# **Assessing Hardware Requirements for Running Large Language Models**

**1. Introduction: The Growing Need for Local LLM Execution and Hardware Assessment**

The landscape of artificial intelligence is rapidly evolving, with Large Language Models (LLMs) at the forefront of this transformation. These sophisticated models, capable of understanding and generating human-like text, are finding applications across diverse fields. Increasingly, there is a growing interest among individuals and organizations to run these powerful LLMs on their local hardware 1. This desire stems from several compelling factors, including the need for enhanced privacy and security by keeping sensitive data on-premise, the ability to access and utilize AI capabilities even without a stable internet connection, the potential for greater customization and fine-tuning of models to specific needs, and the long-term cost savings compared to relying on cloud-based services for continuous usage 1. The increasing availability of open-source LLMs and user-friendly tools further facilitates this trend towards local execution, making advanced AI technology more accessible to a wider audience 3.

However, even the smaller LLMs possess substantial hardware demands, particularly concerning the memory resources, namely Random Access Memory (RAM) and Graphics Processing Unit Video RAM (VRAM), as well as the processing capabilities of both the Central Processing Unit (CPU) and the Graphics Processing Unit (GPU) 1. Understanding these intricate hardware requirements is paramount to ensure that the chosen LLM can be executed smoothly and efficiently on a given system, without encountering frustrating errors or significant performance bottlenecks that render the model unusable 5. Determining whether a specific LLM, characterized by its size (number of parameters) and precision (quantization level), is compatible with the available hardware is a key challenge for many users eager to explore the potential of local AI. Clear and actionable guidance is therefore essential to navigate this complexity and empower users to make informed decisions. This report aims to address this need by providing a comprehensive and practical methodology for users to assess the compatibility of an LLM with their hardware, based on the model's specifications and their system's capabilities.

**2. Understanding the Core Hardware Components for LLMs**

Running LLMs locally requires a combination of hardware components working in concert. Each component plays a distinct yet crucial role in enabling the model to function effectively.

The **Central Processing Unit (CPU)** acts as the brain of the computer system, responsible for managing overall system performance 1. In the context of LLMs, the CPU plays a vital role in tasks such as data preprocessing, handling input and output operations, and coordinating parallel computations between different hardware units 1. For efficient execution, it is generally advisable to have a multi-core processor with high clock speeds to handle these diverse tasks effectively 4. For dedicated LLM servers, server-grade CPUs like Intel Xeon or AMD EPYC are often recommended due to their robust features, including a large number of PCI-Express lanes for connecting multiple GPUs and high-speed storage, substantial memory bandwidth and capacity, and support for Error-Correcting Code (ECC) memory, which is crucial for maintaining data integrity during intensive computations 6. However, for running smaller LLMs or handling lighter workloads, consumer-grade CPUs such as Intel Core i9 or AMD Ryzen 9 can provide solid performance with a good balance of speed and cost 4. While the GPU is the primary engine for LLM computations, a capable CPU remains essential for supporting auxiliary tasks and ensuring overall system responsiveness.

The **Graphics Processing Unit (GPU)** is arguably the most critical hardware component for running LLMs, as it provides the massive parallel processing power required for the intensive matrix multiplications and other computations inherent in these models during both training and inference 1. Within the GPU, the **Video RAM (VRAM)** is of paramount importance for running LLMs locally 1. The model's weights and the intermediate data (activations) generated during processing are primarily stored in the GPU's VRAM 4. If the available VRAM is insufficient to hold the entire model and the necessary data, the system may attempt to offload some of this information to the system's main RAM, which can lead to significant performance degradation due to the slower data transfer speeds between RAM and the GPU 1. In severe cases of VRAM inadequacy, the LLM might fail to load or run altogether, resulting in errors 1. NVIDIA GPUs are generally the preferred choice for running LLMs due to their well-established CUDA (Compute Unified Device Architecture) support and broad compatibility with popular deep learning frameworks like TensorFlow and PyTorch 2. While AMD GPUs offer competitive performance and support the HIP/ROCm frameworks, the ecosystem and software support are currently more extensive for NVIDIA 4. The recommended VRAM capacity is directly related to the size of the LLM, measured by its number of parameters 8. For instance, running a 7B (7 billion parameter) model typically requires 8-16 GB of VRAM, while larger models like 13B might need 16-24 GB, 30B could require 24-48 GB, and models with 65B or more parameters often demand 48 GB or more of VRAM 8. Examples of GPUs commonly recommended for running LLMs include the NVIDIA RTX 3060 (with 12 GB VRAM), the higher-end RTX 3090 and RTX 4090 (both with 24 GB VRAM), professional-grade options like the NVIDIA RTX 6000 Ada, L40S, and H100, as well as AMD's Radeon RX series GPUs 1. Beyond VRAM capacity, the number of CUDA cores (in NVIDIA GPUs) and the memory bandwidth also play a crucial role in the speed and efficiency of LLM computations 1.

