

Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)

The GPU Paradox: Why Your GPU Might Be Slower Than Your CPU for LLM Inference

Mubashir Rahim

Follow

4 min read · May 13, 2025

5

Introduction: A Counter-Intuitive Discovery

When running large language models (LLMs), conventional wisdom suggests pushing as much computation to the GPU as possible. After all, that's why we invest in powerful graphics cards like the RTX 3070, right?

But what if I told you that for certain workloads, your CPU might outperform your dedicated GPU by a factor of 13x? This counter-intuitive finding emerged from extensive benchmarking I conducted with Phi-3.5, and the results challenge everything we thought we knew about optimal LLM deployment.

The Experiment: Testing GPU/CPU Layer Distribution

Modern LLMs like Phi-3.5 are composed of multiple neural network layers (in this case, 32 layers) that can be distributed between your GPU and CPU. The Ollama framework provides a convenient parameter `OLLAMA_GPU_LAYERS` to control this distribution.

One subscription. Endless stories. Become a Medium member for unlimited reading.

Upgrade now

To find the optimal configuration, I created a testing framework that evaluates performance across different GPU/CPU layer allocations:

- 32 GPU / 0 CPU layers: 100% GPU utilization
- 24 GPU / 8 CPU layers: 75% GPU, 25% CPU
- 16 GPU / 16 CPU layers: 50% GPU, 50% CPU
- 8 GPU / 24 CPU layers: 25% GPU, 75% CPU
- 0 GPU / 32 CPU layers: 100% CPU utilization

For each configuration, I tested three different input sizes:

- Short: ~100 tokens
- Medium: ~1,000 tokens
- Long: ~5,000 tokens

The Results: Input Size Changes Everything

The results were nothing short of astonishing:

Processing Time (seconds) — Lower is Better

Configuration	Short (100 tokens)	Medium (1000 tokens)	Long (5000 tokens)
32 GPU / 0 CPU	18.96s	2.18s	1.11s
24 GPU / 8 CPU	11.65s	2.13s	1.13s
16 GPU / 16 CPU	18.91s	1.55s	1.16s
8 GPU / 24 CPU	6.22s	1.16s	1.15s
0 GPU / 32 CPU	1.46s	1.16s	1.22s

Processing Time (seconds) — CPU/GPU

This reveals a striking pattern: The optimal GPU/CPU distribution changes dramatically based on input size.

- For short inputs (100 tokens): CPU-only processing is 13x faster than full GPU
- For medium inputs (1000 tokens): 8 GPU/24 CPU split performs best
- For long inputs (5000 tokens): Full GPU utilization is optimal

Tokens Per Second — Higher is Better

The efficiency metrics tell the same story:

Configuration	Short (100 tokens)	Medium (1000 tokens)	Long (5000 tokens)
32 GPU / 0 CPU	5.28	459.06	4,493.52
24 GPU / 8 CPU	8.58	469.89	4,434.48
16 GPU / 16 CPU	5.29	644.03	4,313.66
8 GPU / 24 CPU	16.09	862.90	4,340.40
0 GPU / 32 CPU	68.70	860.84	4,096.05

Tokens Per Second — Higher is Better— CPU/GPU

The performance gap is staggering: CPU-only processing is 13x more efficient than GPU for small inputs, while GPU becomes substantially more efficient for large inputs.

Understanding the GPU Paradox

What explains this counter-intuitive behavior? Three primary factors:

1. Initialization and Transfer Overhead

GPUs require significant initialization time and data transfer between host memory and GPU memory. For small inputs, this overhead dominates the actual computation time:

Total Time = Initialization Time + Transfer Time + Computation Time

For small inputs, Initialization Time + Transfer Time >> Computation Time, making GPU acceleration counterproductive.

2. Parallelism Benefits Scale with Input Size

GPUs excel at parallel computation, but this advantage only becomes significant when there's enough data to process in parallel. Small inputs simply don't provide enough work to justify the GPU's initialization costs.

3. Memory Bandwidth Bottlenecks

Moving data between CPU and GPU memory creates bottlenecks. This is why balanced configurations (like 16/16 or 8/24) often perform better than extreme for medium-sized inputs.

Real-World Impact: Processing Large Files

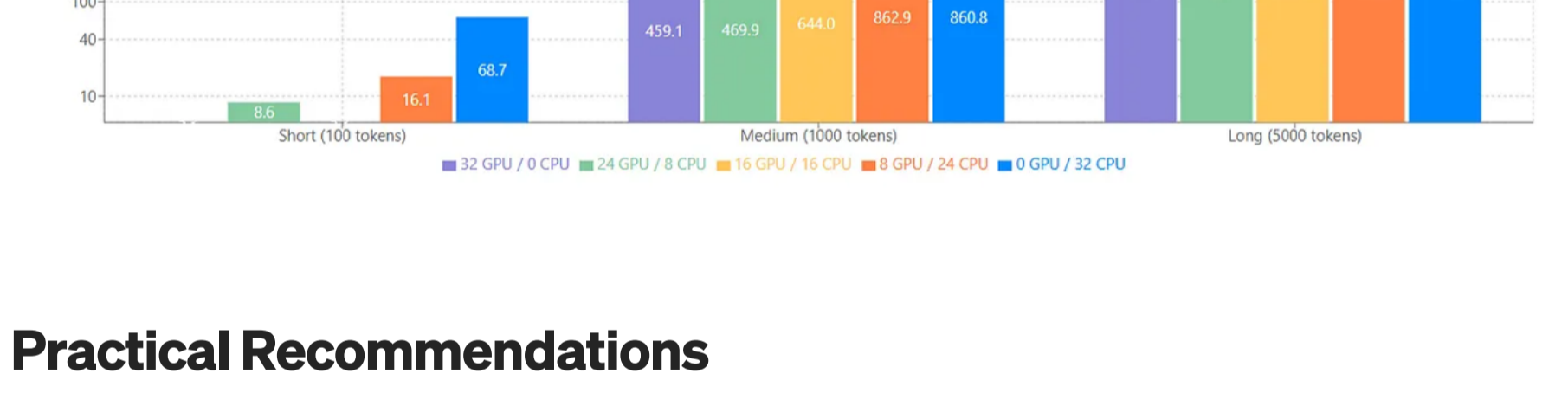
The practical implications become clear when we look at estimated processing times for a large 132,739 token file:

Estimated Processing Time (minutes) for 132K Tokens — Lower is Better

Configuration	Based on Short	Based on Medium	Based on Long
32 GPU / 0 CPU	419.4 min	4.8 min	0.5 min
24 GPU / 8 CPU	257.8 min	4.7 min	0.5 min
16 GPU / 16 CPU	418.3 min	3.4 min	0.5 min
8 GPU / 24 CPU	137.5 min	2.6 min	0.5 min
0 GPU / 32 CPU	32.2 min	2.6 min	0.5 min

Estimated Processing Time (minutes) for 132K Tokens— CPU/GPU

These estimates reveal a dramatic shift in optimal configuration as input size increases. Using the wrong configuration can result in a processing time difference of 13x!



Practical Recommendations

Based on these findings, here are actionable recommendations for optimizing Phi-3.5 inference on similar hardware:

Technical Implementation: Testing Framework

For those interested in conducting similar tests, I've developed a Python script that automatically benchmarks different GPU/CPU layer configurations across various input sizes. The framework includes:

- Progressive testing from short to medium to long inputs
- Multiple runs per configuration for statistical significance
- Warmup runs to avoid measuring initialization costs
- Timeout handling to prevent hanging
- CSV result export for further analysis

The core testing logic looks like this:

```
for prompt_size in ["short", "medium", "long"]:
    for gpu_layers in [32, 24, 16, 8, 0]:
        # Perform warmup run if enabled
        if WARMUP_RUN:
            test_configuration(gpu_layers, "warmup", prompt_size)

        # Run multiple tests and calculate average
        times = []
        for run in range(1, TEST_RUNS + 1):
            elapsed_time, tokens = test_configuration(gpu_layers, run, prompt_size)
            times.append(elapsed_time)

        avg_time = sum(times) / len(times)
        results.append([gpu_layers, 32-gpu_layers, avg_time, token_count])
```

Conclusion: Input Size Matters Most

The most important takeaway from this research is that input size is the primary factor in determining the optimal GPU/CPU layer allocation. This contradicts the common assumption that maximal GPU utilization is always best.

For Phi-3.5 on an RTX 3070, we've seen that:

- Small inputs: CPU-only processing is dramatically faster
- Medium inputs: A balanced approach works best
- Large inputs: Full GPU utilization is optimal

These findings highlight the importance of understanding the specific characteristics of your workload rather than relying on general assumptions about GPU acceleration. By matching your configuration to your typical input size, you can achieve dramatically better performance without any hardware upgrades.

Have you observed similar patterns with other models or hardware configurations? I'd love to hear about your experiences in the comments below.

Artificial IntelligenceMachine LearningGPU ComputingLim OptimizationDeep Learning

5

Written by Mubashir Rahim

1 follower · 3 following

Multidisciplinary engineer with a passion for design and innovation. CAD ,AI ,Machine Learning And Deep Learning

Follow

More from Mubashir Rahim

When AI Forgets Who It Is: My Claude Had an Identity Crisis (And...
Welcome to 2025, where even AI models need therapy.
Dec 10, 2025

Unveiling the Evaluation Metrics: Mastery of Machine Learning...
Unveiling the Gamer's Guide to Mastering Machine Learning Performance. Join us on a...
Aug 17, 2023

See all from Mubashir Rahim

Recommended from Medium

In AI Advances by Delano Pirard
SLM vs LLM: Why 'Bigger is Better' is Dead in 2026
LLM's predicts 3x more SLM usage than GPTs by 2027. Bayer gained +40% accuracy...
4d ago · 557 · 9

Shubham Singh
VL-JEPA: The Future of Vision-Language AI is Here, and It's...
A deep dive into Meta's groundbreaking research that could reshape AI from chatbot...
6d ago · 72 · 2

Omar Faroque
Why SIMD and SIMT in GPU?
Let's say, we want to increase the brightness of a gray scale image, that image is 1MB size.
Aug 24, 2025

In my_ai/ml by Sanjay Basu, PhD
The Split Personality of AI Inference
How LLM-D-Parallel Runs Are Rewriting the Rules of Model Inference
Nov 25, 2025

Tyler Frink
Tensors in Large Language Models
Oct 31, 2025 · 1

Thinking in AI, Philosophy, Investing, and Mind
Ray + vLLM
Architecture Diagram
Nov 22, 2025

See more recommendations