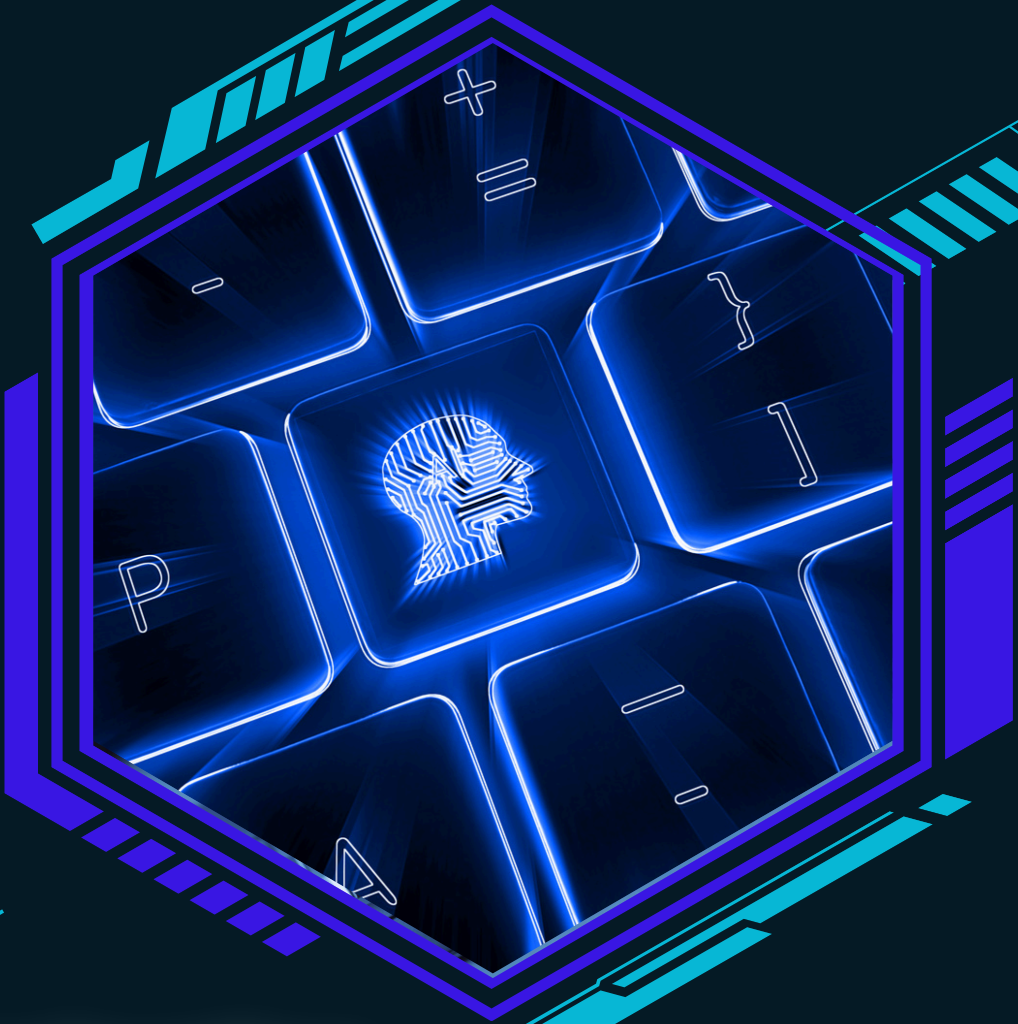
LLMWARE.AI



Accelerating AI-POWERED PRODUCTIVITY WITH AI PCS

PREPARED BY: LLMWARE.AI

8/27/2024

AI-POWERED PRODUCTIVITY REVOLUTION

We stand at the brink of an AI-powered productivity revolution where we can put the power of AI models in everyone’s hands as a personal and business productivity tool. Business users and consumers will be able to access and to use a wide range of AI-enabled workflows with the safety and security of self-hosting and private deployment on their personal devices thanks to a new generation of personal computers called AI PCs that are designed to power efficient AI acceleration and handle AI tasks locally.