**Random Access Memory (RAM)** serves as another critical component when working with LLMs 1. Large language models necessitate substantial RAM during both the training and inference phases to hold model parameters and the data being processed, especially in scenarios where the GPU's VRAM is insufficient and data needs to be temporarily moved to system memory 1. An inadequate amount of RAM can lead to memory errors or a significant slowdown in performance as the system resorts to swapping data between the RAM and the much slower hard drive 4. The recommended RAM capacity varies depending on the size of the LLM being used. For smaller models around 7B parameters, a minimum of 16-32 GB of RAM is often suggested, while 13B models might require 32-64 GB 8. For running larger models and handling extensive datasets, 64 GB of DDR4 or DDR5 RAM is considered ideal, and for very large models (30B+) or large-scale fine-tuning tasks, 128 GB or more might be necessary 4. Notably, NVIDIA recommends having at least twice the amount of CPU system memory as the total GPU VRAM to ensure efficient buffering and data transfer between the CPU and GPU 6. Therefore, RAM acts as a vital secondary memory resource that helps prevent severe performance bottlenecks, particularly when dealing with larger models or situations where VRAM is limited.

Finally, the **Storage** subsystem, ideally using Solid State Drives (SSDs), especially NVMe (Non-Volatile Memory Express) SSDs, plays a crucial role in the overall experience of running LLMs locally 4. Fast storage is essential for quickly loading the large LLM files and the associated datasets into the system's memory 4. NVMe SSDs offer significantly faster read and write speeds compared to traditional Hard Disk Drives (HDDs), which can substantially reduce the time required to load these large files 4. A minimum of 1 TB of NVMe SSD storage is generally recommended, but for users working with multiple large models or extensive datasets, 2 TB or more might be preferable 4. High-performance SSDs are particularly beneficial for scenarios involving frequent access to large model files 4. The disk space required to store the LLM itself can vary considerably based on the model's size and the precision of its weights, ranging from approximately 10-20 GB for a 7B parameter model to over 200 GB for models with 65B or more parameters 8. While storage capacity is important for housing the LLM files, the speed of the storage directly impacts the time it takes to make the model ready for use.

**3. Decoding LLM Characteristics and Their Hardware Implications**

To determine if a specific LLM can run on your hardware, it is essential to understand the key characteristics of the model and how they translate into hardware demands. The two primary characteristics to consider are the number of parameters and the quantization level.

The **number of parameters** in an LLM serves as a fundamental measure of its size and the complexity of the patterns it can learn from data 8. These parameters, essentially the model's learned weights and biases, dictate how it interprets input and generates output 14. Generally, a model with a larger number of parameters possesses a greater capacity for understanding nuances in language and generating more coherent and contextually relevant text, often leading to improved performance on various natural language processing tasks 11. However, this increased capability comes at the cost of significantly higher memory (both VRAM and RAM) and computational requirements 11. Therefore, the parameter count of an LLM is the primary factor to consider when assessing its compatibility with your hardware.

**Quantization** is a crucial technique employed to reduce the memory footprint and potentially accelerate the inference speed of LLMs 3. This process involves converting the model's weights and sometimes its activations from high-precision data types, such as 32-bit floating point (FP32) or 16-bit floating point (FP16/BF16), to lower-precision formats like 8-bit integers (INT8) or even 4-bit integers (INT4) 3. The benefit of using lower precision is that each parameter requires fewer bytes of storage in memory 13: FP32 uses 4 bytes per parameter, FP16/BF16 uses 2 bytes, INT8 uses 1 byte, and INT4 uses just 0.5 bytes 13. Consequently, quantization can lead to a significant reduction in both VRAM and RAM requirements, making it possible to run larger models on hardware with more limited resources 3. For example, a 7B parameter model quantized to 4-bit might only need 4-6 GB of VRAM, whereas its full-precision (FP32) version would require around 28 GB 8. However, it's important to note that there is often a trade-off between the level of quantization and the model's performance, with lower precision potentially leading to some degradation in accuracy 13. Advanced quantization techniques are continuously being developed to minimize this performance loss 24. Tools like llama.cpp and Hugging Face's bitsandbytes library provide functionalities to work with and load quantized versions of LLMs 8. Understanding the quantization level of the LLM you intend to run is therefore crucial for accurately assessing its hardware requirements.

