Estimating the Energy Footprint of Locally Executed Artificial Intelligence Models

Section 1: Introduction to Local AI Energy Footprinting

1.1 The Rise of Local AI and Its Energy Imperative

The landscape of artificial intelligence (AI) is undergoing a significant transformation, characterized by a growing trend towards executing sophisticated models directly on local, consumer-grade hardware. This shift, driven by compelling advantages such as enhanced data privacy, reduced latency for interactive applications, reliable offline accessibility, and the burgeoning open-source AI movement, empowers users to run complex tasks like large language model (LLM) inference and generative image creation without constant reliance on cloud infrastructure. The proliferation of powerful CPUs, GPUs, and increasingly, NPUs in personal computers and laptops has been a key enabler of this democratization.

However, this decentralization of AI computation introduces an often-overlooked consequence: energy consumption. While the focus frequently rests on the immense power demands of large-scale data centers training and serving AI models (where a single AI data center's consumption can rival that of 100,000 homes), the cumulative energy footprint of numerous individual users running AI locally can become substantial. Each local AI task, from generating a paragraph of text to creating a detailed image, consumes electrical power, contributing to overall energy demand and, depending on the electricity grid's carbon intensity, associated greenhouse gas emissions. This report aims to address this emerging concern by focusing on methods to estimate this "footprint," primarily quantifying energy consumption in kilowatt-hours (kWh) and instantaneous power demand in Watts (W), with a secondary consideration for translating these figures into carbon emissions, typically measured in grams of CO2 equivalent (gCO2eq).

The increasing ease with which users can deploy and run AI models locally, facilitated by user-friendly tools and frameworks like Ollama, may inadvertently mask these energy costs. Unlike cloud-based AI services where computational resource usage is often explicitly billed, the energy consumed by local AI tasks is absorbed into a user's general electricity bill, making its specific impact less transparent. This lack of direct cost visibility, coupled with the rapid adoption and potentially frequent execution of local AI, underscores the importance of developing energy literacy within this domain. As AI models continue to grow in size and complexity, their local execution will invariably demand more power. Therefore, understanding and quantifying this "hidden cost" is crucial not only for individual users seeking to manage their energy use but also for the broader goal of promoting sustainable and responsible AI practices. This report

endeavors to provide the foundational knowledge and methodologies necessary to illuminate these energy footprints.

1.2 Objectives and Scope of the Report

The primary objective of this report is to provide a comprehensive framework for estimating the energy footprint of AI models executed on local hardware. This entails several key aims:

- To develop and present methodologies for energy estimation, ranging from simplified approaches for quick assessments to more detailed techniques for greater accuracy.
- To gather, synthesize, and present empirical data regarding the power consumption characteristics of relevant hardware components (GPUs, CPUs, NPUs) and the performance of common AI tasks (LLM inference, image generation).
- To offer practical guidance and best practices for end-users wishing to measure the energy footprint of their specific local AI workloads and systems.

The scope of this investigation is broad, encompassing the critical factors that influence local AI energy consumption. These include:

- **Hardware Dependencies:** Analyzing the power profiles of diverse processing units from major manufacturers (NVIDIA, AMD, Intel, Apple), considering both discrete components and integrated solutions, as well as system-level power aspects.
- **AI Model Characteristics:** Examining the impact of model size (parameter count), architectural differences (e.g., Transformer, Mixture-of-Experts), and quantization techniques (e.g., FP16, INT8, 4-bit GGUF) on energy draw and performance.
- **Software and Frameworks:** Investigating how the choice of inference engines (e.g., Ollama, llama.cpp, TensorRT-LLM), UI frontends (e.g., ComfyUI, Automatic1111), and underlying drivers can affect efficiency.
- **Workload and Utilization Patterns:** Differentiating between task types, the influence of task duration and intensity, and the often-neglected impact of idle power consumption.
- **Measurement Techniques:** Reviewing software and hardware tools available for monitoring power consumption.

The ultimate goal is to provide a robust, evidence-based resource that enables technical project leads, AI/ML researchers, and software developers to make more informed decisions regarding the energy implications of local AI deployment.

1.3 Report Structure

This report is systematically organized to build a comprehensive understanding of local AI energy footprinting.

- **Section 2** will dissect the key factors influencing energy consumption, detailing the roles of hardware, AI models, software, and utilization patterns.
- **Section 3** will introduce the proposed tiered methodologies for estimating energy footprints, balancing simplicity with accuracy, and discussing component versus system-level measurements.
- **Section 4** will present collated hardware and task-specific energy profiles, drawing from benchmarks and manufacturer data to form the basis for estimations.
- **Section 5** will offer practical guidance for users on selecting and using tools to measure their own local AI energy consumption.
- **Section 6** will explore considerations for developing a user-friendly footprint estimation calculator, including UI/UX recommendations and the optional incorporation of carbon intensity data.
- **Section 7** will conclude the report with a summary of critical factors, underlying assumptions, and key recommendations for users, developers, and future research.

Section 2: Key Factors Influencing Local AI Energy Consumption

The energy consumed during local AI execution is not a monolithic value but rather a complex interplay of multiple variables. A thorough understanding of these factors is essential for accurate estimation and effective optimization. This section details the primary drivers of power draw, categorized into hardware dependencies, AI model characteristics, software and framework choices, and workload utilization patterns.

2.1 Hardware Dependencies: The Engine of AI Power Draw

The specific hardware components within a system are the most direct determinants of its power consumption characteristics when running AI workloads. Different processors and supporting components exhibit vastly different power profiles and efficiencies.

2.1.1 Graphics Processing Units (GPUs): The Primary Consumer

GPUs are the workhorses for most local AI tasks due to their architecture, which is highly optimized for parallel computations inherent in deep learning models. This computational prowess, however, typically comes with significant energy demands. The power consumption of a GPU is influenced by its architecture, manufacturing process, clock speeds, and the number of active compute units.

- **NVIDIA GeForce RTX Series:**
    - **High-End Desktop (e.g., RTX 4090, RTX 4080/Super, RTX 5090 (Anticipated)):** The RTX 4090, with a Total Board Power (TBP) of 450W, is a prime example of high performance coupled with high power draw. NVIDIA recommends a minimum system Power Supply Unit (PSU) of 850W for systems equipped with an RTX 4090. In real-world scenarios, the RTX 4090's power consumption can range from 400W to 600W during intensive tasks. The anticipated RTX 5090 is expected to exhibit even higher power excursions, potentially spiking to over 900W for very short durations (<1ms), though its sustained power might be around 575-600W. The RTX 4080 and its Super variant typically have a TBP around 320W.
    - **Mid-Range Desktop (e.g., RTX 4070 Ti Super, RTX 4070/Super, RTX 4060 Ti):** These GPUs offer a more balanced performance-to-power ratio. The RTX 4070 Ti Super has a TBP of 285W. The RTX 4070 and RTX 4070 Super have TBPs of 200W and 220W, respectively. An RTX 4070 Super running Mistral 7B can consume approximately 150W while delivering 50-60 tokens/second. The RTX 4060 Ti has a TBP of 160W, but has been observed drawing up to 180W during LLM inference tasks like Mixtral 8x7b.
    - **Entry/Mainstream Desktop (e.g., RTX 4060):** The RTX 4060 has a TBP of 115W and can run 7B parameter LLMs efficiently.
    - **Older Generations (RTX 30-series, RTX 20-series):** These remain prevalent. The RTX 3090 has a 350W TBP, the RTX 3080 a 320W TBP, the RTX 3070 a 220W TBP, and the RTX 3060 12GB a 170W TBP. For the 20-series, the RTX 2080 Ti has a 250W TBP, and the RTX 2070 a 175W TBP, with the latter observed drawing around 100W for Mistral 7B inference.
    - **Mobile GPUs (Laptop):** These are power-constrained relative to their desktop counterparts. The RTX 4090 Mobile has a base TGP of 115W, configurable up to 175W with Dynamic Boost, though enabling this boost on Linux can be problematic. The RTX 4080 Mobile has a 110W TGP. An RTX 4080 Mobile at 150W was shown by ComputerBase to match the performance of an RTX 4090 Mobile at 125W in certain tests. The RTX 4070 Mobile can go up to 105W (e.g., in ASUS ROG Zephyrus G16), while the RTX 4060 Mobile in a Dell XPS 16 is rated at 50W. Older mobile GPUs like the RTX 3070 Mobile and RTX 3060 Mobile have TGPs of 115W and 80W, respectively.
- **AMD Radeon RX Series:**

    - **High-End Desktop (e.g., RX 7900 XTX, RX 7900 XT):** The RX 7900 XTX has a TBP of 355W. In non-gaming workloads like Blender, a Sapphire Nitro+ model was observed reaching 3.5 GHz GPU clock at 374W, though it hits a 500W power limit in games. The RX 7900 XT has a TBP in the 300-315W range.
    - **Mid-Range Desktop (e.g., RX 7800 XT, RX 7700 XT):** The RX 7800 XT has a TBP of 263W, with average gaming power draw around 230W, and is noted to handle Stable Diffusion efficiently. The RX 7700 XT has a TBP of 245W.
    - **Older Generations (RX 6000-series, RX 5000-series):** The RX 6800 XT has a 300W TBP. The RX 6700 XT has a 230W TBP and can achieve around 33 tokens/second with Mistral 7B using llama.cpp. The RX 5700 XT has a TBP of 225W.
- **Intel Arc GPUs (Desktop):**
    - The Arc A770 (16GB) has a TGP of 225W. It has been noted for relatively high idle power consumption (around 40W) and gaming power draw (around 230W), with overall efficiency generally lower than contemporary NVIDIA and AMD offerings.
    - The Arc A750 (8GB) also has a 225W TGP, exhibiting similar power characteristics to the A770.
- **Apple M-series Integrated GPUs (Laptops & Desktops):**
    - Apple's silicon is designed with a strong emphasis on power efficiency. The M3 Max GPU, during Cinebench tests, consumed around 33W in normal power mode and 15W in low power mode. For Stable Diffusion tasks, an M1 Max GPU was observed drawing approximately 28-30W in regular mode and 18W in low power mode. An M4 Pro 20-core GPU is estimated to consume around 32W.
- **Video RAM (VRAM):** The amount of VRAM (e.g., 8GB, 12GB, 16GB, 24GB) is a critical factor. It dictates the size of models that can be loaded directly into GPU memory, the batch sizes that can be processed, and whether parts of the model or intermediate data (like KV caches) need to be offloaded to slower system RAM or even disk. Insufficient VRAM can lead to significantly slower performance or an inability to run certain models/configurations, thereby indirectly affecting total energy consumption by prolonging task duration or forcing reliance on less power-efficient processing paths (e.g., CPU).

The TBP or TGP figures provided by manufacturers are often indicative of maximum gaming loads or design limits and may not directly reflect power consumption during specific AI inference or training tasks. AI workloads can stress different parts of the GPU (e.g., Tensor Cores, memory bandwidth) compared to gaming shaders. Therefore, while TBP/TGP serves as a useful baseline, measured power during actual AI tasks is more representative. For instance, an RTX 4060 Ti with a 160W TBP was measured drawing up to 180W during LLM inference, while an RTX 4090 system (450W TBP for GPU) can draw 400-600W at the wall during intensive AI or rendering tasks. This discrepancy highlights that the rest of the system components (CPU, RAM, motherboard, storage, cooling) and PSU inefficiency contribute significantly to the total wall power draw, often adding a substantial overhead beyond the GPU's rated power. NVIDIA's recommendation of an 850W PSU for a 450W TBP RTX 4090 implies an allowance of up to 400W for the rest of the system and efficiency losses.