In this white paper, we will show why AI PCs are poised to play a significant role in the miniaturization and decentralized deployment of LLM technologies in the next several years. We will evaluate and compare the performance of several popular AI PCs - Mac M1, M3 and Dell Ultra 9/Intel - for inference speed and use in Generative AI. In addition, we will demonstrate how the combination of the right software optimization such as AI framework with the corresponding model compilers that are mapped to the specific hardware produces a significant difference in inference speed and usability of the hardware.

WHY

AI PCs will fuel decentralization of AI

COMPARE

Inference Speed of Mac M1, M3 and Dell Ultra 9/Intel AI PCs

Laptop Inference Speed Test Result

Our tests found that the Dell/Intel laptop[1] performed up to 51% faster compared to a Mac M1 and up to 37.5% faster compared to a Mac M3 when the appropriate AI framework and inference optimization library is used, in this case OpenVINO for Dell/Intel and llama.cpp for Mac. Based on our findings, when paired with the right software optimization technique, Intel-based AI PCs will enable a wide range of potential use cases and generative AI applications privately, locally and cost-effectively over the next 12-24 months and provide the best total value and performance overall.

Dell Ultra/Intel Laptop was up to 51% faster than Mac M1 and 37.5% faster than Mac M3

Correct software optimization technique is key to delivering fastest inference

AI PC OPPORTUNITY

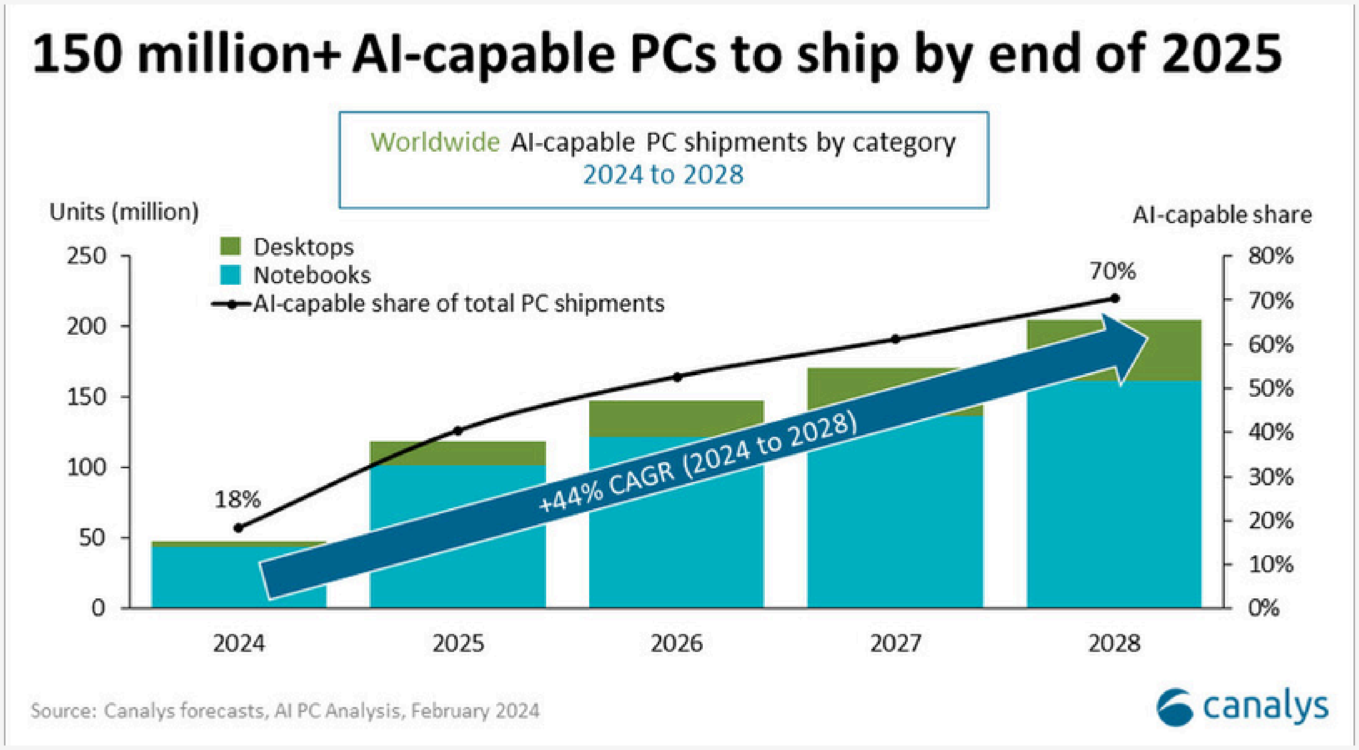
AI PCs can run AI workloads

Since the launch of ChatGPT in November 2022, API-based large language models (>100 billion parameters) have dominated the AI market. Recently, however, these two emergent trends have converged to create the perfect scenario to further decentralized deployment of AI:

- Small specialized models – high-quality model bases specialized for domain-specific, general business or productivity assistant purposes; and
- Quantization and packaging technologies – compressing the size of large models using bit-level quantization to enable larger models in smaller spaces without diminishing the efficacy of the model.

Coupled with next generation laptop-level GPU/CPU/NPU capabilities in AI PCs, it is now possible to unlock the potential to deploy these models with fast inferencing performance on the PC or laptop. These factors have created the opportunity for the AI PC, currently defined as a desktop or notebook possessing a dedicated chipset or block to run on-device AI workloads.[2]

There is tremendous optimism and projected growth for AI PCs. In Q2 of 2024, 14% of the AI PCs shipped globally were AI PCs, which doubled in growth from 7% in Q1 of 2024.[3] The adoption of AI PCs is projected to grow such that by 2027, 60% of all PCs are expected to be AI capable.[4] In the PC market in general, in 2023, Apple shipped less than 10% of PCs across the entire market[5] while Intel dominated this market with 78% of CPUs.[6]