To provide a quick reference, the following table summarizes the approximate VRAM and RAM requirements for LLM inference based on different model sizes and precision levels, drawing from various sources 8:

| **Model Size (Parameters)** | **Precision** | **Approximate VRAM Requirement** | **Approximate RAM Requirement** |
| --- | --- | --- | --- |
| 7B | FP32 | ~28 GB | ~28 GB |
| 7B | FP16/BF16 | ~14 GB | ~14 GB |
| 7B | INT8 | ~7 GB | ~7 GB |
| 7B | INT4 | ~3.5 GB | ~3.5 GB |
| 13B | FP32 | ~52 GB | ~52 GB |
| 13B | FP16/BF16 | ~26 GB | ~26 GB |
| 13B | INT8 | ~13 GB | ~13 GB |
| 13B | INT4 | ~6.5 GB | ~6.5 GB |
| 30B | FP32 | ~120 GB | ~120 GB |
| 30B | FP16/BF16 | ~60 GB | ~60 GB |
| 30B | INT8 | ~30 GB | ~30 GB |
| 30B | INT4 | ~15 GB | ~15 GB |
| 65B+ | FP32 | ~260+ GB | ~260+ GB |
| 65B+ | FP16/BF16 | ~130+ GB | ~130+ GB |
| 65B+ | INT8 | ~65+ GB | ~65+ GB |
| 65B+ | INT4 | ~32.5+ GB | ~32.5+ GB |

**4. Step-by-Step Guide to Assessing LLM Hardware Compatibility**

To effectively determine if a given LLM can run on your hardware, follow these steps:

**Step 1: Obtaining Your System Specifications:** The first crucial step is to have a clear understanding of your computer's hardware capabilities 4. Identify the key components that are most relevant for running LLMs: the type of your CPU, the model and VRAM capacity of your GPU, and the total capacity of your system's RAM 4. In Windows, you can easily find this information by right-clicking on the taskbar, selecting "Task Manager," and then navigating to the "Performance" tab 1. Here, you can see details about your CPU and GPU. Clicking on "GPU" will display information about your graphics card, and clicking on your GPU's name (e.g., "GPU 0") will show detailed specifications, including the dedicated VRAM capacity 1. Similarly, the "Memory" section will show your total RAM capacity. Also, note the type of your storage drive (e.g., SSD, NVMe SSD) as this will impact loading times. Similar methods exist for obtaining this information on other operating systems. Accurate knowledge of your hardware is the essential foundation for the subsequent steps in assessing LLM compatibility.

**Step 2: Estimating LLM Resource Requirements:** Once you know your system's specifications, the next step is to estimate the resource requirements of the specific LLM you are interested in running. Begin by determining the number of parameters of the LLM. This information is typically available in the model's documentation, on the model card on platforms like Hugging Face, or in related research papers 19. Next, identify the quantization level of the model you plan to use (e.g., FP32, FP16, INT8, INT4). This information is usually provided alongside the model files or in the model's description 19. With the parameter count and quantization level known, you can now estimate the VRAM and RAM requirements for inference. You can use the table provided in the previous section as a quick reference. Alternatively, you can use the general rules of thumb: approximately 2GB of VRAM per 1 billion parameters for FP16 precision, 1GB for INT8, and 0.5GB for INT4. Keep in mind that these are general estimations, and actual usage can vary slightly depending on the model's architecture and implementation. It's also important to consider potential overhead for the KV (Key-Value) cache and other runtime operations, which can add to the memory footprint 20. A common formula to estimate VRAM requirements in Gigabytes, including a 20% overhead, is: M = (P x (Q/8)) x 1.2, where 'P' is the number of parameters in billions and 'Q' is the number of bits used for quantization (e.g., 32 for FP32, 16 for FP16, 8 for INT8, 4 for INT4) 22. Remember that while this report focuses on inference, training an LLM from scratch or even fine-tuning requires significantly more resources than simply running it for generating text 13.

