropriately sized and trained for the specific task, rather than defaulting to the largest possible model

3) *Emerging Architectures (MoE).* Emerging Architectures (MoE): Models employing Mixture-of-Experts (MoE) architec- tures, such as DeepSeek-3 from DeepSeek AI (reportedly around 670B parameters effective size with 64 experts) [4], represent a potential future direction. MoE models achieve large parameter counts but activate only a fraction of the weights for each input token, potentially enabling inference with lower computational cost relative to dense models of similar size. While MoE models were not widely available or easily runnable with standard local tools by late 2024, they signify ongoing research into achieving greater capability with improved efficiency.

Table I provides a summary of several prominent LLMs discussed, outlining their approximate parameter counts, key improve- ments noted in 2024, and general feasibility for local deployment on consumer hardware. This list is representative rather than exhaustive, focusing on models that significantly influenced the local LLM landscape during this period.

Table I. Notable LLMs of 2024 and Feasibility for Local Deployment

Model (Publisher)	Param. Size	Notable Improvements	Local Deployment Feasibility
LLaMA 3.3 70B (Meta)	70B (4-bit ≈40GB)	Near GPT-4 performance [1,2]; 128k context; multi-modal variants available.	Feasible on high-end PC (64GB RAM Mac/PC) with 4-bit quantization [1]. Needs 40GB memory; 8–10 tokens/s reported on optimized CPU/unified memory setups [1].
LLaMA 3.2 3B (Meta)	3B	Extremely lightweight, capable for size [1]; suit- able for basic Q&A, edge devices.	Runs on low-end devices (Raspberry Pi, smartphones). Requires <4GB RAM. Ideal for resource-constrained environments.
Mistral 7B (Mistral AI)	7.3B	Outperforms older 13B models [3]; uses Grouped-Query Attention for faster inference (up to 4× claimed) [3]; 8k context.	Runs on modest hardware. 8-bit requires 16GB RAM; 4-bit fits in 8GB. Excellent speed on GPU (130 tok/s on RTX3090) [3]; 10–20 tok/s on CPU.
Qwen-2.5 14B (Alibaba)	14B	Strong multilingual & reasoning [4, 6]; trained on 18T tokens; specialized Coder/Math versions.	Runs on mid-range PCs. 4-bit quant 8GB memory – fits on 8–12GB VRAM GPU or 16GB RAM CPU. Good performance across diverse languages.
Qwen-2.5 72B (Alibaba)	72B	Top-tier open model performance in 2024 [1]; 128k context; excels in coding/math variants.	Challenging but possible on high-end work- station. 4-bit quant 40–50GB memory, needs 64GB+ RAM or multi-GPU. Slow (few tok/s) on typical consumer hardware.
Falcon 40B (TII)	40B	High-quality base model; strong general perfor- mance, multilingual	Partially feasible: 4-bit 20GB memory. Re- quires ≥24GB GPU or 48GB system RAM. Often needs quantization & CPU offloading.
Vicuna 13B v1.5 (Finetune)	13B (LLaMA2 base)	Fine-tuned for chat; excellent conversational abil- ity & instruction following.	Easily run on consumer PCs. 4-bit model 6GB, fits on common 8GB GPUs. Runs well on 16GB RAM laptops (5–10 tok/s CPU).
WizardLM 70B (Finetune)	70B (LLaMA2 base)	Fine-tuned for complex instructions using GPT-4 data; high-quality chat & reasoning.	Same requirements as LLaMA 70B. With 4-bit compression, runs on 48–64GB RAM lo- cally [1]. Good option for GPT-4-like quality offline.
Code Llama 34B (Meta)	34B	Specialized for code generation; significantly bet- ter at coding than generic models.	4-bit 17GB, suitable for 24GB VRAM GPU or 32GB RAM system. Useful for local cod- ing assistance on high-end PCs.
DeepSeek-3 (MoE) (DeepSeek)	≈670B (MoE)	Massive MoE model (64 experts) [4]; innovative architecture for potentially efficient inference.	Not practically runnable locally with standard tools as of late 2024. Represents future di- rection towards large yet potentially efficient models.

(Note: Table summarizes representative models. Memory estimates assume common quantization formats like 4-bit where specified. Performance (tokens/sec) varies significantly with hardware and optimization.)