Given the proliferation of Intel-based PCs and laptops that run on Microsoft Windows, or “Wintel,” we explored (a) whether the dominant and preferred method of deploying LLMs via the GGUF format, which is designed to optimize inferencing on Mac Metal would be equally effective on Wintel laptops; and (b) if the GGUF format does not prove to be the right software optimization technique for such Wintel laptops, then what the performance difference would be if paired with the technique best suited for them.

We tested for “real world” performance on a realistic, replicable test set and answering the basic questions that are representative of enterprise use cases. Specifically, we sought to identify what types of models and use cases are achievable in the short-term where performance will be in an acceptable range.

Key questions were:

- How do Intel Ultra laptops compare with Mac for inferencing AI models?
- How to get the fastest inference performance on each laptop?
- What does this mean for future use cases and potential adoption in the enterprise over the next 12 – 24 months?

Background: Key Trends

Since the emergence of the Llama open source models from Meta, there has been a tremendous acceleration in capabilities of small language models (SLMs), ranging from 1B – 10B parameters over the course of the last 18 months with other popular open source models being launched such as Mistral, Phi3 (Microsoft), Gemma (Google), Qwen, Yi, StableLM, and others. These models generally have the same architecture as the larger models and are trained with generally the same or comparable training set. A good rule of thumb is that these models aspire to have 80-90% of the capability for most tasks, while operating at 1/10 to 1/100 of the cost.

The power of SLMs comes from pivoting from the perspective of utilizing one large frontier model that can perform any task, to conceptualizing models as specialized for a particular task while maximizing smaller footprint, lower cost, more flexible deployment options and speed of finetuning. These attributes allow enterprises to evolve from thinking of AI as one big model to a decentralized environment of dozens or even hundreds of smaller, specialized task-based models, each performing a specific business function.

Each SLM may not have the capability of a large frontier model, but by finetuning and specializing the smaller model, the premise is that it can operate comparably – or with superiority – to the larger model at a fraction of the cost and with much better options for deploying flexibly and cost effectively.