**Step 3: Comparing LLM Requirements with System Specs:** The final step involves a direct comparison between the estimated resource requirements of the LLM and the specifications of your hardware. Compare the estimated VRAM requirement of the LLM with the total VRAM capacity of your GPU. If the LLM's VRAM needs are greater than the VRAM available on your GPU, you are likely to encounter issues such as slow performance due to offloading to system RAM or the model failing to run altogether 1. Similarly, compare the estimated RAM requirement with the total RAM installed in your system. If the LLM requires more RAM than you have, it can also lead to performance problems and instability. Ensure that you also have enough free disk space on a fast SSD or NVMe drive to store the LLM files, which can be quite large 4. While the GPU, particularly its VRAM, is usually the primary bottleneck for LLM inference, a significantly underpowered CPU could still negatively impact the overall experience, especially during the initial loading of the model and the handling of input and output data 1. If the estimated requirements are close to your hardware limits, it's advisable to look for benchmarks or user reports of that specific LLM running on similar hardware to get a more realistic expectation of performance.

**5. Leveraging LLM Hardware Calculators and Benchmarks**

To further simplify the process of assessing LLM hardware compatibility, several online tools and resources are available.

**LLM Hardware Requirement Calculators** are web-based tools specifically designed to estimate the hardware resources needed to run LLMs 22. These calculators typically allow users to input key parameters of the LLM they are interested in, such as the model name (often linking to a pre-populated parameter count), the number of parameters, the desired quantization level, the context length, and sometimes even KV cache settings 22. Based on these inputs, the calculator provides an estimate of the required VRAM, the minimum recommended system RAM, the approximate on-disk model size, and in some cases, the number of GPUs that might be necessary 46. Examples of such calculators include the LLM Inference Hardware Calculator 46, LLM Tools 47, LLM RAM Calculator 48, GPU Poor49?, LLM Model VRAM Calculator (available on SillyTavern and Hugging Face Spaces) 52, LLM vRAM Estimator 53, 🤗 Model Memory Calculator 56, and the GPU Memory Requirement Calculator for AI Models 55. These tools can offer a more user-friendly and often more accurate estimation compared to manual calculations based on general rules of thumb.

Beyond estimations, **understanding benchmark data** can provide valuable real-world insights into the performance of specific LLMs on various hardware configurations 43. Resources like the llama.cpp Vulkan Scoreboard offer community-driven benchmark results for different LLMs (often quantized) running on a wide range of GPUs 58. Additionally, numerous articles and YouTube videos compare the performance of LLMs on popular consumer GPUs such as the NVIDIA RTX 3060, RTX 3090, and others, often providing metrics like tokens generated per second (t/s) 59. When reviewing benchmark data, it is crucial to pay attention to the specific model being tested, its quantization level, and the hardware configuration used, including the GPU model and VRAM capacity, as these factors will significantly influence the reported performance. By examining benchmarks for the LLM you are interested in and comparing the tested hardware to your own system, you can gain a more realistic expectation of the inference speed and overall performance you might achieve.

**6. Addressing Potential Bottlenecks and Performance Considerations**

When attempting to run LLMs locally, it is important to be aware of potential bottlenecks and factors that can impact performance.

Running an LLM on hardware with **insufficient VRAM** is a common bottleneck 1. If the VRAM capacity of your GPU is less than the LLM's requirements, the system will likely try to offload parts of the model and the processing to the system's RAM. However, the data transfer speeds between VRAM and RAM are significantly faster than between RAM and the hard drive (used for swapping), resulting in a substantial slowdown in inference speed, potentially making the model too slow for practical use 1. In more severe cases where the VRAM and even the RAM are insufficient, the LLM might fail to load or run at all, leading to memory-related errors 4. While the GPU is typically the primary performance limiter, a weak CPU can also create bottlenecks, especially during the initial loading of the model into memory, during the preprocessing of input data, and potentially if the offloading between GPU and CPU is not handled efficiently 1. Additionally, using slow storage, such as a traditional HDD instead of an SSD, can significantly increase the time it takes to load the large LLM files into memory, impacting the overall user experience 4.