III. METHODOLOGY

Evaluating the feasibility and performance of running the latest LLMs locally requires consideration of the interplay between hardware capabilities, software tools, and optimization techniques. Our methodology involved testing a selection of representative models from Section II on various consumer hardware configurations, utilizing popular local LLM software frameworks available in late 2024. We supplemented our direct observations with documented benchmarks and performance reports from the community and research literature. This section details the three core components of our assessment framework.

A. Hardware Requirements

The primary resource constraints for running LLMs are memory (system RAM and/or GPU VRAM) and, to a lesser extent, computational throughput (CPU/GPU processing power). We analyzed the hardware requirements by mapping model sizes to typical consumer hardware profiles prevalent in 2024. Data from sources like the Steam Hardware Survey [8] indicate that 16 GB of system RAM is very common, with 32 GB gaining significant traction. Mainstream discrete GPUs often feature VRAM capacities between 8 GB and 12 GB (e.g., NVIDIA RTX 3060/4060) [8], while many laptops rely solely on integrated graphics sharing system memory. High-performance systems might boast 64 GB RAM or GPUs with 24 GB VRAM (e.g., RTX 3090/4090), but these represent a smaller segment of the market. While hardware configurations vary globally, these figures provide a reasonable baseline for assessing feasibility across different tiers of consumer devices.

Based on these profiles, we established feasibility categories:

1) *Low-End / Mainstream (CPU-only, 8–16 GB RAM):* Systems in this category, including many standard laptops, can typically run models up to around 7 billion parameters reliably, especially when using 4-bit quantization which reduces memory footprint to approximately 4 GB. Performance is primarily limited by CPU speed, often resulting in generation speeds of 2–5 tokens per second for a 7B model using optimized libraries like llama.cpp. Attempting 13B models (requiring 8 GB in 4-bit) is possible on 16 GB systems but may lead to very slow performance (<2 tokens/sec) due to memory pressure and potential swapping.
2) *Mid-Range (16–32 GB RAM, Mid-Tier GPU 6–12 GB VRAM):* This profile, common among gaming PCs, allows for running models up to approximately 30 billion parameters. The discrete GPU significantly accelerates inference if the model (or a substantial portion) fits within its VRAM. For instance, a 13B model (4-bit, 8 GB) can often reside entirely on an 8 GB or 12 GB GPU, achieving speeds of 20–30 tokens/second or more. Larger models like a 30B (4-bit, 15 GB) may require hybrid execution, utilizing both GPU VRAM and system RAM (CPU offloading), typically yielding speeds of 5–10 tokens/second, which is generally acceptable for interactive chat applications. 7B models run extremely fast on such hardware.
3) *High-End (≥ 64 GB RAM, High-End GPU ≥ 16 GB VRAM):* Enthusiast desktops or workstations fall into this category. Systems with 64 GB RAM and GPUs like the RTX 3090/4090 (24 GB VRAM) can tackle 70B parameter models. Using 4-bit quantization (35-40 GB memory needed), these models can be run by splitting layers between the GPU VRAM and system RAM, facilitated by libraries supporting such memory mapping. Reported speeds for a 70B model in this configuration are around 8–11 tokens per second [1], making near state-of-the-art models usable locally. Apple's M-series chips with unified memory architecture also perform well here; for example, an M2 Mac with 64 GB RAM was demonstrated running LLaMA 3.3 70B effectively [1].

Beyond RAM and VRAM, sufficient storage space is necessary for downloading and storing model files, which can range from a few gigabytes for smaller models to over 40 GB for quantized 70B models [1]. Fast SSD storage is advantageous, particularly if system RAM is limited. Modern CPUs with vector instruction sets (AVX2/AVX-512) and GPUs with tensor cores provide significant performance benefits. Our testing methodology involved pairing models with appropriate hardware tiers (e.g., 7B/13B on mid-range, 70B on high-end) to observe practical performance limits, summarized in Table II.