For purposes of our testing, we looked at various model sizes and the following models as representative:

Model Name	Model Family	Model Parameters (Billion)	Quantized Binary Size (GB)
bling-tiny-llama	tiny-llama	1.1	0.67
bling-phi-3	phi-3-mini (microsoft)	3.8	2.4
dragon-llama-2	llama-2 (meta)	7.0	4.1
dragon-mistral	mistral (mistral)	7.3	4.4
dragon-yi-9b	yi (01-ai)	8.8	5.7

We used our own finetuned version of each of these model families that is optimized for enterprise fact-based question-answering in complex domains. This allowed us to provide consistency to the model outputs across the size range. For the purpose of this evaluation, we did not focus on accuracy of the models themselves as we have published other work on accuracy by model size, and where we have shown that accuracy generally increases with size, although the phi-3-mini is currently the most accurate model in our testing at only 3.8B parameters.[7]

With models in this size range, we have found that machines with at least 16 GB of RAM can generally provide effective inference, especially with smaller models, e.g., tiny-llama (1.1B), but that 32 GB is a significant improvement and allows much faster inference and even the ability to juggle two or more models concurrently in memory, enabling 'toggling' between different specialized models for different tasks concurrently that is especially important in AI agent workflow scenarios.

As a further observation, we have found that in addition to GPU/NPU/CPU processing capability, operational memory is a very important variable for the AI PC, and we expect and recommend most enterprises to move to 32 GB for any laptops in which they are looking to initiate heavy use of AI models running locally.

Memory is important for AI PCs

We recommend most enterprises to move to 32 GB for heavy AI use

Quantization and Packaging

Historically, model training and fine-tuning has been the province of libraries such as Pytorch and Tensorflow, which provide strong underlying capabilities to manage the backpropagation process (and other specialized matrix operations) used in model training. Today, virtually all of the leading LLMs are packaged primarily in Pytorch, and made available in this form for downstream finetuning, which also usually occurs in Pytorch (although there are a growing number of alternatives).

Once the model has been fully trained, and is ready for deployment, the models can be converted into other formats, such as GGUF, ONNX, TensorRT, or OpenVINO, which bring the benefit of “compiling” and fixing the model graph which results in faster performance. This usually removes the Python dependency in using the model (allowing embedding the model into C++, Java, Go, Javascript and other production applications), and then the model inferencing packaging technology is responsible for quantizing the model, usually to 4 bits, which results in a reduction in the model binary size of at least 4-8X.

As recently as just 2-3 years ago, most LLMs were kept in “floating point 32” in which 4 bytes (of 8 bits each) were used to represent a single floating point number (e.g, one of the model parameters) and would have a correlated memory consumption. For example, the simple math is that a model with 7 billion parameters in floating point 32 would take 28 GB of memory.

Increasingly, model fine-tuning and Pytorch model packaging adapted to variants of “floating point 16” in which 2 bytes are used for each parameter, such that a pytorch_model.bin file for a 7 billion parameter model is usually approximately 14 GB.

ADVANCES IN QUANTIZATION TECHNIQUES MAKE AI MODELS ACCESSIBLE ON AI PCS

Starting in 2023, a number of innovative quantization approaches emerged to evolve to 8-bit, and then to 4-bit, and even 2-bit. There are a variety of statistical techniques used, but the basic principle is to try to “shrink” the size of each parameter to an ever smaller representation without losing much in quality in the output.

The rapid progression from a 28 GB file to ~4 GB file is the journey of model quantization over the last 2 years, and a big part of the reason that high-quality 7 billion parameter models are accessible on standard laptops.

Today, the most popular quantizations are generally 4-bit (although there are a minority of 2-bit, 3-bit, and 5-bit). With 4-bit quantization, a 7 billion parameter model could be compressed to 3.5 GB. In most cases, a “smart” quantization technique is applied, which will reduce parameters to 4-bit where possible, but will keep parameters that may be more influential at 6-bit, 8-bit or 16-bit, such that often times in practice a 7 billion parameter quantized model will be ~4 GB.

Quantizations in Brief

llama.cpp/GGUF
Best for Macs and Nvidia

ONNX
Best for Windows ecosystem

OpenVINO
Best for Intel chip technologies

TensorRT
Best for Nvidia

Throughout 2023, an influential open source library, llama.cpp – and the associated GGUF format, emerged as the biggest beneficiary and catalyst of quantizing and running smaller language models on laptops and PCs. The stated mission of llama.cpp was to build an integrated inferencing engine, in C/C++, that optimized LLM performance on laptops and edge devices, especially for Mac Metal [8]. In addition to llama.cpp, there are a few other libraries to optimize inferencing:

- TensorRT has long been a model compilation standard used by Nvidia to optimize models on their GPUs, but is not used much outside of the Nvidia ecosystem.
- ONNX and ONNX Runtime have emerged as an important alternative in the Windows ecosystem in particular, offering many of the same approaches as llama.cpp.
- OpenVINO, sponsored by Intel, is optimized for Intel chip technologies – CPU, GPU and NPU.