2.1.2 Central Processing Units (CPUs): Supporting Role and CPU-Only Inference

While GPUs often take center stage for AI, CPUs play a vital supporting role and can also be the primary compute unit for AI inference, especially in systems without powerful discrete GPUs or for specific model types.

- **Desktop CPUs:**
    - Modern desktop CPUs like Intel Core i7/i9 (e.g., 13th/14th Gen) and AMD Ryzen 7/9 (e.g., 7000/9000 series) have varying TDPs and actual power consumption. For example, an Intel Core i7-13700K has a Processor Base Power (PBP) of 125W and a Maximum Turbo Power (MTP) of 253W. The AMD Ryzen 9 7900 has a 65W TDP but typically draws around 86W at stock settings and can go up to 200W with Precision Boost Overdrive (PBO) enabled.
    - For CPU-only LLM inference, performance is heavily dependent on factors like core count, clock speed, cache size, and critically, system memory bandwidth. An 8-billion parameter LLM on a CPU-only setup might achieve 10-20 tokens/second depending on the architecture. A Framework Laptop 16 with a Ryzen 7940HS (a mobile CPU but in a desktop-replacement context here) achieved 11.96 tokens/second for Llama 8B Q4_K_M generation and 1.36 tokens/second for a 70B model.
- **Mobile CPUs:**
    - Laptop CPUs, such as Intel Core Ultra series and AMD Ryzen Mobile series, are designed with a greater emphasis on power efficiency. Intel Core Ultra 7 155H has a base power of 28W and can turbo up to 115W, with system base power around 64W. AMD Ryzen 7 7840HS has a configurable TDP between 35W and 54W.

- **CPU Power in GPU-Accelerated Systems:** Even when a powerful GPU is present, the CPU consumes power for data loading, pre-processing, post-processing, managing the AI framework, and running the operating system and other background tasks. The interaction between CPU and GPU, especially data transfer over PCIe, also contributes to system power.

CPUs generally range from 4W to over 200W depending on the model and load. The efficiency of CPU-only inference is often much lower than GPU inference in terms of performance per Watt for models that can leverage GPU parallelism.

2.1.3 Neural Processing Units (NPUs): Emerging Efficiency

A newer class of processor, the NPU, is becoming increasingly common in System-on-Chips (SoCs), particularly in laptops and mobile devices. NPUs are designed specifically to accelerate AI and machine learning tasks with high energy efficiency, often outperforming GPUs or CPUs for certain AI operations on a power-per-inference basis.

- **Apple Neural Engine (ANE):** Apple's M-series chips integrate a powerful NPU. For a Llama-3.2 1B model, the ANE was reported to draw approximately 1.7W compared to 8W for the GPU on the same chip, albeit with a slight reduction in tokens/second (45 t/s on ANE vs. 56 t/s on GPU). An M1 Max ANE consumed ~1.8W (47 t/s) and an M4 Pro ANE ~2.8W (62 t/s) for Llama3.2-1B, while the respective GPUs drew ~20W for roughly twice the speed.
- **Intel AI Boost (Core Ultra NPUs):** Intel's Core Ultra processors feature an integrated NPU. The iGPU on an Intel Ultra 5 125H was found to be up to 30% faster and consume about three times less power than the CPU for inference. The Core Ultra 9 285H includes an NPU delivering up to 13 TOPS, with the overall SoC having a base power of 35-45W and a peak of 115W.
- **AMD Ryzen AI:** AMD's Ryzen AI platform also incorporates NPUs. The Ryzen AI 9 365 SoC showed an average power consumption of 18.8W (peak 33.9W) across various benchmarks. The more powerful Ryzen AI Max+ PRO 395 averaged 48W (peak 72W). AMD's architecture for some LLMs uses the NPU for accelerating Time-To-First-Token (TTFT) and the iGPU for subsequent token generation.

NPUs excel at specific types of AI computations, often involving lower precision and repetitive calculations, making them particularly well-suited for inference tasks on edge devices where power budgets are tight.

2.1.4 System Memory (RAM) and Storage

- **RAM:** The speed and capacity of system RAM can significantly influence performance, especially for CPU-only inference or when GPU VRAM is insufficient, forcing model layers or data to be swapped to system RAM. This swapping process is slow and energy-intensive. For CPU-bound AI tasks, memory bandwidth is a critical performance bottleneck. Larger models may require substantial system RAM (32GB, 64GB, or even more) if they cannot fit entirely in VRAM.
- **Storage:** The speed of the storage device (e.g., NVMe SSD vs. SATA SSD or HDD) primarily affects model loading times. While this might be a small fraction of the total energy for very long inference tasks, for frequent model switching or short tasks, faster loading can reduce overall "task energy" if the measurement window includes loading.

2.1.5 Motherboard and Power Supply Unit (PSU) Efficiency

- **Motherboard:** The motherboard itself consumes power to operate its chipsets, VRMs, and connected peripherals. While often a smaller portion of the total system draw under heavy AI load, it's a constant factor.
- **PSU Efficiency:** The PSU converts AC power from the wall outlet to DC power for the components. This conversion is not 100% efficient; some energy is lost as heat. PSU efficiency ratings (e.g., 80 PLUS Bronze, Gold, Platinum, Titanium) indicate how much power is delivered to components versus drawn from the wall. For example, an 80 PLUS Gold PSU is typically 87-90% efficient at 20-100% load. This means if components require 500W, a 90% efficient PSU will draw ~555W from the wall. This is critical when comparing software-reported component power to wall-measured system power.

The interplay between these hardware components defines the system's baseline and peak power consumption characteristics, directly impacting the energy footprint of any local AI task.

Table 1: Hardware Tier Definitions for Local AI (2024-2025)

To provide a structured approach for estimating energy consumption, it is useful to define broad hardware tiers relevant to local AI tasks in the 2024-2025 timeframe. These tiers are characterized by typical GPU and CPU combinations, VRAM, and system RAM.

Tier Name	Typical GPU Series/Models	Typical CPU Class	Typical VRAM (GB)	Typical System RAM (GB)	Key AI Use Cases
High-End GPU Desktop	NVIDIA RTX 4090, RTX 4080/Super, RTX 5090 (Anticipated); AMD RX 7900 XTX/XT	Intel Core i7/i9 (13th/14th/Ultra Gen), AMD Ryzen 7/9 (7000/9000 Series)	16-24+	32-64+	Large LLMs (30B+ Q, 70B+ heavy Q), high-res image gen, model fine-tuning, fast inference
Mid-Range GPU Desktop	NVIDIA RTX 4070/Ti/Super, RTX 4060 Ti (16GB); AMD RX 7800 XT, RX 7700 XT	Intel Core i5/i7 (13th/14th/Ultra Gen), AMD Ryzen 5/7 (7000/8000 Series)	12-16	16-32 (32+ recommended)	Medium LLMs (13B-30B Q), Mixtral (Q), SDXL, good inference speed
Entry GPU Desktop	NVIDIA RTX 4060, RTX 3060 (12GB), RTX 2070/Super; AMD RX 6700 XT, RX 6600 XT/6650 XT; Intel Arc A750/A770 (8GB/16GB)	Intel Core i3/i5 (12th+ Gen), AMD Ryzen 3/5 (5000+ Series)	8-12 (16GB for Arc A770)	16-32	Smaller LLMs (7B-13B Q), Stable Diffusion 1.5/2.1, experimentation
CPU-only Desktop	No discrete GPU, or iGPU only (e.g., Intel UHD/Iris Xe, AMD Radeon	Intel Core i5/i7/i9 (Recent Gens), AMD Ryzen 5/7/9 (Recent	N/A (uses Syste	32-64+ (bandwidth critical)	Smaller LLMs (e.g., 7B Q) with patience, tasks less

	Graphics on Ryzen non-APU/APU)	Gens, esp. with high core counts)	m RAM)		sensitive to latency
High-Performance AI Laptop	NVIDIA RTX 4070M/4080M/4090M; Apple M2/M3/M4 Max/Ultra	Intel Core Ultra 7/9 H/HX Series, AMD Ryzen 7/9 HS/HX Series; Apple M2/M3/M4 Max/Ultra	8-16 (GPU), Unified (Apple)	16-32+ (GPU), 36-128+ (Apple Unified)	Capable LLM inference (quantized), image gen on-the-go, NPU-accelerated tasks
Standard Laptop (iGPU/NPU)	Intel Arc iGPU (Core Ultra), AMD Radeon iGPU (Ryzen 7000/8000 APU), with NPU (Intel AI Boost, AMD Ryzen AI)	Intel Core Ultra 5/7 U/H Series, AMD Ryzen 5/7 U/HS Series (with NPU)	N/A (uses System RAM)	16-32	Light LLM inference (small models, heavily Q), NPU-optimized tasks, efficient AI
Apple M-Series Laptop (Base/Pro)	Apple M1/M2/M3/M4 (Base/Pro integrated GPU & Neural Engine)	Apple M1/M2/M3/M4 (Base/Pro)	Unified	16-36+ (Unified)	Efficient LLM inference (small-medium Q models), Stable Diffusion (CoreML opt.)

Sources for Table 1: Synthesized from numerous sources including S2, S3, S4, S6, S7, S8, S12, S18, S29, S43, S44, S45, S46, S97, S99, S101, S103, S105, S111, S112, S113, S119, S124, S125, S126, S127, S136, S137, S139, S140, S141, S142, S143, S160, S161, S163, S170, S171, S172, S173, S174, S175, S176, S188, S189, S190, S191, S196, S197, S200, S201, S202, S208, S209, S210, S211, S212, S213, S214, S215, S222, S224, S225, S226, S227, B4, B5, B6, B8, B11, B12, B15.

(Note: "Q" refers to quantized models. Performance and model capability within tiers can vary significantly.)

This table serves as a general guideline. Specific component choices within each tier, as well as the other factors discussed below, will further refine the energy footprint.

2.2 AI Model Characteristics: The Workload Definition

The nature of the AI model itself is a fundamental determinant of the computational workload and, consequently, its energy consumption.

2.2.1 Model Size and Parameters (e.g., 7B, 13B, 32B, 70B+)

The number of parameters in a model (often measured in billions, e.g., 7B for a 7-billion parameter model) is a primary indicator of its complexity and resource requirements. Larger models generally necessitate more computational operations (FLOPs) per inference step and demand more VRAM to store weights and activations.

For instance, an NVIDIA RTX 4090 with 24GB of VRAM can accommodate 70B+ parameter models with appropriate quantization, whereas a 12GB card like the RTX 4070 would struggle with a model like Mixtral 8x7B (a Mixture-of-Experts model with a large effective parameter count) unless it is heavily quantized and potentially offloaded to system RAM. The training of Llama 2 70B, for example, involved GPUs consuming 400W for approximately 1.7 million GPU-hours, illustrating the scale of energy involved with very large models, even if inference is a fraction of that. Higher parameter counts typically translate to longer processing times per token or image, and potentially higher sustained power draw if the hardware is fully utilized.

2.2.2 Model Architecture (e.g., Transformer, MoE)

Beyond raw parameter count, the specific architecture of an AI model influences its computational intensity and memory access patterns. Most modern LLMs are based on the Transformer architecture, but variations exist (e.g., Llama, Mistral, Qwen, Phi-3) which can have different efficiencies even at similar parameter scales.

Mixture-of-Experts (MoE) models, such as Mixtral 8x7B, have unique characteristics. While they might have a very large total parameter count, only a subset of "experts" (and thus parameters) are activated for any given input token. This can lead to faster inference than a dense model of equivalent total parameters, but VRAM requirements can still be substantial to hold all experts. The efficiency of these architectures can vary across different hardware platforms.