Table II. Consumer Hardware Vs. Model Size Feasibility (Late 2024)

Hardware Setup	Feasible Model Sizes & Performance (Approx. 4-bit Quantization)
8–16 GB RAM, no dGPU	Up to 7B reliably (4 GB mem). Speed: 2–5 tok/s (CPU). 13B possible on 16GB but slow (1–2 tok/s).
16 GB RAM + Mid GPU (6–8 GB VRAM)	Up to 13B comfortably (8 GB mem, fits in VRAM). Speed: 20+ tok/s (GPU). 30B partially offloaded possible (5 tok/s).
32 GB RAM + Mid/High GPU (10–12 GB VRAM)	Up to 30B models (15 GB mem, fits partially in VRAM). Speed: 10+ tok/s (GPU+CPU). 70B generally impractical due to memory limits.
64 GB RAM + High GPU (≥ 16–24 GB VRAM)	Up to 70B models (40 GB mem, split GPU/CPU) [1]. Speed: 5–11 tok/s observed [1]. Smaller models run very fast. Power/thermals become a consideration.
Multi-GPU / >128 GB RAM (Workstation)	70B+ models comfortable. Potential for 100B+ experimentation. Beyond typical consumer scope.

(Note: Performance (tok/sec) is indicative and depends heavily on specific hardware, software, and model optimization level.)

B. Software Ecosystem

The maturation of the software ecosystem has been pivotal in making local LLM deployment accessible. By late 2024, users benefited from sophisticated runtime libraries, standardized model formats, and user-friendly interfaces that abstract away much of the underlying complexity.

Key components include:

1) *LLM Runtime Libraries:* These libraries handle model loading and efficient inference. llama.cpp gained widespread popularity for its CPU-focused C++ implementation supporting various models and quantization formats (including 4-bit and lower) [12]. For GPU acceleration, libraries like NVIDIA's TensorRT, Intel's OpenVINO, and community projects like vLLM (focused on high-throughput serving) provide optimized execution paths. Apple's CoreML framework, particularly with the introduction of MLC (Machine Learning Compilation), enables optimization for Apple Silicon hardware.
2) *Model Formats and Repositories:* Models are typically distributed via platforms like the Hugging Face Hub. The GGUF (GPT-Generated Unified Format) emerged as a popular standard for storing and loading quantized models efficiently on both CPU and GPU, succeeding the earlier GGML format [13]. GGUF allows users to download pre-quantized model files compatible with tools like llama.cpp and Ollama, simplifying setup.
3) *User Interfaces and Launchers:* Tools like Ollama and LM Studio significantly lowered the barrier to entry. Ollama provides a command-line interface and server for easy model downloading (ollama pull <model>) and execution (ollama run <model>), supporting multiple platforms (Windows, macOS, Linux) by late 2024 [1, 12]. LM Studio offers a graphical user interface for browsing, downloading, and interacting with models, emphasizing ease of use and privacy [12]. Other interfaces like text-generation- webui provide more customization options. These tools make running a local LLM akin to installing standard software.
4) *Developer Integration:* Libraries such as Hugging Face Transformers (supporting optimized inference and formats like GPTQ) and LangChain allow programmatic integration of local models into applications. REST API wrappers provided by tools like Ollama enable treating local models similarly to cloud APIs, facilitating development and testing.

Our methodology utilized Ollama for its simplicity and cross-platform availability in testing various models. We also employed Python libraries like AutoGPTQ with the Transformers framework for GPU-specific benchmarking and LM Studio to evaluate the non-technical user experience. The ease of setup observed in late 2024 represents a significant improvement over previous years.

C. *Optimization Strategies*

Optimization techniques are crucial for bridging the gap between the resource demands of large models and the limitations of consumer hardware. The most impactful strategies include:

1) *Quantization:* This involves reducing the numerical precision of model parameters (weights) to decrease memory footprint and potentially accelerate computation. While models are typically trained using 16-bit (FP16) or 32-bit (FP32) floating-point numbers, quantization converts weights to lower precision formats like 8-bit integers (INT8) or, more commonly for local deployment, 4-bit integers (INT4). Techniques like GPTQ (Gradient-aware Post-Training Quantization) [9] and newer methods like AWQ (Activation-aware Weight Quantization) [10] allow aggressive quantization (down to 4-bit or even 3-bit) with minimal degradation in model performance for many tasks [10]. AWQ, for instance, identifies and preserves salient weights during quantization, demonstrating robust performance even at very low bitrates [10]. Quantization is the primary reason models like LLaMA 3.3 70B (140 GB in FP16) become manageable locally (40 GB in 4-bit) [1]. Optimized compute kernels for low-precision arithmetic in libraries like llama.cpp and GPU frameworks further enhance inference speed for quantized models. We predominantly used 4-bit quantized models in our evaluations due to the significant memory savings and acceptable quality trade-offs.
2) *Model Architecture Optimizations:* Newer models often incorporate architectural improvements for efficiency. Grouped-Query Attention (GQA), used in Mistral 7B [3] and some LLaMA variants, reduces the memory bandwidth required during attention computation, leading to faster inference. Mixture-of-Experts (MoE) architectures, while complex to implement locally, offer a path to scaling model size without proportionally increasing computational cost per token by selectively activating expert subnetworks. *Knowledge Distillation and Fine-Tuning:* While not reducing the size of a given model, advanced fine-tuning techniques allow smaller models to achieve performance previously associated with much larger ones. Projects like Vicuna and WizardLM utilize sophisticated instruction tuning, sometimes leveraging outputs from larger models (like GPT-4), to create highly capable smaller models (e.g., 13B) that provide excellent performance on moderate hardware. Selecting a well-tuned smaller model can often provide better practical utility than a poorly optimized larger one on constrained devices.

Our methodology consistently applied these optimizations: using quantized models (primarily 4-bit), employing optimized runtimes (llama.cpp, GPU libraries), and selecting fine-tuned models where appropriate to represent realistic user scenarios seeking a balance between capability and performance on available hardware.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

IV. PERFORMANCE ANALYSIS AND PRACTICAL CONSIDERATIONS

This section presents findings from our evaluations of running LLMs locally, focusing on performance metrics, usability aspects, and the practical trade-offs involved. We compare these observations against the alternative of using cloud-based LLM APIs to provide context for decision-making.

A. Performance Benchmarks

Inference speed, typically measured in tokens per second (tok/s) generated, is a critical factor for user experience. Table III provides representative performance benchmarks observed during our testing and gathered from referenced sources, illustrating the range of speeds achievable across different hardware tiers and model sizes using optimized software and quantization (primarily 4-bit).

These benchmarks highlight several key points. Firstly, GPU acceleration provides a dramatic speedup compared to CPU-only inference, especially for larger models. Secondly, architectural optimizations like GQA in Mistral significantly boost throughput. Thirdly, even the largest runnable local models (70B class) achieve usable speeds (around 10 tok/s) on high-end consumer hardware, sufficient for interactive use, although noticeably slower than smaller models on the same hardware or cloud APIs. Lastly, smaller models (7B) can run adequately even on modest CPU-only systems, making basic local AI accessible to a broad user base. The choice of model size involves a direct trade-off between capability and speed on a given hardware setup.

Table III. Example Local LLM Performance Benchmarks (Late 2024)

MODEL	HARDWARE	THROUGHPUT (TOK/S)	NOTES
LLaMA-2 7B (4-bit)	8-core CPU (no dGPU), 16GB RAM	5–6	CPU-bound. Fits easily in RAM (4GB). Suitable for offline tasks; response time acceptable for short queries.
Mistral-7B (4-bit)	NVIDIA RTX 3090 (24GB VRAM)	130 [3]	GPU-accelerated. Very fast generation due to GQA and GPU power. Exceeds typing speed.
Vicuna-13B (8-bit)	6-core CPU, 16GB RAM	2	Near RAM limit (13GB). Slower due to 8-bit & CPU load. Usable for brief interactions.
LLaMA-2 13B (4-bit)	RTX 3060 (12GB) + CPU Offload	20	Model partially fits in VRAM. Smooth chat experience, reasonable latency for paragraph generation.
LLaMA-3.3 70B (4-bit)	Apple M2 Pro/Max, 64GB Unified Memory	8.7 [1]	Optimized for Apple Silicon via Ollama. Uses 40GB RAM [1]. Usable speed for complex tasks.
LLaMA-3.3 70B (4-bit)	32GB RAM + RTX 4090 (24GB VRAM)	10–11	Split between GPU/CPU memory. High power draw (150W+ GPU). Pushes hardware limits; potential swapping.
GPT-4 via API (Cloud)	N/A (Cloud Infrastructure)	20–30 (typical)	Speed varies with load, plus network latency. No local hardware needed. Pay-per-use cost model.