We expect that there will continue to be a heterogeneous environment for lower level model inference packaging with adoption among each of these standards with increased optimizations for different platforms.

Performance Test

Using publicly-available and widely-used laptops, and technologies that are all open source and available for others to replicate, our goal was to construct a baseline 'real world' benchmark to evaluate the effectiveness of end-to-end inferencing.

We used the following machines for our testing that we purchased ourselves:

• Mac M1 (circa 2022) – with 32 GB of RAM – this was the specification of a high-end Mac with MSRP of $3499 in August 2022.

• Mac M3 Max (circa 2024) – with 36 GB of RAM – this is a top of the line Mac with MSRP of $3499 in March 2024.

• Dell Inspiron Intel Ultra9 (MeteorLake) with 32 GB of RAM, available from Dell directly for $1099 MSRP in June 2024.

We constructed our test based on an open source example file that we published almost a year ago, which has the benefit of being both widely-available, widely and consistently used in our previous testing, and also avoids any potential "cherry-picking" in the design to potentially bias one of the technologies being tested.

In this testing scenario, we aimed to reproduce a Retrieval Augmented Generation (RAG) scenario for fact-based question-answering. The test consisted of 21 fact-based question-answering context passages, across a wide range of business, financial and general news topics, with context passages ranging between 100 – 500 tokens, combined with a fact-based question, and then answers typically in the range of 10-100 tokens.

The last two questions generate larger answers, and are usually a good quick test of potential 'saturation' as the model runs, and do take the longest time in the process. We cap generation at 100 tokens maximum, with the expectation of using the model as a productivity tool.[9]

The full set of questions and answers can be found at this link:

https://www.github.com/llmware-ai/llmware/tree/main/examples/Models/bling_fast_start.py

Our open source library contains over 100 examples of how we use small language models as a productivity tool for enterprise use cases, an example of which can be found here[10] which shows how small language models can be used to extract text to query external web sources to complete a complex research report for financial analysis:

https://youtu.be/y4WvwHqRR60?si=FivOMgS-CbVHSQ2r

We compared Mac M1, Mac M3 and Dell Inspiron Intel Ultra 9 for Model Inference Speed

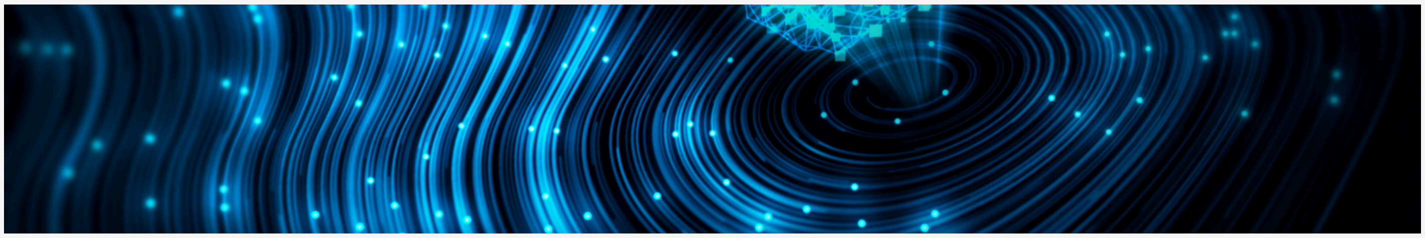
Fastest Inference on each Platform

Our main comparison is between a representative state of the art (available) Mac and a Windows laptop using Intel technologies, with the goal of packaging the model to achieve the fastest inference performance on each machine.

By leveraging the Mac Accelerate and Metal acceleration libraries, llama.cpp showed the potential for a GPU-like inference performance running locally on Mac M1 and M3 laptops, with generally acceptable performance standard of only a few seconds per inference for real-world use cases with smaller models.