2.2.3 Quantization Techniques (FP16, INT8, GGUF Qx_K_M, AWQ, etc.)

Quantization is a critical technique for running large AI models on resource-constrained local hardware. It involves reducing the numerical precision of the model's weights and, in some cases, activations (e.g., from 32-bit floating point (FP32) or 16-bit floating point (FP16) to 8-bit integer (INT8) or even 4-bit/3-bit representations).

- **Impact on VRAM and Size:** The most immediate benefit of quantization is a reduction in model size and VRAM footprint, often by a factor of 2x to 8x or more. This allows larger models to fit on GPUs with limited VRAM or reduces the need for offloading to system RAM.
- **Impact on Speed:** Quantization can lead to faster inference, especially if VRAM bandwidth was a bottleneck or if the hardware has specialized units for lower-precision arithmetic. For example, NVIDIA GPUs support lower precision computations like INT8 and FP8, allowing more multiplication units in a given chip area.
- **Impact on Quality:** There is typically a trade-off between the degree of quantization and model accuracy (quality of output). Highly aggressive quantization (e.g., 2-bit or 3-bit) may lead to noticeable degradation, while moderate levels (e.g., 4-bit, 5-bit, or 8-bit) often maintain good performance with minimal quality loss for many models. Formats like GGUF offer various quantization levels (e.g., Q4_K_M, Q5_K_M, Q8_0), allowing users to choose a balance.
- **Impact on Power and Energy:** The effect of quantization on power consumption is nuanced.
    1. Reduced data movement due to smaller model sizes can decrease energy related to memory access.
    2. Simpler arithmetic operations (e.g., integer vs. floating-point) could theoretically require less energy per operation on the processing units themselves. The BitNet b1.58 paper, for example, highlights massive theoretical energy savings by primarily using integer additions instead of floating-point multiplications.
    3. If quantization leads to significantly faster inference (more tokens/images per second) without a proportional increase in instantaneous power draw (Watts), then the total energy per task (Power × Time) will decrease. For example, an RTX 4070 Super running Mistral 7B at ~150W achieved 50-60 tokens/second, while an older GTX 1070 at ~100W achieved 25-30 tokens/second; the newer card is more power-hungry but significantly more energy-efficient per token.
    4. However, if the GPU still operates near its power limit to process these "cheaper" operations more quickly, the instantaneous power draw might not decrease substantially. The primary energy saving then comes from the reduction in task duration.
    5. Specific quantization schemes also have different computational overheads. For instance, S118 suggests that for VRAM-constrained scenarios (offloading), GGUF quantizations like NF4 and Q4_0 can be faster than Q8_0 or even FP16 (which wouldn't fit), indicating that the

benefits of fitting more in VRAM outweigh the specific computational costs of those quantization types.

The complex interplay between quantization level, available VRAM, hardware architecture, and the resulting inference speed means that simple assumptions (e.g., "half the bits equals half the power") are unlikely to hold true. Empirical measurements of both performance and power are necessary to determine the actual energy efficiency of different quantization strategies on specific hardware.

2.3 Software and Framework Choices: The Execution Environment

The software stack, from low-level drivers to high-level user interfaces, plays a crucial role in how efficiently AI models are executed and can significantly impact both performance and power consumption.

2.3.1 Inference Engines (e.g., llama.cpp, Ollama, TensorRT-LLM, PyTorch)

The choice of inference engine is critical, as different engines are optimized to varying degrees for specific hardware architectures and model types.

- **TensorRT-LLM:** NVIDIA's TensorRT-LLM is highly optimized for NVIDIA GPUs, leveraging Tensor Cores and specific architectural features. Benchmarks indicate it can be 30-70% faster than llama.cpp on the same NVIDIA hardware for models like Mistral 7B. It tends to utilize GPU VRAM more fully while reducing CPU and system RAM usage. For example, on an RTX 4090, TensorRT-LLM achieved 170.63 tokens/second for Mistral 7B Q4_K_M, compared to 100.43 tokens/second for llama.cpp with GPU offload. Such a speedup, if not accompanied by a proportional increase in instantaneous power, would lead to lower energy consumption per task.
- **llama.cpp:** A widely used C++ library for LLM inference, popular for its cross-platform compatibility and CPU inference capabilities. It also supports GPU offloading via CUDA (NVIDIA), Metal (Apple), and Vulkan. Its performance can be highly dependent on compilation flags and quantization formats (like GGUF).
- **Ollama:** A popular tool that simplifies downloading and running LLMs. It often uses llama.cpp as its backend inference engine. Therefore, its performance and power characteristics can be similar to a direct llama.cpp setup if the underlying parameters (model, quantization, context size, GPU layers) are identical. However, default configurations or Ollama's management layer might introduce differences. A notable issue reported with Ollama is increased idle system power consumption (around 8W extra) when CUDA is enabled, due to the PCIe bus being kept active even when no inference is running.

- **PyTorch:** A general-purpose deep learning framework. While versatile, inference directly in PyTorch without specialized optimization libraries might not be as performant or power-efficient as dedicated inference engines for deployment.

The choice of inference engine can thus lead to substantial differences in energy efficiency, not just raw speed. An engine that achieves higher throughput without a commensurate increase in power draw will generally be more energy-efficient per unit of AI work.

2.3.2 User Interface Frontends (e.g., ComfyUI, Automatic1111, LM Studio, Jan)

For tasks like image generation with Stable Diffusion or interacting with LLMs, users often employ graphical frontends. Examples include ComfyUI and Automatic1111 for Stable Diffusion, and LM Studio or Jan for LLMs. These frontends typically utilize one of the aforementioned inference engines in the backend. While the frontend itself adds some CPU and RAM overhead, the primary power consumption during active AI processing will still be dominated by the backend engine's utilization of the GPU or CPU. However, some frontends might be more lightweight or better optimized than others. For instance, ComfyUI has been reported to be faster and use resources more efficiently than Automatic1111, especially on older GPUs with limited VRAM.

2.3.3 Driver Versions and System Libraries (CUDA, ROCm)

The performance and efficiency of AI workloads can also be affected by the versions of GPU drivers (e.g., NVIDIA Game Ready/Studio drivers, AMD Adrenalin Edition) and foundational system libraries like CUDA for NVIDIA GPUs or ROCm for AMD GPUs. Updated drivers and libraries often include optimizations for newer hardware, specific model architectures, or improved implementations of AI operations, which can lead to better performance and potentially lower power consumption for the same task.

The software environment, therefore, is not a passive component but an active modulator of the hardware's energy footprint during AI tasks.

2.4 Workload and Utilization Patterns: How AI is Used

The way AI models are employed—the type of task, its duration and intensity, and system activity during idle periods—significantly influences overall energy consumption.

2.4.1 Task Type (LLM Inference vs. Image Generation)

Different AI tasks impose different computational demands.

- **LLM Inference:** This typically involves two main stages: a "prefill" stage where the input prompt is processed in parallel, and a "decode" (or generation) stage where tokens are generated auto-regressively one by one. For typical interactive

chat applications with moderate input prompt lengths and longer generated outputs, the decode stage often dominates the overall processing time and thus energy consumption. The computational patterns involve large matrix multiplications and significant memory bandwidth usage for accessing weights and the KV cache.
- **Image Generation (e.g., Stable Diffusion):** These models usually involve an iterative denoising process over a fixed number of steps (e.g., 20-50 steps). Each step applies a neural network (often a U-Net) to refine a latent representation of the image. The workload characteristics differ from token-by-token LLM generation.

2.4.2 Task Duration and Intensity

- **Duration:** The longer an AI task runs, the more total energy it will consume, assuming a relatively constant power draw. This is straightforward: Energy = Power × Time.
- **Intensity:** Task intensity can be influenced by several factors:
    - For LLMs: Length of the input prompt, number of tokens to generate, batch size (if processing multiple requests simultaneously), and context window size (longer contexts require more VRAM for the KV cache and can slow down attention mechanisms).
    - For Image Generation: Image resolution (e.g., 512x512 vs. 1024x1024), number of inference steps, use of high-resolution fix, batch size, and complexity of ControlNets or LoRAs used.
    - Higher intensity generally leads to more computation per unit of output, potentially increasing instantaneous power draw or, more commonly, extending task duration. Performance metrics like tokens/second for LLMs and seconds/image or images/minute for image generation are crucial for normalizing energy consumption across different settings and hardware.

2.4.3 Idle Power Consumption

Systems consume power even when not actively performing AI tasks. This idle power can be a significant contributor to the overall energy footprint, especially if systems are left powered on for extended periods or if AI frameworks keep hardware components in a high-power state unnecessarily.

- The issue with Ollama and CUDA keeping the PCIe bus active, leading to an ~8W increase in idle system power, is a case in point.
- Intel Arc GPUs have also been noted for higher idle power consumption (around 40W for the GPU alone) compared to competitors.

- Even efficient modern GPUs like the RTX 4070 draw around 10W at idle, with the entire system consuming more. Understanding and minimizing idle power is important for a complete energy footprint assessment.

2.4.4 Background Processes and System Load

Other applications and background processes running on the system consume CPU, RAM, and potentially GPU resources, adding to the total power draw. When measuring the energy footprint of a specific AI task, it is crucial to minimize or account for this background load to ensure the measurements accurately reflect the AI workload's impact.

The combination of these factors—hardware, model, software, and utilization—creates a complex, high-dimensional space for energy consumption. Effective estimation requires considering these interdependencies. The observation that quantization's impact on power is non-linear, or that software framework overhead can be a hidden variable, underscores this complexity. For instance, while quantization reduces VRAM load and can increase speed, especially when VRAM-bound, its direct effect on GPU power per computation isn't straightforward. A GPU might use its full power budget to process these "cheaper" operations faster, leading to a net energy saving primarily through reduced time. Similarly, optimized inference engines like TensorRT-LLM can significantly improve throughput on NVIDIA GPUs compared to more general engines like llama.cpp. If this speed gain isn't offset by a proportionally larger power increase, the energy per processed token will be lower. These interactions necessitate careful benchmarking and a nuanced approach to estimation, rather than relying on simplistic assumptions.

Section 3: Methodologies for Estimating Energy Footprint

Estimating the energy footprint of locally run AI models requires a structured approach that can accommodate varying levels of available information and user technical expertise. This section proposes a tiered estimation framework, discusses the nuances of component-level versus system-level power measurement, and emphasizes the importance of accounting for task duration, performance, and idle states.

3.1 Tiered Estimation Framework: Balancing Simplicity and Accuracy

A tiered framework allows users to obtain an estimate with minimal input, while also providing paths for more accurate assessments with more detailed information. This addresses the inherent trade-off: simpler methods are easier to use but generally less precise, whereas more accurate methods demand greater user effort, such as performing direct measurements.

- **3.1.1 Level 1: Basic Estimation (Hardware Category & Task Type)**
    - *Concept:* This level aims to provide a coarse, order-of-magnitude estimate with minimal user input. Users select their general hardware category (as defined in Table 1, e.g., "Mid-Range GPU Desktop," "High-Performance AI Laptop") and a broad AI task type (e.g., "LLM Inference - 7B Model," "Image Generation - 1024x1024").
    - *Inputs:* User selects from predefined hardware categories and task descriptions.
    - *Output:* An estimated typical power draw range (Watts) for the system during that task, and potentially a typical task duration or performance metric (e.g., tokens/second range). This allows for a basic energy estimate (Watt-hours).
    - *Data Sources:* This level would rely on synthesized averages and typical ranges derived from the benchmark data collated in Section 4. For example, a "Mid-Range GPU Desktop" running a "LLM Inference (7B model)" might be associated with a default system power range of 200-350W based on data for RTX 4060/4070 class systems.
    - *Utility:* Provides a quick, accessible estimate for users without detailed knowledge of their hardware specifics or AI model parameters. It serves as an initial awareness tool.
- **3.1.2 Level 2: Intermediate Estimation (Specific Hardware, Model, Quantization)**
    - *Concept:* This level offers a more refined estimate by allowing users to input specific details about their hardware and the AI model they are running.
    - *Inputs:*
        - Specific GPU model (e.g., NVIDIA GeForce RTX 4070, AMD Radeon RX 7800 XT).
        - Specific CPU model (e.g., Intel Core i7-13700K, AMD Ryzen 7 7800X3D).
        - AI Model name and size (e.g., Llama 2 7B, Mistral 7B Instruct, Stable Diffusion XL).
        - Quantization level (e.g., FP16, GGUF Q4_K_M, GGUF Q8_0, AWQ).
        - Optionally, the software framework being used (e.g., Ollama, llama.cpp).