Fortunately, several **strategies exist for optimizing LLM execution** even on hardware with certain limitations 2. One of the most effective techniques is **quantization**, where using lower-precision versions of the LLM (like INT8 or INT4) can dramatically reduce the memory requirements, allowing you to run larger models on GPUs with less VRAM 3. Tools like LM Studio and llama.cpp offer the ability for **GPU layer offloading**, which allows users to specify how many layers of the LLM are processed on the GPU (utilizing VRAM) while the remaining layers reside in the system RAM 2. This hybrid approach can provide a significant performance improvement over running the entire model on the CPU, although the overall speed will depend on the amount of available VRAM. Reducing the **batch size** (the number of input sequences processed at once) and the **context length** (the maximum length of the input sequence the model can handle) can also help to decrease VRAM usage during inference 12. Furthermore, specialized **memory optimization frameworks** like vLLM are being developed to minimize memory waste and increase the efficiency of LLM serving 5. Finally, it is always an option to consider a **hybrid approach**, using local LLMs for specific tasks or for processing privacy-sensitive data, and falling back to more powerful cloud-based services for tasks that require larger models or higher computational resources 7.

**7. Conclusion: Empowering Users to Make Informed Decisions About Local LLM Deployment**

In conclusion, determining whether a specific Large Language Model can run on your local hardware involves a careful assessment of both the model's characteristics and your system's capabilities. Key considerations include the model's number of parameters, its quantization level, and the specifications of your CPU, GPU (especially its VRAM capacity), and RAM. Following a structured approach, starting with obtaining your system specifications and then estimating the LLM's resource requirements, is essential for making an informed decision. Leveraging the growing number of online LLM hardware requirement calculators and exploring benchmark data for similar hardware configurations can further enhance the accuracy of your assessment.

It is important to remember that running LLMs locally often involves a balance between the desired model size and performance and the limitations of your available hardware. While insufficient hardware, particularly VRAM, can lead to significant performance bottlenecks, various optimization techniques, such as quantization and GPU layer offloading, can help mitigate these issues and potentially enable you to run models that might otherwise be too demanding. By understanding the interplay between LLM characteristics and hardware requirements, and by utilizing the available tools and optimization strategies, users can make well-informed decisions and unlock the exciting potential of running LLMs on their own machines.