We have found that 4-bit quantization GGUF (with a libllama.cpp compiled with both Metal and Accelerate) is by a far the fastest inferencing on a Mac laptop, so we only considered GGUF as our inferencing technology for the Mac in all of our tests. GGUF is widely used and accessible with large catalog of model support and for local Mac inferencing, GGUF is a hands-down winner as the fastest and easiest way to run local models, using llama.cpp.

It was less clear to us what would be the fastest inferencing approach on a Wintel laptop and in fact, it was our initial testing that disabused us of the naïve view that GGUF would be the fastest on all platforms.



Performance Result on Dell Ultra-9/Intel using Different Inference Optimization Libraries

Here is the comparison of inference performance on the Dell Ultra-9/Intel laptop, using consistently a 1.1 billion parameter tiny-llama model using our 21-question RAG test. The processing time shows the total runtime for all 21 questions:

Machine	Model	Params (B)	Quant	Framework	Processing (sec)
Ultra9-32GB	bling-tiny-llama	1.1	32	PyTorch	114.9
Ultra9-32GB	bling-tiny-llama	1.1	4_K_M	GGUF	112.9
Ultra9-32GB	bling-tiny-llama	1.1	8-INT	Optimum-Intel	93.5
Ultra9-32GB	bling-tiny-llama	1.1	int4	ONNX	65
Ultra9-32GB	bling-tiny-llama	1.1	int4	OpenVino (CPU)	54.5
Ultra9-32GB	bling-tiny-llama	1.1	int4	OpenVino (GPU)	15

We found that the OpenVINO (GPU) version of the model produces the fastest inference time - 7.6x faster - compared to the PyTorch version and 7.5x faster compared to the GGUF version.

As is shown in our testing, GGUF had negligible impact on inferencing models in our Dell Ultra-9/Intel laptop. We tried multiple ways to optimize GGUF (different compile options in llama.cpp) but reached the general conclusion pretty quickly that GGUF is primarily optimized for Mac and CUDA today, with Intel performance that is only slightly better than using a naïve Pytorch without any quantization.

We then saw significant improvement with ONNX. Satisfied with the significant performance gain, we originally planned to only use ONNX until we started to do testing with OpenVINO. We would still consider ONNX a strong viable alternative generally, especially as it has support for pluggable ‘execution providers’, including OpenVINO.

We then deployed OpenVINO with the Intel GPU Plugin, and our testing clearly showed that OpenVINO was by far superior for Intel-based machines, with massive performance improvements unlocked.

In each case, we took reasonable steps to optimize the performance for the underlying technology, but focused on ‘plain vanilla’ deployments and publicly-available builds and instructions, in the attempt to try to create a fair baseline of comparison.

Dell Ultra-9/Intel performed BEST with OpenVINO (GPU) framework

7.6x Faster than PyTorch

7.5x Faster than GGUF

Key is to match the right software to hardware

This preliminary evaluation showed us that because OpenVINO produced such superior results in our Dell Ultra9/Intel laptop, to conduct a fair evaluation of inference speeds on each of the laptops, we needed to use the quantization format best suited to each machine. Therefore, to frame the performance test as “best” on Mac versus “best” on Wintel, we decided to use these versions of the models:

- Mac – 4-bit GGUF
- Dell/Intel – ‘int4’ OpenVINO on Intel GPU (Ultra)