    - *Output:* A more tailored power draw estimate (Watts) based on specific hardware profiles (from Section 4, Tables 2-5), typical performance figures (tokens/second or seconds/image), and VRAM/RAM usage estimates.
    - *Data Sources:* Utilizes the detailed hardware/task profiles gathered in Section 4. For example, if a user specifies an RTX 4090 and Llama 2 13B (Q4_K_M) with Ollama, the system could reference data similar to that in S35 (70.9 tok/s, 92% GPU utilization) to inform the power estimate.
    - *Utility:* Caters to users who know their system components and the model they are running, providing a more accurate estimate than Level 1.
- **3.1.3 Level 3: Advanced Estimation (User-Provided Measurements & Detailed Inputs)**

    - *Concept:* This is the most accurate tier, relying on the user to perform direct measurements of their system's power consumption for the specific AI task.
    - *Inputs:*
        - Measured average system idle power (Watts) with AI framework loaded.
        - Measured average system load power (Watts) during the AI task (preferably wall power).
        - Actual task duration (seconds or minutes).
        - Number of tokens generated or images produced during the measured duration.
        - Optionally, specific component power readings if available and relevant for deeper analysis.
    - *Output:* A calculated energy consumption figure (Watt-hours or kWh) specific to the user's system and exact workload. Performance metrics like energy per token or energy per image can also be derived.
    - *Data Sources:* User's own measurements using tools and techniques described in Section 5.
    - *Utility:* Provides the most personalized and accurate footprint assessment. This level is crucial given the general scarcity of comprehensive, publicly available wall power benchmarks for diverse AI workloads (a point underscoring the "Proxy Problem" discussed later).

The successful implementation of this tiered approach in any estimation tool hinges on clearly communicating the assumptions and expected accuracy of each level.

3.2 Component-Level vs. System-Level (Wall Power) Measurement

A critical distinction in power measurement is between component-level data and total system-level power draw, especially power measured at the wall outlet.

- **Component-Level Power:** This refers to the power consumed by individual hardware components, such as the GPU or CPU.
    - GPU power is often reported as Total Graphics Power (TGP) or Total Board Power (TBP) in manufacturer specifications. Software tools like HWiNFO or GPU-Z can also report GPU power draw, often labeled as "GPU Power" or "GPU Chip Power Draw."
    - CPU power is often cited by its Thermal Design Power (TDP) or specific power states like PL1/PL2 for Intel, or package power reported by software.
    - While useful for understanding the load on individual components, summing these software-reported values or relying solely on TDP/TGP provides an *underestimate* of the total system power drawn from the wall. This is because it omits the power consumed by other components (motherboard, RAM, storage drives, fans, peripherals) and, crucially, the energy losses incurred during AC-to-DC conversion by the PSU.
- **System-Level (Wall Power) Measurement:** This is the total electrical power consumed by the entire computer system, measured at the AC wall outlet using a device like a Kill A Watt meter or a professional power analyzer.
    - Wall power is the most holistic and relevant metric for determining the true energy footprint and operational cost, as it accounts for all components and PSU inefficiencies.
    - The challenge is that users require external hardware (a power meter) to perform these measurements, which may not always be available. HWiNFO's "System Power" metric, for example, is an *estimation* derived from summing available internal component sensor readings and is not a direct measurement of wall power. User reports suggest HWiNFO's component sum can be 25-35% lower than actual wall power.

The estimation methodology should prioritize wall power data when available. When relying on component power data (e.g., from software sensors or TGP/TDP specs), it's crucial to acknowledge that this represents a fraction of the total system draw and to consider applying a PSU efficiency factor (e.g., assuming 85-90% efficiency for a Gold/Platinum PSU, meaning wall draw is 10-15% higher than component sum) to approximate wall power, albeit with caveats about accuracy. GPUs are noted for their

high energy consumption during parallel processing tasks, and NPUs are emerging as potentially more efficient alternatives for specific AI workloads.

3.3 Accounting for Task Duration and Performance

The total energy consumed by an AI task is a product of the average power draw during the task and the duration of the task:

$$\text{Energy(Wh)} = \text{AveragePower(W)} \times \text{Duration(h)}$$

It is insufficient to consider only instantaneous power (Watts). A less powerful but slower system might consume more total energy to complete a task than a more powerful but faster system, even if the latter has a higher instantaneous power draw.

- **Performance Metrics for Normalization:** To make meaningful comparisons across different hardware, software, and model configurations, energy consumption should be normalized by the amount of work done.
    - For LLMs: Energy per token, or more practically, energy per 1000 tokens generated.
    - For Image Generation: Energy per image.
    - These normalized metrics (e.g., tokens/Wh, images/Wh, or Joules/token) provide a measure of energy efficiency.
- **LLM Inference Stages:** LLM inference consists of a prefill stage (processing the input prompt) and a decode stage (generating output tokens). The decode stage often dominates overall inference time in typical chat applications. These stages can have different computational intensities and thus different power draw profiles. While detailed modeling of each stage's power might be overly complex for a general estimator, acknowledging their existence is important if very granular analysis is attempted. For most practical purposes, an average power draw over the entire generation process (prefill + decode) is used.

3.4 Incorporating Idle and Background Power

Systems consume power even when not actively engaged in AI computation. This idle power can contribute significantly to the overall energy footprint, especially for systems that are frequently on or for AI frameworks that maintain high idle power states.

- **Measuring Idle Power:** It is advisable to measure the system's idle power consumption with the AI software/framework loaded but not actively processing any requests. This provides a baseline.
- **Task-Specific Energy vs. Total Operational Energy:**
    - For estimating the energy of a *specific, discrete AI task*, one approach is to subtract the baseline idle power from the load power and multiply this

delta by the task duration: $Energy_{task} = (Power_{load} - Power_{idle}) \times Duration_{task}$.
    - However, for the *total operational energy* associated with using local AI, the cumulative idle power over the period the system is powered on for AI use should also be considered. For example, the Ollama CUDA issue reportedly adds ~8W to idle system power continuously while the service is running. Intel Arc GPUs have also been noted for high idle power consumption (around 40W for the GPU alone). Even an efficient GPU like the RTX 4070 draws around 10W at idle, contributing to the system's overall idle draw.
- **Minimizing Confounding Factors:** Users should be advised to close unnecessary background applications when measuring AI task power to ensure the readings are as attributable as possible to the AI workload itself.

The prevalence of gaming-focused power benchmarks presents a "Proxy Problem." While these benchmarks indicate system power under high GPU load, AI workloads can differ in their utilization patterns of GPU sub-components (e.g., sustained Tensor Core usage for LLMs versus fluctuating shader loads in games). Gaming power draw can serve as a rough upper-bound proxy if AI-specific data is unavailable, but this limitation must be clearly communicated. This underscores the need for more AI-specific power benchmarking.

Section 4: Hardware and Task-Specific Energy Profiles

This section synthesizes available data on power consumption and performance for various hardware components and AI tasks. This information forms the empirical basis for the Level 1 and Level 2 estimation methodologies outlined in Section 3. Data is drawn from manufacturer specifications, technical reviews, and user benchmarks. It is important to note that "Typical Board Power" (TBP) or "Total Graphics Power" (TGP) for GPUs, and "Thermal Design Power" (TDP) for CPUs, are often indicative of maximums or typical gaming loads and may not precisely reflect power draw during specific AI inference tasks. Measured system wall power is the most comprehensive metric but is less commonly available.

4.1 Desktop GPU Power Profiles (NVIDIA)

- **RTX 40 Series:**
    - **RTX 4090:** Features a 450W TBP. NVIDIA recommends a minimum system PSU of 850W. For LLM inference using Ollama, Llama 2 13B (quantized) achieved 70.9 tok/s with 92% GPU utilization, and Llama 3.1 8B (quantized) achieved 95.51 tok/s with 94% GPU utilization; system

power was not specified for these tests, but GPU utilization was high. For Stable Diffusion XL (1024x1024, 20+5 steps), generation times range from 6.2 to 9.6 seconds per image. Real-world total system power draw during intensive tasks like AI or rendering can range from 400W to over 600W.
    - **RTX 4080 / Super:** The RTX 4080 has a TBP of 320W. For Stable Diffusion XL, the RTX 4080 achieved 7.2-14.2 seconds per image. Llama.cpp benchmarks with Llama 3.1 8B showed its performance-per-Watt to be comparable to the RTX 4090 for text generation, though slightly lower for prompt processing. A system with an RTX 4080 Super and a 7800X3D CPU can draw 250-320W (GPU) plus 50-70W (CPU) during gaming, suggesting system totals in the 300-400W+ range, excluding other components and PSU inefficiency.
    - **RTX 4070 Ti Super:** This GPU has a TBP of 285W.
    - **RTX 4070 Ti:** Also features a 285W TBP.
    - **RTX 4070 / Super:** The RTX 4070 has a 200W TBP, while the RTX 4070 Super is rated at 220W TBP. An RTX 4070 Super running Mistral 7B via llama.cpp was reported to consume around 150W for the GPU, achieving 50-60 tokens/second. An RTX 4070-based system showed GPU idle power at 10W and average gaming power at 186W (for the GPU). An undervolted RTX 4070 Super can draw around 165W for the GPU under load.
    - **RTX 4060 Ti (8GB/16GB):** The 8GB version has a 160W TBP. An RTX 4060 Ti (presumed 16GB due to model size) running Mixtral 8x7b (4-bit quantized) showed a maximum GPU load of 180W, delivering around 17 tokens/second. With Ollama, 7B parameter models on an RTX 4060 8GB (a distinct, lower-power card) achieved over 40 tokens/second with GPU utilization between 70-90%.
    - **RTX 4060:** This GPU has a 115W TBP. Running 7B models with Ollama, it can achieve over 40 tokens/second with GPU utilization in the 70-90% range.
- **RTX 30 Series:**

    - **RTX 3090:** Features a 350W TBP. For Stable Diffusion XL, it achieved 10.56-17.3 seconds per image. Its training throughput per Watt was found to be comparable to the RTX 4090 in some deep learning tasks, despite the 4090's higher raw TBP.
    - **RTX 3080:** Has a 320W TBP.
    - **RTX 3070:** Rated at 220W TBP.
    - **RTX 3060 12GB:** Has a 170W TBP.

- **RTX 20 Series:**
    - **RTX 2080 Ti:** Features a 250W TBP.
    - **RTX 2070:** Has a 175W TBP. It was observed consuming around 100W for the GPU when running Mistral 7B at 25-30 tokens/second.