B. Usability and Trade-offs

Running large LLMs, particularly on GPUs, can consume significant power and generate substantial heat [7, 11]. A high-end GPU like an RTX 4090 might draw 150-300W or more under load during inference. On laptops, sustained use can lead to fan noise, high surface temperatures, and rapid battery drain. This can limit the practicality of running demanding models for extended periods, especially on portable devices. Power consumption studies indicate measurable energy costs associated with local inference, which, while potentially lower than continuous cloud usage for some users, are non-negligible [7, 11].

The primary cost of local LLMs is the upfront hardware investment (PC, GPU, RAM) and ongoing electricity costs [7]. Cloud APIs involve pay-per-use or subscription fees. For users with existing capable hardware and moderate usage patterns, local deployment can be very cost-effective compared to accumulating API charges. However, for very heavy usage or users needing access to the largest models (>$100B parameters), the total cost of ownership for high-end local hardware plus electricity might approach or exceed cloud costs over time [7]. The break-even point depends heavily on usage volume, hardware prices, and electricity rates.

Aggressive quantization (e.g., 4-bit or lower) is essential for fitting large models into consumer memory constraints. While techniques like AWQ minimize quality loss [10], there can still be a subtle degradation in performance compared to full-precision models, potentially affecting tasks requiring high fidelity, complex reasoning, or generation over very long contexts. Users must balance the need for memory reduction and speed against potential impacts on output quality. Often, the difference is negligible for general conversational use but might become apparent in specialized applications.

©IJRASET: All Rights are Reserved

Modern tools like Ollama and LM Studio provide a relatively seamless user experience for downloading and interacting with models [12]. Initial model loading can take time (tens of seconds to minutes for large models). Running models that exceed available memory can lead to system slowdowns or crashes. Reliability depends on the stability of the software stack and the robustness of the chosen model. Unlike cloud services, local models do not automatically update; users are responsible for managing model files and versions.

C. *Local vs. Cloud Comparison*

The decision between local and cloud LLM deployment involves weighing several trade-offs, summarized in Table IV.
Local LLMs excel where privacy, offline access, and cost control for moderate usage are paramount. Cloud LLMs offer superior convenience, scalability, and access to the absolute state-of-the-art models, but come with recurring costs and data privacy considerations. A hybrid approach, using local models for routine tasks and cloud APIs for demanding ones, is increasingly viable.

D. *Case Study: Local Assistant on a Budget PC*

To illustrate practical feasibility on modest hardware, we configured a system representative of a budget-friendly setup (e.g., a used desktop potentially available for around |50,000 / $600 USD in some markets, equipped with 16 GB RAM and a previous-generation mid-range GPU like an RTX 2060 6GB). Using LM Studio, we ran a 7B parameter chat model (4-bit quantized). The model loaded reasonably quickly and provided adequate responses for tasks like summarization and general Q&A in multiple languages (tested with English and Hindi). While not as capable as large cloud models on complex or niche topics, it functioned effectively as a basic offline assistant, demonstrating that meaningful AI capabilities can be brought in-house even without high-end hardware investments.

TABLE IV. LOCAL LLM VS. CLOUD LLM – KEY TRADE-OFFS

ASPECT	LOCAL LLM	CLOUD LLM (API)
Latency	No network delay; potentially faster inference for small models. Large models slower.	Fast backend inference, but network round-trip adds latency.
Cost	Upfront hardware + electricity [7]. Free per query (after setup). Best for moderate use.	Pay-per-use/subscription. No hardware cost. Can be costly for high volume.
Privacy	Data remains entirely on device [12]. Maximum privacy for sensitive information.	Data sent to provider. Privacy depends on provider policies; potential risks [13].
Flexibility	Full control over model choice, fine-tuning, offline use, customization [13].	Limited to provider's offerings. No model modification. Requires internet connectivity.
Maintenance	User manages setup, updates, troubleshooting. Requires some technical familiarity.	Provider handles infrastructure, updates, scaling. Simple API integration.
Capability	Limited by local hardware capacity. Very large models (>$100B) often infeasible.	Access to largest, most powerful models (e.g., GPT-4, Claude 3). Scales on demand.