Results of the Performance Test

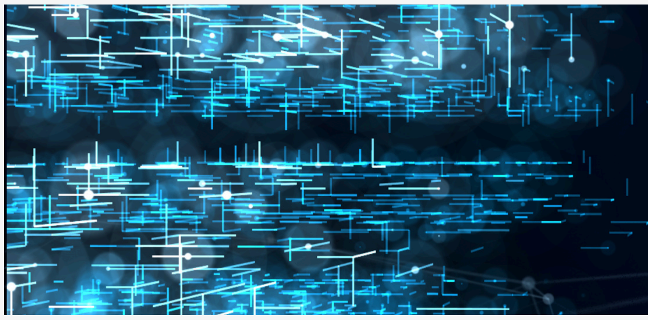
RAG "bling_fast_start" example - 21 fact-based questions with different contexts and relatively short output generations							
Model Name	Model Parameters (B)	Mac M1 (secs)	Mac M3 Max (secs)	Intel Ultra (secs)	M3 vs M1 (%)	Intel Ultra vs M1	Intel Ultra vs M3
bling-tiny-llama	1.1	31.30	23.27	15.27	25.67%	51.22%	34.38%
bling-phi-3	3.8	81.10	61.40	43.03	24.29%	46.94%	29.91%
dragon-llama2	7	128.30	97.65	75.93	23.89%	40.82%	22.24%
dragon-mistral	7.3	113.20	96.80	71.23	14.49%	37.07%	26.41%
dragon-yi-9b	8.8	172.50	143.75	89.80	16.67%	47.94%	37.53%

We ran the 21-question RAG performance test three times with each model on each platform, taking the average of those three runs. We tried to keep the machines relatively 'clean' only to ensure a fair baseline comparison, but did not take any unusual steps to set up a lab environment. As with any hardware environment, there can be saturations.

We found especially if many applications and processes were running concurrently, this could lead to saturations. As such, there is possibility for variations in the results below, but we found them to be consistent in a narrow range once the environment was in a relatively 'clean' state with minimal background applications running.

In our testing, the Dell Ultra9/Intel laptop performed up to 51% faster compared to a Mac M1 and up to 37.5% faster compared to a Mac M3. The biggest performance gains compared to both Mac M1 and M3 were in the 1B parameter and the 9B parameter LLM sizes. The Dell Ultra9/Intel laptop also significantly outperformed Mac M1 and M3 in every model size we tested, and in every single run of our testing.

In addition, given that the Dell Ultra9/Intel laptop was able to answer 21 questions in 15.27 seconds, we were able to get sub-second response times for the bling-tiny-llama model.



DELL ULTRA 9/INTEL LAPTOP SIGNIFICANTLY OUTPERFORMED MAC M1 AND M3 IN EVERY MODEL SIZE WE TESTED

Conclusion

As long-time Mac users, we were surprised at the performance we saw from the Dell/Intel Ultra GPU, with notably faster performance than both a Mac M1 (which is our standard development machine) and even a brand-new state of the art M3 Max. The superior inference speed is particularly remarkable considering the price to value ratio, given that we purchased each of the Macs for $3499 MSRP and the Dell/Intel laptop for only $1099 MSRP. It was also notable that the performance improvement was generally greater as model size increased.

As a quick rule of thumb, with 21 questions, if a test took 43 seconds to complete, it implies ~2 seconds per question. Similarly, a 15 second response time is equivalent to sub-second question answering with a locally running LLM. As performance gets into this range of <3-4 seconds per question, the range of use cases widens considerably in terms of acceptable user experience.

On the contrary, on a Mac, using 7B parameter models is often ‘border-line’ in terms of user experience. In many use cases, it is often too slow, and as a result, we typically try to use tiny-llama, phi-3 and stable-lm models. However, we believe that the Intel Ultra performs fast enough on 7b parameter models that they can be used more comfortably in a wider range of use cases – and even much larger models like the yi-9B (which provides very high quality) is also in an acceptable speed performance range.