4.2 Desktop GPU Power Profiles (AMD)

- **RX 7000 Series:**
    - **RX 7900 XTX:** Specified with a 355W TBP. In gaming, total system power can be substantial. Some specific machine learning training scenarios have shown it to be up to 30% faster than an RTX 4090, though this is highly workload-dependent. Its typical board power is 355W, compared to the RTX 4090's 450W and RTX 4080's 320W. A custom Sapphire Nitro+ model was observed drawing 374W (GPU only) in Blender, reaching 3.5 GHz, but is limited to 500W in games.
    - **RX 7900 XT:** Has a TBP in the range of 300-315W.
    - **RX 7800 XT:** Features a 263W TBP. Average gaming power draw for the GPU is around 230W, and it is reported to handle Stable Diffusion models efficiently.
    - **RX 7700 XT:** Has a 245W TBP.
- **RX 6000 Series:**
    - **RX 6800 XT:** Rated at 300W TBP. Stock GPU power draw is around 255-300W.
    - **RX 6700 XT:** Has a 230W TBP. When running Mistral 7B with llama.cpp (GPU offload), performance is around 33 tokens/second (similar to RX 6600).
    - **RX 6600:** Achieved 33 tokens/second with Mistral 7B (GPU offload) using llama.cpp.
- **RX 5000 Series:**
    - **RX 5700 XT:** Features a 225W TBP.

4.3 Desktop GPU Power Profiles (Intel Arc)

- **Arc A770 (16GB):** Has a TGP of 225W. Reviews note a relatively high GPU idle power of around 40W and gaming power draw around 230W for the card itself. Its performance is generally comparable to an NVIDIA RTX 3060.

- **Arc A750 (8GB):** Also has a 225W TGP. It exhibits similar power characteristics, with gaming power consumption for the card around 250W and idle near 40W.

4.4 Laptop GPU and Integrated Graphics/NPU Power Profiles

Laptop components operate under stricter thermal and power envelopes. NPUs are emerging as specialized low-power AI accelerators.

- **NVIDIA Mobile GPUs:**
    - **RTX 4090 Mobile:** Base TGP of 115W, configurable up to 175W with Dynamic Boost. However, enabling Dynamic Boost effectively on Linux systems has been reported as problematic.
    - **RTX 4080 Mobile:** Features a 110W TGP. Testing by ComputerBase indicated that an RTX 4080 Mobile might need about 20W more power to match the performance of an RTX 4090 Mobile at certain performance levels, suggesting the 4090M can be more efficient at equivalent performance points.
    - **RTX 4070 Mobile:** TGP varies by laptop design; for example, in a Razer Blade 14 or an ASUS ROG Zephyrus G16 where it can go up to 105W.
    - **RTX 4060 Mobile:** Configured at 50W in the Dell XPS 16.
    - **RTX 3070 Mobile:** Has a 115W TGP.
    - **RTX 3060 Mobile:** Features an 80W TGP.
- **Apple M-Series (GPU & NPU):**
    - Apple's M-series chips are known for their power efficiency due to the unified memory architecture and tight integration.
    - **M3 Max:** In Cinebench, the GPU consumed 33W (normal mode) and 15W (low power mode); the total SoC power was around 47W under such loads. For Stable Diffusion on an M1 Max (comparable high-end tier), the GPU drew 18W in Low Power Mode and around 28-30W in regular mode. An SDXL task (1024x1024, 30 steps) took about 70 seconds on an M3 Max.
    - **M2 Max:** While specific power for Mistral 7B (103.8 tok/s, 0.27s TTFT) wasn't stated, M2 Max is a common high-end Apple chip for such tasks. A baseline M2 MacBook Air running Stable Diffusion (CoreML, 50 steps) generated images in under 18 seconds.
    - **ANE (Apple Neural Engine) for LLMs (Llama3.2-1B):** M1 Max ANE: ~1.8W power draw, delivering 47 tok/s. M4 Pro ANE: ~2.8W power draw, delivering 62 tok/s. In comparison, the GPUs on these chips delivered roughly twice the speed but consumed around 20W. This highlights the NPU's efficiency for specific AI workloads.

    - Apple's official power consumption figures for Mac Studio models (which use M-series Ultra/Max chips) list idle power from 6-13W and maximum system power from 115W (M1 Max) to 295W (M2 Ultra) measured from the wall, covering all system losses. These are for general compute-intensive tasks, not necessarily AI-specific.
- **Intel Core Ultra with Arc iGPU & AI Boost NPU:**
    - **Core Ultra 7 155H:** CPU base power is 28W, with turbo up to 115W; typical system base power consumption is around 64W. System idle power for a laptop with 155H can be as low as ~4W. In UL Procyon AI Inference benchmarks, a system with Core Ultra 7 155H showed average total system power consumption of 23.9W (max 48.1W) using the GPU via Windows ML, and 25.8W (max 57W) using the GPU via OpenVINO.
    - **Core Ultra 5 125H:** The iGPU on this chip was found to consume about three times less power than the CPU for LLM inference, while offering up to 30% faster performance.
    - **Core Ultra 9 285H:** This CPU has an NPU rated for 13 TOPS, with the system having a base power of 35-45W and a peak of 115W.
- **AMD Ryzen APUs with Radeon iGPU & Ryzen AI NPU:**
    - **Ryzen 7 7840HS:** Has a configurable TDP between 35W and 54W.
    - **Ryzen AI 9 HX 370:** Featured in laptops like the Framework Laptop 13 and ASUS ROG Zephyrus G16.
    - **Ryzen AI Max+ PRO 395:** This high-end mobile SoC averaged 48W power consumption with a peak of 72W across nearly 200 benchmarks on an HP ZBook Ultra G1a. It demonstrated significantly faster LLM inference (up to 12.2x TTFT improvement) compared to Intel's Lunar Lake in specific scenarios.
    - AMD's Ryzen AI platform often utilizes the NPU for optimizing Time-To-First-Token (TTFT) in LLMs, while the integrated Radeon graphics (iGPU) handles the bulk of subsequent token generation.
- **Other Integrated Graphics:**
    - Intel Iris Xe MAX (discrete mobile, older gen): 25W TDP.
    - AMD Radeon 680M (iGPU, older gen): 50W TDP.

4.5 CPU-Only Inference Profiles

When no capable discrete GPU or NPU is available or utilized, AI inference falls back to the CPU.

- **Framework Laptop 16 (Ryzen 7940HS):** For Llama 8B Q4_K_M, prompt processing (pp512) was 87.73 tok/s, and token generation (tg16) was 11.96 tok/s. For Llama 70B Q4_K_M, pp512 was 9.16 tok/s, and tg16 was 1.36 tok/s. Specific power consumption for these CPU-only tasks was not provided.
- **AMD Ryzen 9 Desktop CPUs:**
    - **Ryzen 9 9950X:** General estimates for an 8B model on CPU-only are around 10-20 tok/s. This CPU averaged 148W with a peak of 210W across a wide range of (mostly non-AI) benchmarks.
    - **Ryzen 9 7900/7900X:** The 7900 (65W TDP) consumes ~86W stock, up to 200W with PBO. In Cinebench, the 7900 drew 90W stock and 185W when overclocked. The 7900X (170W TDP) drew 108W in gaming and 185W in multi-threaded workloads (CPU-only power).
- **Intel Core Desktop CPUs:**
    - **Core i7-13700K:** Has a Processor Base Power of 125W and Maximum Turbo Power of 253W.
    - **Core i7-13700 (non-K):** TDP of 65W, PL2 (Max Turbo Power) of 219W.
- **General CPU Power:** CPU power consumption can range broadly from 4W to over 200W depending on the specific chip and workload intensity. For CPU-only inference, system memory bandwidth is a critical performance determinant.

4.6 Impact of Model Quantization on Power and Performance

Quantization significantly impacts VRAM usage and performance, with nuanced effects on power.

- **Llama 2 7B Q4_K_M vs. Q8_0 on RTX 4070 (or similar class like 4070 Super):**
    - An RTX 4070 Super (220W TBP) running Mistral 7B (a model of similar size to Llama 2 7B) at a Q4-equivalent quantization consumed ~150W (GPU power) for 50-60 tok/s.
    - GGUF Q8_0 models are generally more precise but slower and larger than Q4_K_M or Q5_K_M variants. Q6_K is often suggested as a good balance, being similar in quality to Q8_0 but faster and smaller.
    - For a 12GB card like the RTX 4070, Llama 3 8B (FP16, ~16GB) won't fit entirely, but high-quality quantizations (Q5_K_M, Q8_0) fit easily with good speed.
    - The primary benefit of 4-bit quantization on VRAM-constrained hardware is fitting larger models or achieving higher token generation rates by keeping more layers on the GPU, thus avoiding slower CPU offloading. The direct impact on instantaneous GPU power (Watts) is complex; the GPU might run faster at a similar wattage (improving energy/token) or at a slightly lower wattage.

- **BitNet b1.58 (Extreme Quantization Example):**
    - This 1.58-bit model shows massive theoretical energy savings (e.g., a 70B BitNet b1.58 is claimed to be 41.2 times more energy-efficient end-to-end than a 70B FP16 LLaMA model). This efficiency stems primarily from replacing most FP16 multiply-accumulate operations with simpler, lower-energy integer additions. While not directly comparable to consumer GGUF quantizations, it illustrates the potential energy benefits of aggressive, hardware-aware quantization.
- **General Trends:** Lower bit-depth quantizations (e.g., 4-bit) usually lead to higher tokens/second on VRAM-limited GPUs because more layers can reside on the faster GPU memory, reducing or eliminating slow offloading to CPU/system RAM. The impact on instantaneous GPU power draw (Watts) is less straightforward: the GPU might operate at a similar power level but process tokens faster (leading to better energy per token), or it might operate at a slightly lower power level. The total energy per task (Power × Time) is the crucial metric. S118 provides a qualitative speed ranking for GGUF quants under VRAM-constrained (offload) conditions: NF4 > Q4_0 > Q4_1 ≈ FP8 > Q4K_S > Q8_0.

4.7 Impact of Software Frameworks on Power and Performance

The choice of software framework can significantly alter performance and, by extension, energy efficiency.

- **TensorRT-LLM vs. Llama.cpp (NVIDIA GPUs):**
    - TensorRT-LLM is specifically optimized for NVIDIA GPUs and can be 30-70% faster than llama.cpp for models like Mistral 7B. It tends to achieve this by more fully utilizing VRAM and Tensor Cores, while potentially using less system RAM and CPU.
    - Example: For Mistral 7B (Q4_K_M) on an RTX 4090, llama.cpp (GPU offload) yielded 100.43 tok/s, while TensorRT-LLM achieved 170.63 tok/s. Max GPU utilization was slightly higher for TensorRT-LLM (88.5% vs 83.5%), and VRAM use was also higher (72.1% vs 64%), but average system RAM used was lower (4.98GB vs 7.105GB). This higher throughput, if power draw doesn't scale proportionally, suggests lower energy per token for TensorRT-LLM.
- **Ollama vs. Llama.cpp:**
    - Ollama frequently utilizes llama.cpp as its backend. Performance and power characteristics can be very similar if Ollama is configured to use the same llama.cpp parameters (e.g., context size, GPU layers, flash attention) as a direct llama.cpp invocation.
    - However, differences in default settings or any overhead from Ollama's management layer could lead to variations. A notable issue is the reported

higher idle power consumption (~8W system increase) when Ollama service is running with CUDA enabled, due to the PCIe bus being kept active.

- **ComfyUI vs. Automatic1111 (Stable Diffusion):**
    - User reports suggest ComfyUI may be faster and more resource-efficient, particularly on GPUs with limited VRAM, compared to Automatic1111. This could translate to lower energy per image generated.

The performance per Watt is a crucial metric for efficiency, but comprehensive, directly comparable data across a wide array of hardware, models, quantizations, and frameworks for local AI *inference* is still relatively sparse in publicly available benchmarks. Many power benchmarks focus on gaming, which, while indicative of high load, may not perfectly represent AI workload characteristics. This highlights a significant area for future standardized benchmarking.