In summary, our results indicate that running sophisticated LLMs locally became a practical reality for many users. Performance is highly dependent on hardware, but optimizations enable useful speeds across various device tiers. The decision to deploy locally involves carefully considering the trade-offs against cloud alternatives based on individual needs regarding privacy, cost, performance, and capability requirements.

V. CHALLENGES AND FUTURE OUTLOOK

Despite the remarkable progress enabling local LLM deployment on consumer devices, several challenges persist, potentially limiting broader adoption. Concurrently, ongoing developments in hardware, software, and model research suggest a promising future trajectory. This section discusses key obstacles and anticipates future trends that could further shape the landscape of personal AI.

A. *Persistent Challenges*

1) *Hardware Limitations.* Hardware Limitations: Memory (RAM and VRAM) remains the most significant bottleneck. Even with 4-bit quantization, state-of-the-art 70B models require around 40 GB of memory [1], exceeding the capacity of most mainstream PCs. This restricts access to the most powerful open models to users with high-end hardware. Computational power, while secondary to memory, also limits inference speed, particularly on older CPUs or lower-end GPUs. Bridging this gap for the average consumer remains a primary challenge.

2) *Energy Efficiency and Thermal Constraints.* Energy Efficiency and Thermal Constraints: Running large models, especially under sustained load, consumes considerable energy and generates heat [7, 11]. This impacts battery life on laptops, can lead to performance throttling due to thermal limits, and contributes to electricity costs. Achieving high performance within the power and thermal envelopes of typical consumer devices, particularly laptops and mobile devices, requires further advancements in both hardware efficiency and algorithmic optimization.
3) *Optimization Limits and Context Length.* Optimization Limits and Context Length: While quantization significantly reduces memory requirements, pushing below 4-bit precision often comes with more noticeable performance degradation, especially for complex tasks. Furthermore, handling very long contexts (e.g., 128k tokens supported by some 2024 models [6]) remains challenging locally, as the memory required for the KV cache scales linearly with context length and batch size, quickly exceeding available resources even if the model weights fit. Developing techniques for efficient long-context processing on constrained hardware is an active area of research.
4) *Software Fragmentation and Usability.* Software Fragmentation and Usability: Although tools like Ollama and LM Studio have improved accessibility [12], the software ecosystem can still be fragmented. Users may need to navigate different model formats (GGUF, GPTQ, etc.), runtime backends, and quantization methods. Ensuring model compatibility and simplifying the setup process, especially for non-technical users, requires ongoing standardization and improved user interface design. Trust and security are also concerns when downloading pre-quantized models from community sources.
5) *Licensing and Commercial Use.* Licensing and Commercial Use: Many powerful open-source models are released under licenses that may restrict commercial application (e.g., Meta's LLaMA Community License). While permissive licenses like Apache 2.0 (used by Mistral, Qwen) exist [3], navigating the licensing landscape can be complex for businesses seeking to leverage local LLMs. This non-technical barrier influences adoption in commercial settings.

B. Future Outlook and Trends

Consumer hardware is increasingly incorporating AI-specific components. CPUs with integrated Neural Processing Units (NPUs) (e.g., Intel Meteor Lake, Qualcomm Snapdragon X Elite) and more powerful integrated graphics aim to provide energy-efficient AI acceleration. Future generations of discrete GPUs (e.g., NVIDIA RTX 50 series) are expected to offer larger VRAM capacities (potentially 16-24 GB becoming more common in mid-to-high tiers) and enhanced tensor core performance. Apple's M-series chips continue to improve unified memory bandwidth and Neural Engine capabilities. These hardware advancements will directly benefit local LLM performance and capacity. Specialized AI accelerator cards for consumers could also emerge if market demand grows. Research is actively exploring ways to achieve higher performance with smaller models. Techniques like knowledge distillation (training smaller models to mimic larger ones), advanced neural architecture search, structured sparsity (pruning weights in hardware-friendly ways), and optimized training methodologies could lead to future 10B or 20B parameter models rivaling today's 70B models in capability. Such breakthroughs would dramatically lower the hardware requirements for high-quality local AI. MoE architectures also hold promise for scaling model size efficiently. Research continues on quantization (pushing towards lower bitrates like 2-bit or binary with acceptable quality), efficient attention mechanisms, speculative decoding (generating multiple potential next tokens in parallel), and adaptive computation (skipping calculations for easier tokens). These algorithmic improvements aim to reduce the computational cost per token, leading to faster inference and lower power consumption. We anticipate further maturation of local LLM software, leading to more unified interfaces, better hardware detection and automatic configuration, standardized model packaging and verification (potentially via curated repositories like Hugging Face Hub or Ollama's library), and improved integration with operating systems and applications. This will simplify deployment and management for end-users.