#### Works cited

1. How to Run an LLM Locally with Pieces, accessed March 19, 2025, <https://pieces.app/blog/how-to-run-an-llm-locally-with-pieces>
2. Tech Primer: What hardware do you need to run a local LLM? | Puget Systems, accessed March 19, 2025, <https://www.pugetsystems.com/labs/articles/tech-primer-what-hardware-do-you-need-to-run-a-local-llm/>
3. How to run an LLM locally on your PC in less than 10 minutes - The Register, accessed March 19, 2025, <https://www.theregister.com/2024/03/17/ai_pc_local_llm/>
4. Recommended Hardware for Running LLMs Locally - GeeksforGeeks, accessed March 19, 2025, <https://www.geeksforgeeks.org/recommended-hardware-for-running-llms-locally/>
5. Understanding VRAM and How Much Your LLM Needs - RunPod Blog, accessed March 19, 2025, <https://blog.runpod.io/understanding-vram-and-how-much-your-llm-needs/>
6. Hardware Recommendations for Large Language Model Servers - Puget Systems, accessed March 19, 2025, <https://www.pugetsystems.com/solutions/ai-and-hpc-workstations/ai-large-language-models/hardware-recommendations/>
7. What are the Hardware Requirements for Large Language Model (LLM) Training?, accessed March 19, 2025, <https://www.appypie.com/blog/hardware-requirements-for-llm-training>
8. Hardware requirements for running the large language model Deepseek R1 locally., accessed March 19, 2025, <https://www.rnfinity.com/news-show/Hardware-requirements-for-running-large-language-model-Deepseek-R1-on-a-local-machine>
9. Running LLMs on your computer locally — focus on the hardware! | by Michael McAnally, accessed March 19, 2025, <https://michael-mcanally.medium.com/running-llms-on-your-computer-locally-75717bd38d5e>
10. PSA: Local LLM Hardware Requirements : r/homeassistant - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/homeassistant/comments/1hovutx/psa_local_llm_hardware_requirements/>
11. www.hyperstack.cloud, accessed March 19, 2025, <https://www.hyperstack.cloud/blog/case-study/how-much-vram-do-you-need-for-llms#:~:text=The%20number%20of%20parameters%20directly,is%20important%20to%20prevent%20bottlenecks.>
12. How Much VRAM Do You Need for LLMs? - Hyperstack, accessed March 19, 2025, <https://www.hyperstack.cloud/blog/case-study/how-much-vram-do-you-need-for-llms>
13. Some basic knowledge of LLM: Parameters and Memory Estimation | by Baicen Xiao, accessed March 19, 2025, <https://medium.com/@baicenxiao/some-basic-knowledge-of-llm-parameters-and-memory-estimation-b25c713c3bd8>
14. Understanding Large Language Model Parameters and Memory ..., accessed March 19, 2025, <https://www.unite.ai/understanding-large-language-model-parameters-and-memory-requirements-a-deep-dive/>
15. The Definitive Guide to LLM Parameters and Model Evaluation - Galileo AI, accessed March 19, 2025, <https://www.galileo.ai/blog/llm-parameters-model-evaluation>
16. LLM Model Parameter & Memory Required for Training and Inference - Medium, accessed March 19, 2025, <https://medium.com/@plthiyagu/llm-model-parameter-memory-required-for-training-and-inference-634963b36b59>
17. How Much Memory Needed for LLM | dasarpAI, accessed March 19, 2025, <https://dasarpai.com/dsblog/how-much-memory-needed-for-llm>
18. I'm an LLM idiot confused by all the options and not knowing how to find a model that fits with my local hardware. I had GPT provide some info. Any smart people here wanna fact check or sign off? : r/SillyTavernAI - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/SillyTavernAI/comments/1j9jkck/im_an_llm_idiot_confused_by_all_the_options_and/>
19. Simple Guide to Calculating VRAM Requirements for Local LLMs - TWM, accessed March 19, 2025, <https://twm.me/posts/calculate-vram-requirements-local-llms/>
20. GPU memory requirements for serving Large Language Models - UnfoldAI, accessed March 19, 2025, <https://unfoldai.com/gpu-memory-requirements-for-llms/>
21. GPU Memory Essentials for AI Performance | NVIDIA Technical Blog, accessed March 19, 2025, <https://developer.nvidia.com/blog/gpu-memory-essentials-for-ai-performance/>
22. How much VRAM do I need for LLM inference? | Modal Blog, accessed March 19, 2025, <https://modal.com/blog/how-much-vram-need-inference>
23. Optimizing LLMs for Speed and Memory - Hugging Face, accessed March 19, 2025, <https://huggingface.co/docs/transformers/main/llm_tutorial_optimization>