Table 2: GPU Power Consumption (TGP/TBP or Measured) and Performance for LLM Inference

This table centralizes LLM-specific performance and power data for various GPUs. "GPU Power" refers to TGP/TBP from specs or measured GPU-only power. "System Power" is wall power if available. Performance is in tokens/second (t/s).

GPU Model	Model + Quantization	Framework	GPU Power (W)	System Power (W)	Perf. (t/s gen)	VRAM (GB)	Source(s)
NVIDIA RTX 4090	Llama 2 13B (Q unspecified)	Ollama	450 (TBP) / ~414 (Used)	N/A	70.9	~9.8 (41%)	
NVIDIA RTX 4090	Llama 3.1 8B (Q unspecified)	Ollama	450 (TBP) / ~423 (Used)	N/A	95.51	~15.6 (65%)	

NVIDIA RTX 4090	Mistral 7B Q4_K_M	llama.cpp	450 (TBP) / ~376 (Used)	N/A	100.43	~15.4 (64%)	
NVIDIA RTX 4090	Mistral 7B AWQ	TensorRT-LLM	450 (TBP) / ~398 (Used)	N/A	170.63	~17.3 (72%)	
NVIDIA RTX 4070 S	Mistral 7B (Q unspecified)	llama.cpp	~150 (Measured GPU)	N/A	50-60	N/A	
NVIDIA RTX 4060 Ti	Mixtral 8x7B 4-bit GPTQ	vLLM	180 (Max GPU load)	N/A	~17	N/A	
NVIDIA RTX 4060	DeepSeek-Coder 7B (Q4)	Ollama	115 (TBP) / ~105 (Used)	N/A	52.93	<8	
NVIDIA RTX 4060	Mistral 7B (Q4)	Ollama	115 (TBP) / ~104 (Used)	N/A	50.91	<8	
NVIDIA RTX 2070	Mistral 7B (Q unspecified)	llama.cpp	~100 (Measured GPU)	N/A	25-30	N/A	

AMD RX 6600	Mistral 7B (Q unspecified)	llama.cpp	N/A	N/A	33	N/A	
Apple M1 Max (GPU)	Llama3.2-1B (Q unspecified)	Unknown	~20 (SoC Est.)	N/A	~94 (2x ANE)	Unified	
Apple M4 Pro (GPU)	Llama3.2-1B (Q unspecified)	Unknown	~20 (SoC Est.)	N/A	~124 (2x ANE)	Unified	

N/A: Not Available in cited snippets. "Used" GPU power is estimated from TBP and reported utilization if direct measurement isn't available.

Table 3: GPU Power Consumption (TGP/TBP or Measured) and Performance for Image Generation

This table focuses on image generation tasks, primarily Stable Diffusion.

GPU Model	Task (Stable Diffusion)	Framework	GPU Power (W)	System Power (W)	Perf. (s/img)	VRAM (GB)	Source(s)
NVIDIA RTX 4090	SDXL 1024x1024 (20+5 steps)	Various	450 (TBP)	400-600 (Est.)	6.2 - 9.6	24	
NVIDIA RTX 4080	SDXL 1024x102	Various	320 (TBP)	N/A	7.2 - 14.2	16	

	4 (20+5 steps)						
NVIDIA RTX 3090	SDXL 1024x1024 (20+5 steps)	Various	350 (TBP)	N/A	10.56 - 17.3	24	
NVIDIA RTX 4070	Stable Diffusion (general)	Various	200 (TDP)	N/A	N/A	12	
Apple M3 Max	SDXL 1024x1024 (30 steps)	CoreML/Other	~30 (SoC Est.)	N/A	~70	Unified	
Apple M2 (Air)	SD 512x512 (50 steps)	CoreML	N/A	N/A	<18	Unified	
AMD RX 7800 XT	Stable Diffusion (general)	Various	263 (TBP) / ~230	N/A	N/A	16	

N/A: Not Available in cited snippets. System Power for RTX 4090 is a general estimate for intensive tasks.

Table 4: CPU & NPU Power Consumption and Performance for LLM Inference

This table covers CPU-only scenarios and emerging NPU-accelerated workloads.

Processor Model (CPU/NPU)	Model + Quantization	Framework	Comp. Power (W)	System Power (W)	Perf. (t/s gen)	RAM (GB)	Source(s)
AMD Ryzen 7940HS (CPU)	Llama 8B Q4_K_M	llama.cpp	N/A	N/A	11.96	<16	
AMD Ryzen 7940HS (CPU)	Llama 70B Q4_K_M	llama.cpp	N/A	N/A	1.36	>40	
AMD Ryzen 9 9950X (CPU)	Generic 8B model	Various	148 (Avg CPU Sys Load)	N/A	10-20 (Est.)	N/A	
Intel Core Ultra 5 125H (iGPU)	Various LLMs	Ollama	~3x less than CPU	N/A	Up to 30% > CPU	Unified	
Apple M1 Max (ANE)	Llama3.2-1B (Q unspecified)	ANEMLL	~1.8 (NPU Est.)	~10-12 (System)	47	Unified	
Apple M4 Pro (ANE)	Llama3.2-1B (Q unspecified)	ANEMLL	~2.8 (NPU Est.)	N/A	62	Unified	

Intel Core Ultra 7 155H (System)	Procyon AI Bench (GPU WinML)	N/A	N/A	23.9 (Avg Wall)	N/A (Score 258)	N/A	
AMD Ryzen AI Max+ PRO 395 (SoC)	Various LLMs	LM Studio	48 (Avg SoC)	N/A	Up to 12x LNL	Unified	

N/A: Not Available. Comp. Power for Apple ANE is estimated NPU contribution to SoC power. System power for M1 Max ANE from S27 (10-12W total system for ANE task vs 18-20W for GPU task).

Table 5: Impact of Quantization on LLM Inference (Representative Examples)

This table illustrates trade-offs for different quantization levels. Data is sparse for direct power comparison of Q-levels on the *same* hardware.

Base Hardware	**Model**	**Framework**	**Quantization**	**VRAM/RAM (GB)**	**Perf. (t/s)**	**GPU/Comp. Power (W)**	**System Power (W)**	**Source(s)**
NVIDIA RTX 4090	Mistral 7B	llama.cpp	Q4_K_M	~15.4	100.43	~376 (Est. Used)	N/A	
NVIDIA RTX 4090	Mistral 7B	TensorRT	AWQ (4-bit)	~17.3	170.63	~398 (Est. Used)	N/A	

NVIDIA RTX 4070	Llama 3 8B	Various	Q8_0	<12	Good speed	200 (TBP)	N/A	
NVIDIA RTX 4070	Mixtral 8x7B	Various	Q3_K_M (heavy)	~12 (fits)	Slower	200 (TBP)	N/A	
Apple M1 Air (GPU)	Llama 3.2 3B	LM Studio	GGUF Q4	Unified	24-27	N/A	18-20 (System)	
Apple M1 Air (ANE)	Llama 3.2 3B	ANEMLL	Unspecified	Unified	16-17	N/A	10-12 (System)	
Generic Consumer GPU	Llama 2 70B	Various	FP16	~140	Slower	High	Very High	(Training data, extrapolated for inference VRAM)
Generic Consumer GPU	Llama 2 70B	Various	Q4_K_M	~38-40	Faster (if VRAM-bound)	Moderate-High	High	

Note: Direct power draw comparisons for different GGUF quantizations (e.g., Q4 vs Q8) on the exact same hardware and task are scarce in the provided snippets. Speed and VRAM are more commonly reported. "Est. Used" power is a rough calculation from TBP and reported utilization.

Section 5: User Guidance for Measuring Local AI Footprint

To obtain accurate energy footprint estimates, particularly for Level 3 of the proposed tiered framework, users can employ a combination of software and hardware tools. This section provides guidance on recommended tools, best practices for measurement, and how to interpret the outputs to estimate task-specific energy consumption.

5.1 Recommended Measurement Tools

- 5.1.1 Software-Based Monitoring Tools:
  These tools read sensor data from system components. While convenient, they typically do not measure total system power from the wall and may not capture all component power draws or PSU inefficiencies.
    - **HWiNFO (Windows):**
        - A comprehensive hardware monitoring tool that can display "CPU Package Power," "GPU Power," and an estimated "System Power".
        - The "System Power" metric in HWiNFO is an *estimation* based on the sum of power readings from available component sensors. It does *not* represent the true total system power draw from the wall outlet, as it typically excludes power consumed by all motherboard components, some peripherals, and crucially, does not account for PSU conversion losses. Some user comparisons suggest HWiNFO's sum of component powers can be 25-35% lower than actual wall measurements.
        - HWiNFO is valuable for observing the relative power consumption of the CPU and GPU and understanding which components are most active during an AI task.
        - **Usage Guide:** Launch HWiNFO and select "Sensors-only" mode. In the sensor list, locate "CPU Package Power" (or similar, specific to CPU vendor) and "GPU Power" (or "GPU Chip Power Draw"). HWiNFO also allows logging of sensor data over time, which is useful for analyzing trends and calculating averages during a task.
    - **GPU-Z (Windows):**

        - A lightweight utility focused on providing detailed information about graphics cards, including GPU clock speeds, temperatures, fan speeds, and power consumption (often listed as "GPU Chip Power Draw" or similar metrics like "Board Power Draw").
        - It features a logging capability, allowing sensor data to be saved to a file for later analysis.
        - GPU-Z is primarily useful for GPU-specific power metrics and does not measure total system power.
    - `powermetrics` **(macOS):**
        - A powerful command-line utility exclusive to macOS that provides detailed, high-quality power-related measurements for CPU, GPU, and other system components and processes.
        - It can report `CPU ms/s` (CPU time used per second) and `GPU ms/s` (GPU time used per second) on a per-process or per-coalition (group of related processes) basis.
        - Requires administrator privileges (`sudo`) to run. A common command for per-process CPU and GPU usage is: `sudo powermetrics –samplers tasks –show-process-coalition –show-process-gpu -n 1 -i 5000` This command takes one sample over a 5-second interval. The `-i` value (interval in milliseconds) can be adjusted.
        - The `mach power` command can combine `powermetrics` with `rapl` (Running Average Power Limit) for profiling browser power usage, which might be adaptable for other applications.
    - `asitop` **(macOS Apple Silicon):**
        - A command-line interface (CLI) tool specifically for Apple Silicon Macs, which utilizes `powermetrics` as its backend to display real-time utilization and power data.
        - It shows CPU power (E-cluster and P-cluster), GPU power, ANE (Apple Neural Engine) utilization (indicated by its power consumption), a chart of CPU/GPU power over time, peak power, and rolling average power.
        - Requires `sudo` to run due to its reliance on `powermetrics`.
    - **Framework-Specific Logging &** `nvidia-smi`**:**
        - Some AI frameworks or their associated tools might offer built-in performance logging (e.g., tokens/second in Jan AI), but direct power measurement is rare.
        - For NVIDIA GPUs on Windows and Linux, the `nvidia-smi` (NVIDIA System Management Interface) command-line utility can report

real-time GPU power draw, utilization, temperature, and memory usage. This is a reliable source for GPU-only power.

- 5.1.2 Hardware-Based Monitoring Tools:
  These devices measure power consumption externally, providing the most accurate figure for total system draw.