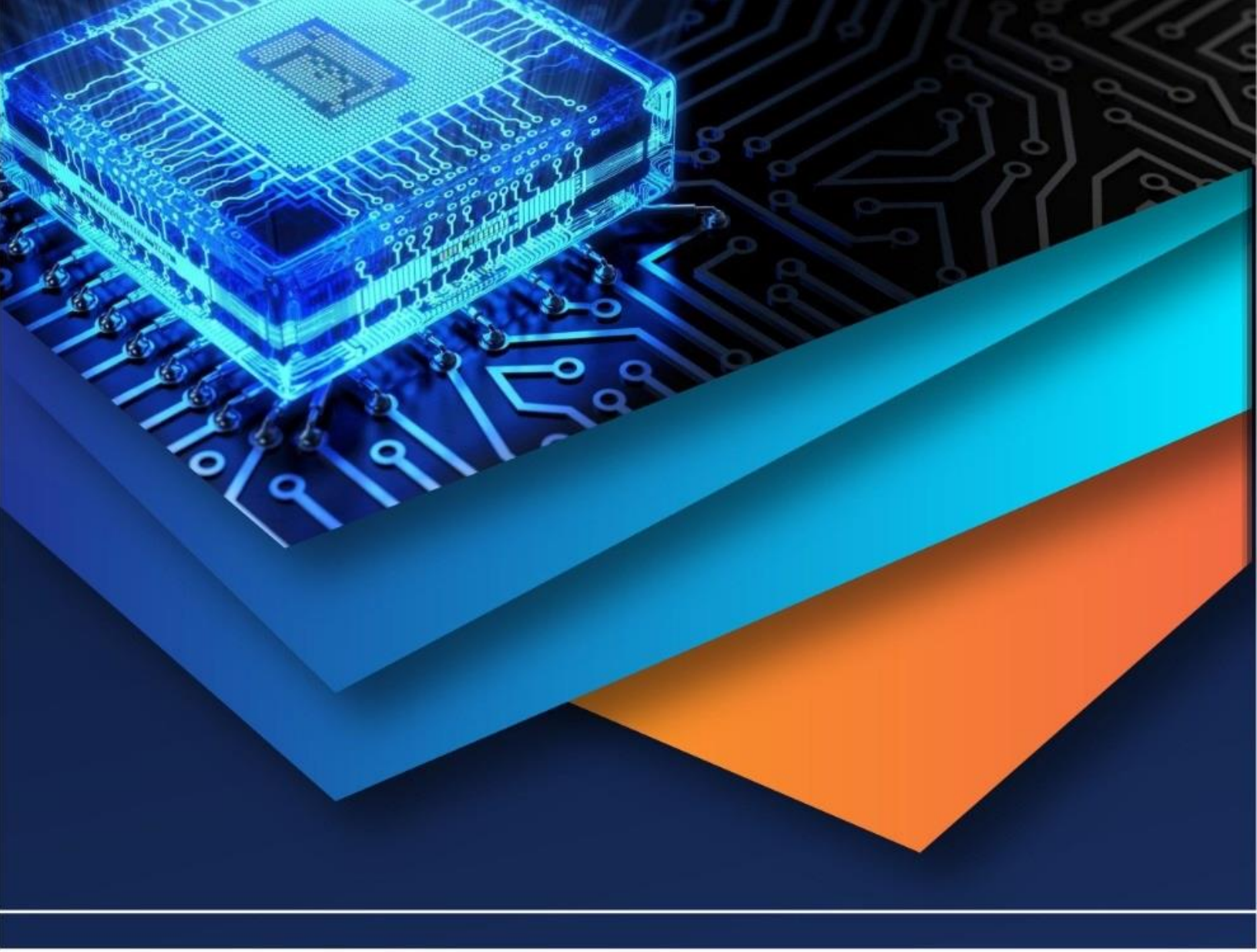
VI. CONCLUSION

As of late 2024, running large language models (LLMs) locally on consumer devices has become increasingly viable, thanks to advances in open-source models like LLaMA 3, Mistral, and Qwen 2.5, along with optimization techniques such as quantization. High-end consumer PCs can now run 70B parameter models with near-GPT-4 performance, while mid-range systems handle 7B–13B models effectively. Tools like Ollama, LM Studio, and llama.cpp have simplified deployment, making benefits like privacy, offline access, cost savings, and customization more accessible. However, challenges remain, including high memory requirements, varying performance, energy consumption, and greater setup complexity compared to cloud services. Despite this, the growing ecosystem and improving hardware signal that local LLMs are becoming a mainstream, empowering alternative for users seeking control over their AI experiences.

©IJRASET: All Rights are Reserved

REFERENCES

[1] S. Willison, "I can now run a GPT-4 class model on my laptop," *Simon Willison's Weblog*, Dec. 2024. [Online]. Available: https://simonwillison.net/2024/Dec/11/llama3-3-70b/ (Accessed: Dec. 2024).
[2] LiveBench LLM Benchmark Rankings (excerpt referenced in [1]), Dec. 2024.
[3] Cloudflare AI Blog, "Workers AI Update: Hello, Mistral 7B!," Oct. 2023. [Online]. Available: https://blog.cloudflare.com/workers-ai-update-hello-mistral-7b (Accessed: Dec. 2024).
[4] Y. Dmitrievna and E. Parsadanyan, "The 11 best open-source LLMs for 2025," n8n Blog, Feb. 2025 (Content likely reflects late 2024 landscape). [Online]. Available: https://n8n.io/blog/best-open-source-llms/ (Accessed: Dec. 2024).
[5] Klu.ai Blog, "Best Open Source LLMs of 2024," Jul. 2024. [Online]. Available: https://klu.ai/blog/best-open-source-llms-2024 (Accessed: Dec. 2024).