24. Quantization in LLMs: Why Does It Matter? - Dataiku blog, accessed March 19, 2025, <https://blog.dataiku.com/quantization-in-llms-why-does-it-matter>
25. How to Estimate Memory Requirements for Running Local LLMs - Msty, accessed March 19, 2025, <https://msty.app/blog/llm-memory-requirement>
26. What is Quantization? | IBM, accessed March 19, 2025, <https://www.ibm.com/think/topics/quantization>
27. A Comprehensive Study on Quantization Techniques for Large Language Models - arXiv, accessed March 19, 2025, <https://arxiv.org/html/2411.02530v1>
28. Ultimate Guide to LLM Quantization for Faster, Leaner AI Models - Lamatic Labs, accessed March 19, 2025, <https://blog.lamatic.ai/guides/llm-quantization/>
29. A Guide to Quantization in LLMs | Symbl.ai, accessed March 19, 2025, <https://symbl.ai/developers/blog/a-guide-to-quantization-in-llms/>
30. Exploring quantization in Large Language Models (LLMs): Concepts and techniques | by Karthikeyan Dhanakotti | Data Science at Microsoft | Medium, accessed March 19, 2025, <https://medium.com/data-science-at-microsoft/exploring-quantization-in-large-language-models-llms-concepts-and-techniques-4e513ebf50ee>
31. What's 4-bit quantization? How does it help Llama2 - Kaggle, accessed March 19, 2025, <https://www.kaggle.com/code/lorentzyeung/what-s-4-bit-quantization-how-does-it-help-llama2>
32. QLoRA and 4-bit Quantization - Chris McCormick, accessed March 19, 2025, <https://mccormickml.com/2024/09/14/qlora-and-4bit-quantization/>
33. Unsloth - Dynamic 4-bit Quantization - Unsloth AI, accessed March 19, 2025, <https://unsloth.ai/blog/dynamic-4bit>
34. Lossless 4-bit quantization for large models, are we there? : r/LocalLLaMA - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1h0aev6/lossless_4bit_quantization_for_large_models_are/>
35. [2310.16836] LLM-FP4: 4-Bit Floating-Point Quantized Transformers - arXiv, accessed March 19, 2025, <https://arxiv.org/abs/2310.16836>
36. Quantized 8-bit LLM training and inference using bitsandbytes on AMD GPUs - ROCm Blogs, accessed March 19, 2025, <https://rocm.blogs.amd.com/artificial-intelligence/bnb-8bit/README.html>
37. [vLLM — Quantization] bitsandbytes: 8-bit Optimizers, LLM.int8(), QLoRA, and k-bit Inference Scaling Laws | by Don Moon | Byte-Sized AI | Medium, accessed March 19, 2025, <https://medium.com/byte-sized-ai/vllm-quantization-bitsandbytes-efaec31f00df>
38. LLM.int8() - Hugging Face, accessed March 19, 2025, <https://huggingface.co/docs/bitsandbytes/reference/nn/linear8bit>
39. Shrinking Elephants: A Funny Guide to 4-bit and 8-bit Quantization for LLMs with LoRA, accessed March 19, 2025, <https://medium.com/@shikharstruck/shrinking-elephants-a-funny-guide-to-4-bit-and-8-bit-quantization-for-llms-with-lora-ddf9f1a62070>
40. How to shrink Large Language Models. Quantization and 1-bit LLMs explained - Medium, accessed March 19, 2025, <https://medium.com/@madalina.lupu.d/how-to-shrink-large-language-models-quantization-and-1-bit-llms-explained-c50c76df0b42>
41. LLM Quantization: Techniques, Advantages, and Models - TensorOps, accessed March 19, 2025, <https://www.tensorops.ai/post/what-are-quantized-llms>
42. LLM Series - Quantization Overview | by Abonia Sojasingarayar - Medium, accessed March 19, 2025, <https://medium.com/@abonia/llm-series-quantization-overview-1b37c560946b>
43. From no GPU to a 3060 12gb, what can I run? : r/LocalLLaMA - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1765g70/from_no_gpu_to_a_3060_12gb_what_can_i_run/>
44. The Ultimate Guide to Hardware Requirements for Training and Fine-Tuning Large Language Models (LLMs) - Towards AI, accessed March 19, 2025, <https://pub.towardsai.net/the-ultimate-guide-to-hardware-requirements-for-training-and-fine-tuning-large-language-models-7b5fe3884f64>
45. How much VRAM do I need for LLM model fine-tuning? | Modal Blog, accessed March 19, 2025, <https://modal.com/blog/how-much-vram-need-fine-tuning>
46. alexziskind1/llm-inference-calculator - GitHub, accessed March 19, 2025, <https://github.com/alexziskind1/llm-inference-calculator>
47. manuelescobar-dev/LLM-Tools: Open-source calculator for LLM system requirements. - GitHub, accessed March 19, 2025, <https://github.com/manuelescobar-dev/LLM-Tools>