    - **Kill A Watt Meter (or similar plug-in power meters):**
        - These devices are plugged directly into a wall outlet, and the PC (or any appliance) is then plugged into the meter.
        - They measure the true total system power consumption from the wall, thereby including the power used by all internal components (CPU, GPU, RAM, motherboard, drives, fans) as well as the inefficiency losses of the PSU. This is the most comprehensive measurement for overall energy footprint.
        - Most models can display instantaneous power (Watts), cumulative energy consumption (kWh), voltage (V), current (Amps), and frequency (Hz).
        - **Usage Guide:**
            1. Plug the Kill A Watt meter into a standard wall outlet.
            2. Plug the computer's power cord into the Kill A Watt meter.
            3. Turn on the computer and allow it to boot to its normal idle state.
            4. Observe the wattage reading for idle power.
            5. Run the desired AI task.
            6. Monitor the wattage reading throughout the task. If it fluctuates, take readings at regular intervals or note the average if the meter provides one.
            7. Once the task is complete, note the final cumulative kWh reading if measuring total energy for a longer task, or use average wattage and duration to calculate energy.

5.2 Best Practices for Accurate Measurement

To ensure that power measurements are reliable and comparable, several best practices should be followed:

- **Isolate Workloads:** Before starting measurements for an AI task, close all unnecessary background applications, browser tabs, and services. This helps

ensure that the measured power consumption is primarily attributable to the AI workload and not confounded by other system activities.

- **Consistent Settings:** For comparative analysis (e.g., evaluating different models, quantizations, or framework settings), ensure all other parameters are kept constant. This includes using the exact same AI model version, quantization level, input prompt, generation parameters (number of tokens, image resolution, inference steps), software framework versions, and even driver versions if possible.
- **Warm-up Period:** Hardware components, especially GPUs and CPUs, can exhibit different power characteristics when cold versus when at a stable operating temperature. For tasks lasting more than a few minutes, allow the system to run the workload for a brief period (e.g., 5-10 minutes) to "warm up" before starting formal measurements. This ensures readings reflect steady-state operation.
- **Measurement Duration:**
    - For short, discrete AI tasks (e.g., generating a single image, inferring a short text response), measure the power consumption over the entire duration of the task.
    - For longer or continuous tasks (e.g., processing a large batch of images, running an LLM for an extended chat session), measure power over a representative period (e.g., 10-30 minutes) once the workload has stabilized.
- **Accounting for Idle vs. Load:**
    - **Baseline Idle Power:** Before running the AI task, measure the system's idle power consumption. This should be done with the operating system loaded and any AI frameworks or necessary software also loaded but not actively processing. This establishes a baseline.
    - **Load Power:** Measure the system's power consumption while the AI task is actively running.
    - **Task-Specific Power Estimation:** A common way to estimate the power attributable specifically to the AI task is to subtract the baseline idle power from the load power: $Power_{task} = Power_{load} - Power_{idle}$. The energy consumed by the task itself can then be estimated as $Energy_{task} = Power_{task} \times Duration_{task}$.
    - **Total Operational Energy:** For a more complete picture, especially if the system is dedicated to AI tasks or left on for extended periods around tasks, the total energy would be $Energy_{total} = Power_{load} \times Duration_{task} + Power_{idle} \times Duration_{idle}$. The idle power bug with Ollama/CUDA is a good example of why idle states matter.
- **Logging and Averaging:** Power consumption can fluctuate during an AI task.

    - If using software tools with logging capabilities (e.g., HWiNFO, GPU-Z), log the relevant power sensors at a reasonable interval (e.g., every 1-5 seconds) throughout the task. The logged data can then be used to calculate an average power draw.
    - If using a manual device like a Kill A Watt meter that shows instantaneous wattage, take readings at regular intervals (e.g., every 30 seconds or every minute) and average them, or use the meter's cumulative kWh reading if running the task for a sufficiently long period.

5.3 Interpreting Tool Outputs and Estimating Task Energy

Understanding what the different tools report and how to use that data is key.

- **From Software Tools (HWiNFO, GPU-Z, `nvidia-smi`):**
    - These tools primarily provide **component-level power data** (e.g., "GPU Power," "CPU Package Power").
    - Summing the reported CPU and GPU power gives a *minimum estimate* of the power consumed by these two active components. It does not include power for RAM, motherboard, storage, fans, or PSU losses. Therefore, this sum will be lower than the actual total system power draw from the wall.
    - This data is most useful for *relative comparisons* (e.g., "Does model A use less GPU power than model B on my system under these settings?") or for understanding which component is the primary consumer during a task.
    - Avoid using the sum of software-reported component powers as an absolute measure of total system energy consumption unless no other measurement method is available, and if so, clearly state this limitation. An approximate adjustment for other components and PSU inefficiency (e.g., adding 30-50W for other components and then dividing by PSU efficiency like 0.85-0.90) might provide a rougher estimate of wall power, but this is highly speculative.
- **From Kill A Watt Meter (or similar wall plug meter):**
    - This provides the **true total system power draw (Watts)** from the wall.
    - To calculate energy for a task:
        - Note the average wattage reading (Pavg) during the AI task.
        - Measure the duration of the task (T) in hours.
        - Calculate energy: $\text{Energy(Wh)} = \text{Pavg(W)} \times \text{T(h)}$.
        - To convert to kilowatt-hours (kWh), divide by 1000: $\text{Energy(kWh)} = \text{Energy(Wh)}/1000$.

        - Example: An AI task runs for 15 minutes (0.25 hours). The Kill A Watt meter shows an average system power draw of 280W during this task. Energy=280W×0.25h=70Wh=0.070kWh.
    - If the meter provides a cumulative kWh reading, you can note the reading before and after the task; the difference is the energy consumed.
- **Relating Performance to Energy (Energy Efficiency):**
    - Once total energy for a task is known, and the work output is quantified (e.g., number of tokens generated, number of images created), energy efficiency metrics can be calculated:
        - Tokens per Watt-hour (tokens/Wh) = Total Tokens Generated / Total Energy (Wh)
        - Images per Watt-hour (images/Wh) = Total Images Created / Total Energy (Wh)
        - Alternatively, Joules per token (since 1Wh=3600J): Joules/token = (Total Energy (Wh) × 3600) / Total Tokens Generated.
    - These normalized metrics are crucial for comparing the true efficiency of different setups, as they account for both power draw and speed.

The discrepancy between software-reported component power and actual wall-measured system power is significant. Software tools are convenient for relative analysis and identifying power-hungry components. However, for an accurate assessment of the total energy footprint and for calculating operational costs or environmental impact, direct measurement of wall power using a device like a Kill A Watt meter is strongly recommended. If only software data is available, it should be treated as an underestimate, and this limitation must be acknowledged.

Table 6: Comparison of Power Measurement Tools

Tool Name	Type	Key Metrics Measured	Typical Granularity	Pros	Cons	Ease of Use	Primary Use Case for AI Footprinting

HWiNFO	Software (Win)	CPU Pkg Power, GPU Power, Est. System Power, Temps, Clocks, Util.	Real-time, Logging	Comprehensive component data, logging, customizable.	"System Power" is an estimate, not true wall power; doesn't capture PSU loss or all minor components. Accuracy varies.	Moderate	Detailed component-level analysis, relative power changes, identifying bottlenecks. Rough estimate of system power if no wall meter.
GPU-Z	Software (Win)	GPU Power (Chip/Board), VRAM Usage, GPU Clock, Temp, Util.	Real-time, Logging	Lightweight, good for GPU-specific data, logging.	GPU-only, not system power.	Easy	GPU-specific power monitoring and logging.
powermetrics	Software (macOS)	CPU ms/s, GPU ms/s (per process/coalition), Wakeups, C-states, P-states.	Sample-based	Detailed macOS-specific power events, scriptable, good for process-level attribution.	macOS only, command-line, output can be verbose, indirect power (ms/s needs interpret	Advanced	Detailed analysis of CPU/GPU activity by process on macOS.

					ation for Watts).		
asitop	Software (macOS)	CPU Power, GPU Power, ANE Power, Power Chart, Peak Power, Avg Power, Util.	Real-time CLI	Apple Silicon specific, uses powermetrics backend, good overview of SoC components.	Apple Silicon macOS only, sudo required, some metrics are guesstimates.	Moderate	Real-time monitoring of Apple Silicon SoC power distribution.
nvidia-smi	Software (Win/Lin)	GPU Power Draw, Util., Temp, Mem Usage.	Real-time CLI	Official NVIDIA tool, accurate for GPU power.	NVIDIA GPUs only, GPU-only, command-line.	Moderate	Quick check or scripting of NVIDIA GPU power and utilization.
Kill A Watt Meter	Hardware	True Wall Power (W), Cumulative Energy (kWh), Voltage (V), Current (A), Frequency (Hz).	Real-time display	Measures actual total system power from wall (most accurate for footprint), includes PSU loss.	Requires physical device, manual logging for averages unless meter has logging, measure	Easy	Accurate measurement of total system energy consumption for specific tasks or overall usage.

					s entire plugged-in system (PC + monitor if on same plug).		Ideal for Level 3 estimation.

Sources for Table 6: S30, S31, S32, S33, S34, S65, S66, S67, S68, S69, S70, S81 (Jan for perf.), S178, S179, S191, B26, B27, B28, B29, B30, B41, B69, B84, B130.

Section 6: Considerations for a Footprint Estimation Calculator

Translating the research findings and methodologies into a practical footprint estimation calculator requires careful consideration of user interface (UI), user experience (UX), data presentation, and the communication of inherent uncertainties. The goal is to create a tool that is both accessible to a broad range of users and provides meaningful, transparent estimates.

6.1 UI/UX Recommendations for Simplicity and Clarity

A successful calculator should guide the user effectively, regardless of their technical depth.

- **Tiered Input System:**
    - The calculator should ideally default to **Level 1 estimation**. This could involve simple dropdown menus where users select their general hardware category (e.g., "Mid-Range GPU Desktop," "Standard Laptop with iGPU/NPU" from Table 1) and the type of AI task (e.g., "LLM Inference - 7B Model Q4," "Stable Diffusion - 512x512 Image"). This provides an immediate, albeit coarse, estimate.
    - A clear option should allow users to progress to **Level 2 estimation**. Here, input fields would appear for specific components (e.g., GPU model, CPU model), AI model details (name, parameter size), and quantization level. Autocomplete suggestions for common hardware and models could enhance usability.
    - A distinct section or mode should cater to **Level 3 estimation**, where users input their own measured data (idle power, load power, task duration, performance metrics like tokens generated). This section would perform the energy calculation based on user data.

- **Clear Labeling and Tooltips:** All input fields and output results should be clearly labeled. Technical terms such as TGP (Total Graphics Power), TBP (Total Board Power), quantization types (e.g., GGUF Q4_K_M), tokens/second, and inference steps should be explained concisely through tooltips or embedded information icons. This ensures users understand what data is being asked for and what the results signify.
- **Visual Feedback:** Where appropriate, simple visual aids can enhance understanding. For instance, a bar chart could illustrate the estimated power range, or if component contributions are estimated, a pie chart could show an approximate breakdown (e.g., GPU vs. rest of system). However, over-complicating visuals should be avoided to maintain clarity.
- **Mobile-Friendly Design:** Given that users might access such a tool from various devices, a responsive design that adapts well to different screen sizes (desktop, tablet, mobile) would improve accessibility.

6.2 Presenting Uncertainty, Assumptions, and Data Sources

Transparency is paramount for a credible estimation tool, especially given the inherent variability in local AI energy consumption.

- **Acknowledge Estimation Nature:** The calculator must clearly communicate that its outputs are *estimates* and not precise predictions. Factors like specific system configuration, ambient temperature, background processes, and minor variations in hardware or software can influence actual power draw. The concept of uncertainty in energy forecasts, as highlighted in fields like solar energy, is analogous; exact prediction is impossible.
- **Provide Uncertainty Ranges:** Instead of displaying a single numerical value for power or energy, the calculator should provide a plausible range (e.g., "Estimated system power: 250W - 350W," "Estimated task energy: 120Wh - 180Wh"). This reflects the inherent variability and manages user expectations regarding precision. This approach is similar to how P-values (e.g., P50, P90) are used in energy yield forecasting to quantify confidence levels and downside/upside scenarios.
- **Explicitly List Assumptions:** Key assumptions underpinning the estimations should be clearly stated. Examples include:
    - "Estimates for Level 1 and 2 assume typical CPU load during GPU-bound AI tasks."
    - "System power estimates based on component data assume an average PSU efficiency of X% (e.g., 85% for 80+ Gold)."
    - "Performance metrics (tokens/second, images/second) are based on averages from public benchmarks and may vary."