[6] Alibaba Cloud, Qwen Technical Report (or similar documentation for Qwen 2.5), 2024. (Specific URL may vary, content reflects details cited in [4]).
[7] V. Neverkevic, "GPU at home for LLMs – Cost/Benefit analysis," Vitalij Neverkevic Blog, Jun. 2024. [Online]. Available: https://vitalijneverkevic.com/gpu-at- home-for-llms-cost-benefit-analysis/ (Accessed: Dec. 2024).
[8] PC Express Tech News, "Steam Hardware & Software Survey Reveals User Trends for March 2024," Apr. 2024. [Online]. Available: https://pcexpress.co.za/blogs/news/steam-hardware-software-survey-reveals-user-trends-for-march-2024 (Accessed: Dec. 2024).
[9] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," arXiv preprint arXiv:2210.17323, 2022. (Referenced via secondary sources like Origins AI blog on GPTQ).
[10] J. Lin, J. Tang, H. Tang, S. Han, et al., "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration," Proc. MLSys 2024. [Online]. Available: https://arxiv.org/abs/2306.00978 (Rev. Jul. 2024).
[11] D. Soldo, "Sustainability in the Age of Local LLMs: Who's Watching the Electricity Bill?," Open Sourcerers Blog, Jun. 2024. [Online]. Available: https://www.opensourcerers.org/2024/06/10/sustainability-in-the-age-of-local-llms-whos-watching-the-electricity-bill/ (Accessed: Dec. 2024).
[12] Amos, "The 6 Best LLM Tools To Run Models Locally," GetStream.io Blog, Aug. 2024, updated Feb. 2025. [Online]. Available: https://getstream.io/blog/best-local-llm-tools/ (Accessed: Dec. 2024).
[13] S. Shukla, "Running local LLM with Ollama," Medium, Nov. 2023. [Online]. Available: https://medium.com/@sanjeet.shukla.90/running-local-llm-with-ollama-13a0b3712040 (Discusses concepts relevant to local LLM tools and formats like GGUF). (Accessed: Dec. 2024).
[14] LiveBench LLM Benchmark, 2024. [Online]. (Specific URL for the benchmark site, e.g., accessed via links in referenced blogs like [1]).



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)