48. LLM RAM Calculator - Estimate LLM Memory Requirements, accessed March 19, 2025, <https://llm-calc.rayfernando.ai/>
49. LLM check - Rahul Chand, accessed March 19, 2025, <https://rahulschand.github.io/gpu_poor/>
50. VRAM calculator for LLMs : r/LocalLLaMA - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1baq8ub/vram_calculator_for_llms/>
51. [D] How to calculate VRAM usage of a LLM model during fine-tuning? - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/MachineLearning/comments/1flnxtg/d_how_to_calculate_vram_usage_of_a_llm_model/>
52. LLM Model VRAM Calculator - SillyTavern, accessed March 19, 2025, <https://sillytavernai.com/llm-model-vram-calculator/>
53. LLM vRAM Estimator - smcleod.net, accessed March 19, 2025, <https://smcleod.net/vram-estimator/>
54. LLM Model VRAM Calculator - a Hugging Face Space by NyxKrage, accessed March 19, 2025, <https://huggingface.co/spaces/NyxKrage/LLM-Model-VRAM-Calculator>
55. GPU Memory Requirement Calculator for AI Models - AI Product Engineer, accessed March 19, 2025, <https://aiproduct.engineer/tutorials/how-much-gpu-memory-is-required-for-running-the-model>
56. Model memory estimator - Hugging Face, accessed March 19, 2025, <https://huggingface.co/docs/accelerate/usage_guides/model_size_estimator>
57. Model Memory Utility - a Hugging Face Space by hf-accelerate, accessed March 19, 2025, <https://huggingface.co/spaces/hf-accelerate/model-memory-usage>
58. Performance of llama.cpp with Vulkan #10879 - GitHub, accessed March 19, 2025, <https://github.com/ggml-org/llama.cpp/discussions/10879>
59. Performance Analysis: Running LLMs on Ollama with an RTX 3060 Ti GPU Server, accessed March 19, 2025, <https://www.databasemart.com/blog/ollama-gpu-benchmark-rtx3060ti>
60. RTX 3090 vs RTX 3060: inference comparison : r/LocalLLaMA - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1augktf/rtx_3090_vs_rtx_3060_inference_comparison/>
61. LLM Inference Speeds, accessed March 19, 2025, <https://dmatora.github.io/LLM-inference-speed-benchmarks/>
62. Run Llama 2 70B on Your GPU with ExLlamaV2 | Towards Data Science, accessed March 19, 2025, <https://towardsdatascience.com/run-llama-2-70b-on-your-gpu-with-exllamav2-588141a88598/>
63. Mistral 7B System Requirements: What You Need to Run It Locally, accessed March 19, 2025, <https://www.oneclickitsolution.com/centerofexcellence/aiml/run-mistral-7b-locally-hardware-software-specs>
64. RTX 3060 vs RTX 4090: Which GPU is Best for AI Applications? - Toolify AI, accessed March 19, 2025, <https://www.toolify.ai/ai-news/rtx-3060-vs-rtx-4090-which-gpu-is-best-for-ai-applications-955694>
65. RTX 3060 vs RTX 3090: LLM Performance on 7B, 14B, 32B, 70B Models - YouTube, accessed March 19, 2025, <https://www.youtube.com/watch?v=VGyKwi9Rfhk>
66. The Local LLM Reality Check: What Actually Happens When You Try to Run AI Models on Your Computer - Cline Blog, accessed March 19, 2025, <https://cline.bot/blog/the-local-llm-reality-check-what-actually-happens-when-you-try-to-run-ai-models-on-your-computer>
67. Exploring GPU Performance Across LLM Sizes - Puget Systems, accessed March 19, 2025, <https://www.pugetsystems.com/labs/articles/exploring-gpu-performance-across-llm-sizes/>
68. What llms can I run on my rtx 3060 12gb vram for the coding and generative ai purposes : r/LocalLLaMA - Reddit, accessed March 19, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1fxhsgr/what_llms_can_i_run_on_my_rtx_3060_12gb_vram_for/>
69. Accelerate Larger LLMs Locally on RTX With LM Studio | NVIDIA Blog, accessed March 19, 2025, <https://blogs.nvidia.com/blog/ai-decoded-lm-studio/>
70. Top NVIDIA GPUs for LLM Inference | by Bijit Ghosh | Medium, accessed March 19, 2025, <https://medium.com/@bijit211987/top-nvidia-gpus-for-llm-inference-8a5316184a10>
71. Running Local LLMs, CPU vs. GPU - a Quick Speed Test - DEV Community, accessed March 19, 2025, <https://dev.to/maximsaplin/running-local-llms-cpu-vs-gpu-a-quick-speed-test-2cjn>
72. 6 Best Consumer GPUs For Local LLMs and AI Software in Late 2024 - YouTube, accessed March 19, 2025, <https://www.youtube.com/watch?v=Bi0NGT2E7nE>
73. vRAM Requirements for LLM Fine-Tuning | by Dzmitry Ashkinadze | Medium, accessed March 19, 2025, <https://medium.com/@dzmitry.ashkinadze/vram-requirements-for-llm-fine-tuning-ec35e42240d8>