    - "Idle power is assumed to be Y Watts for this hardware category unless specified by the user."
- **Cite Data Sources (General):** While detailed citations for every data point might clutter the UI, a general statement about the origin of the data used for default profiles can build trust (e.g., "Estimations are based on aggregated data from manufacturer specifications, technical reviews (e.g., TechPowerUp, Phoronix, ComputerBase, KitGuru, Guru3D), and community benchmarks.").

6.3 Optional Extension: Incorporating Carbon Intensity

To provide a more direct link to environmental impact, the calculator could offer an optional feature to estimate the carbon dioxide (CO_2) emissions associated with the energy consumed by the AI task. This makes the footprint more tangible.

- **Methodology:** The fundamental calculation, based on the Software Carbon Intensity (SCI) specification, is: CO_2 Emissions (gCO_2eq)=Energy Consumed (kWh)×Electricity Carbon Intensity (gCO_2eq/kWh) .[1]
- **Data Sources for Carbon Intensity:** The calculator would need access to regional electricity carbon intensity data.
    - **EPA eGRID (United States):** The US Environmental Protection Agency's Emissions & Generation Resource Integrated Database (eGRID) provides annual average CO_2, CH_4, and N_2O emission rates per MWh for electricity generation at national, state, and subregional levels. The data can be used to derive gCO_2eq/kWh figures. For example, the 2022 eGRID data (released in 2024) shows a US national average total output emission rate of 823.1 lb CO_2/MWh, which converts to approximately 373.3 gCO_2/kWh (using 1 MWh = 1000 kWh, 1 lb = 453.592 g). eGRID also provides marginal emission rates.
    - **Electricity Maps API/Website (Global):** Electricity Maps offers an API and website providing real-time, historical, and forecasted carbon intensity data (gCO_2eq/kWh) for numerous countries and regions worldwide. Their API can distinguish between 'direct' (operational) and 'lifecycle' emission factors.
    - **UK Carbon Intensity API (Great Britain):** National Grid ESO provides an API for current and forecasted carbon intensity (gCO_2/kWh) for Great Britain and its 14 DNO regions. A typical GB value might be around 155 gCO_2/kWh, but it varies significantly by region and time.
    - **LowCarbonPower.org (Global):** This site provides aggregated country-specific carbon intensity data. For 2024, it reports Germany at 325.7 gCO_2eq/kWh, India at 635.84 gCO_2eq/kWh, and a world average of

421.57 gCO2eq/kWh. (Note: Nowtricity reports Germany at 321 gCO2eq/kWh for 2024).

- **User Input:** The user would select their country and, if available and relevant, their specific region or state.
- **Output:** The calculator would display the estimated CO2 emissions for the AI task in grams or kilograms of CO2 equivalent.
- **Caveats and Nuances:**
    - It should be clearly stated that grid carbon intensity varies significantly by location and over time (hourly, daily, seasonally) due to changes in the electricity generation mix (e.g., availability of renewables). The figures used are often annual averages or real-time snapshots/forecasts and may not reflect the exact intensity at the moment the AI task was run unless live data is integrated.
    - The distinction between operational (direct) emissions from power generation and lifecycle emissions (which include embodied carbon of power plants, fuel extraction, etc.) should be clarified if the data source provides this (Electricity Maps does). The SCI methodology primarily focuses on operational emissions from electricity use for the 'O' component, and separately accounts for embodied hardware carbon ('M'). This calculator feature would focus on the 'O' component's carbon impact.

Table 7: Average Electricity Carbon Intensity by Region (Selected Examples for Calculator Default)

This table provides sample average carbon intensity values that could be used as defaults in the calculator. Users should ideally be able to select their specific region for more accuracy. Values are in gCO2eq/kWh.

Country/Region	Carbon Intensity (gCO2eq/kWh)	Year of Data	Source & Notes
United States (Avg)	~373	2022 (eGRID)	EPA eGRID (823.1 lb CO2/MWh total output). Varies significantly by eGRID subregion.

Germany	~321 - 326	2024	Nowtricity, LowCarbonPower.org. Electricity Maps provides real-time/historical.
United Kingdom (GB Avg)	~150 - 200 (highly variable)	Real-time /Avg	UK Carbon Intensity API. Example: 155 gCO2/kWh (snapshot). Varies by region and time.
India	~636	2024	LowCarbonPower.org. Electricity Maps provides real-time/historical (e.g., ~697 gCO2eq/kWh snapshot).
World Average	~422	2024	LowCarbonPower.org.

Note: These are illustrative values. A live calculator should ideally pull from an API like Electricity Maps for more current and granular data or allow user input. The basis (direct vs. lifecycle) should be consistent.

By implementing these UI/UX considerations and transparently handling data and its uncertainties, a footprint estimation calculator can be a valuable tool for raising awareness and promoting more energy-conscious local AI practices.

Section 7: Summary of Factors, Assumptions, and Key Recommendations

This report has explored the multifaceted challenge of estimating the energy footprint of locally run AI models. The analysis has covered key influencing factors, methodologies for estimation, specific hardware and task energy profiles, user measurement guidance, and considerations for a practical estimation tool. This concluding section recapitulates the critical variables, summarizes underlying assumptions, and offers key recommendations for users, developers, and future research in this domain.

7.1 Recapitulation of Critical Variables Affecting Energy Footprint

The energy consumption of local AI is determined by a complex interplay of factors:

- **Hardware:** The specific GPU (manufacturer, model, VRAM, architecture), CPU (class, core count, architecture), presence and utilization of NPUs, system RAM (capacity, speed), and PSU efficiency are primary determinants. Newer hardware generations often offer better performance per Watt, but absolute power draw for high-end components remains significant.
- **AI Model:** Model size (parameter count), architecture (e.g., Transformer, MoE), and particularly the level of quantization (e.g., FP16, INT8, various GGUF levels) critically affect computational load, memory requirements, and thus energy use.
- **Software:** The choice of inference engine (e.g., llama.cpp, TensorRT-LLM, Ollama), UI frontend, and versions of drivers and libraries can lead to substantial differences in performance and efficiency.
- **Utilization:** The type of AI task (LLM inference vs. image generation), its duration and intensity (prompt length, output tokens, image resolution, batch size), and system idle power (especially with frameworks that keep hardware active) all contribute to the total energy consumed.

7.2 Summary of Underlying Assumptions in Estimation Methodologies

The proposed tiered estimation framework relies on several assumptions, the transparency of which is crucial for user understanding:

- **Use of Typical/Average Values:** Level 1 and Level 2 estimates are based on aggregated data from benchmarks and specifications, representing typical or average performance and power draw. Actual user experience can vary.
- **PSU Efficiency:** When estimating wall power from component power data, an assumption about PSU efficiency must be made (e.g., 85-90%). This is an approximation.
- **Proxy Data:** Due to a scarcity of AI-specific system-level power benchmarks, gaming benchmarks are sometimes used as a proxy for general high-load conditions. This is an acknowledged limitation, as AI workloads can differ.
- **Controlled Environment for Benchmarks:** The data used for profiles often comes from benchmarks run in controlled environments, which may not fully reflect diverse real-world user setups and background activity.

7.3 Key Recommendations for Users, Developers, and Future Research

- **For Users:**
    - **Hardware Selection:** When choosing hardware for local AI, consider not just raw performance but also performance per Watt. For laptops, NPUs and efficient iGPUs (like those in Apple Silicon, Intel Core Ultra, AMD Ryzen APUs) can offer a good balance for lighter AI tasks. For desktops,

evaluate GPUs based on the types of models you intend to run and their VRAM requirements, keeping in mind that higher-end cards have substantial power needs.
    - **Task Optimization:**
        - Select appropriate model quantization levels that balance quality with VRAM/RAM capacity and desired speed. Lower precision often means faster inference on constrained hardware.
        - Close unnecessary background applications to reduce system load and ensure more power is available for the AI task, also leading to cleaner measurements.
        - If using frameworks like Ollama, be mindful of potential idle power issues and stop the service when not in use if power saving is critical.
    - **Measurement:** For the most accurate understanding of your specific footprint, measure your system's power consumption at the wall using a device like a Kill A Watt meter during typical AI workloads.
    - **Carbon-Aware Usage:** If using the carbon intensity extension of an estimator, consider scheduling non-critical AI tasks for times when grid carbon intensity is lower (e.g., higher renewable energy production), if such data is available for your region.
- **For Developers (of AI Models and Frameworks):**
    - **Prioritize Efficiency:** Focus on optimizing AI models and inference engines not only for speed and accuracy but also for power and energy efficiency. This includes exploring novel architectures, advanced quantization techniques (like the principles behind BitNet b1.58), and efficient kernel implementations. The trend towards custom silicon and chiplet-based designs for AI aims to address GPU power consumption limitations. Algorithms like L-Mul, claiming up to 95% energy reduction for certain operations, warrant investigation.
    - **Improve Power Reporting:** Integrate more granular and accessible power reporting features directly into AI frameworks and tools. This would empower users to better understand the energy implications of different settings and models.
    - **Minimize Idle Power:** Actively work to reduce the idle power consumption of AI software. Frameworks should release hardware resources and allow components like the PCIe bus to enter low-power states when not actively processing. Lazy initialization of CUDA contexts is one such strategy.
- **For Future Research:**

- **Comprehensive Benchmarking:** There is a pressing need for more standardized, publicly available benchmarks that measure total system (wall) power consumption alongside performance (tokens/second, images/minute) for a wide range of local AI workloads. These benchmarks should cover diverse hardware (GPUs, CPUs, NPUs from all major vendors), AI models (various sizes and architectures), quantization levels (FP16, INT8, popular GGUF types like Q4_K_M, Q5_K_M, Q8_0, etc.), and software frameworks (Ollama, llama.cpp, TensorRT-LLM, ComfyUI, Automatic1111, etc.).
- **Quantization Impact Studies:** Dedicated research is needed to systematically evaluate the precise impact of different quantization methods on both instantaneous power draw and total energy per task across various hardware architectures.
- **Adaptive Power Scaling:** Investigate and develop adaptive power scaling mechanisms within AI frameworks that can dynamically adjust hardware power states (e.g., GPU/CPU clock frequencies, active cores) based on the real-time demands of the AI task, aiming to optimize for energy efficiency without undue performance compromise.
- **Lifecycle Assessment Data:** Greater availability and standardization of embodied energy/carbon data for hardware components would improve the accuracy of comprehensive lifecycle assessments for local AI.

The journey towards efficient local AI involves navigating a complex "efficiency frontier" where performance, VRAM capacity, model capability, and energy consumption are all interlinked. Newer hardware and software often push this frontier, offering more capability per Watt. However, the "best" configuration is subjective and depends on individual user priorities—be it raw speed, energy conservation, or output quality.

Ultimately, the ease of deploying and running AI models locally should not overshadow the associated energy costs. Fostering a "Sustainable Local AI" mindset, through awareness, better tools for estimation, and a commitment to efficiency from both users and developers, is crucial. As local AI becomes more powerful and pervasive, a proactive approach to managing its energy footprint will benefit individual users through reduced costs and better device thermals, and contribute to more responsible technology use on a larger scale. This report aims to provide a foundational step in that direction.