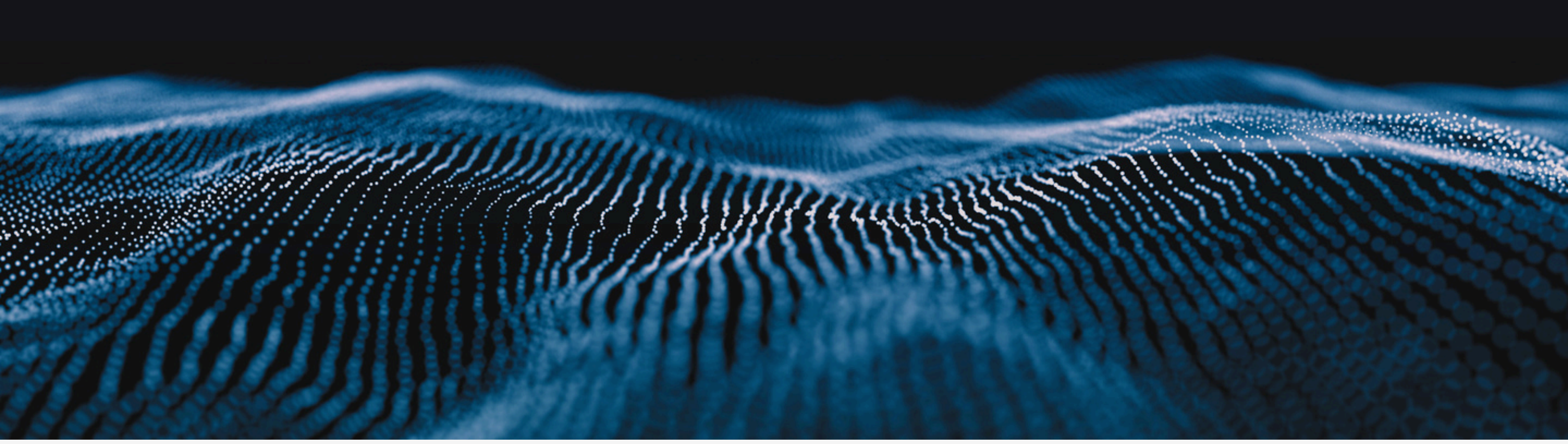
Key Take-Aways

1. Intel laptop GPU performance is surprisingly fast, once the right model packaging and software optimization technology is deployed to unlock the GPU capability (OpenVINO).

2. GGUF is not universal to all inferencing, and in fact, most enterprises will need to be able to support multiple formats, including OpenVINO and ONNX, to roll out self-hosted and locally-deployed model applications across their enterprise.

3. Price/performance – the Mac M3 Max retails at $3499, and the Dell Inspiron Ultra 9 was $1099 in June 2024. On a price-to-performance ratio, the Intel Ultra performance is especially impressive and surprising, and captures what is potentially game-changing about the roll-out of the AI PC – it can be the laptop of wide-spread distributed use of generative AI in the enterprise – cost-effectively and safely.

DELL INSPIRON ULTRA/INTEL WAS ESPECIALLY IMPRESSIVE AND SURPRISING BY BEING UP TO 51% FASTER IN PERFORMANCE WHILE LESS THAN 1/3 OF THE PRICE OF THE MACS WE TESTED



Next Steps

With the ability to run high-speed inference of models up to 9B parameters on a relatively standard issue enterprise-grade Wintel laptop, the AI PC is going to be a really exciting platform and space to watch in 2025. While we continue to be fans of GGUF, Mac and CUDA, we are shifting a lot of our efforts now to OpenVINO, ONNX and optimizing for AI PCs in the run-up to Intel's next generation Lunar Lake laptops to be launched later this year.

Explore how LLMWare.ai can help optimize your AI workflow. Contact us today.

Website:
llmware.ai

Contact:
Namee Oberst

Endnotes

[1] Laptop Specs: Dell Inspiron 14 Plus 7440, Installed RAM 32 GB, Intel Core Ultra 9 185H 2.50 Ghz Processor, Windows 11
[2] Canalys. "Now and Next for AI-Capable PCs." January 2024, p.5
[3] Canalys. "14% of PCs Shipped Globally in Q2 were AI Capable." August 13, 2024
[4] Canalys. "Now and Next for AI-Capable PCs." January 2024, p.5
[5] https://www.statista.com/topics/10435/apple-mac/#topicOverview
[6] https://www.extremetech.com/computing/intel-holds-78-global-market-share-for-cpus-analyst
[7] We have not run this accuracy testing on the yi-9B, which we expect would perform at least comparably or better.
[8] https://llama-cpp-python.readthedocs.io/en/latest/install/macos/
[9]Out of Scope – we did not consider model loading time, which can vary between different technologies and platforms, but is usually a 'one time' cost and is very environment I/O specific. We also did not consider multiple model use cases, which will be increasingly important in AI PC productivity applications, but will save that for a future evaluation.
[10] https://github.com/llmware-ai/llmware/blob/main/examples/SLIM-Agents/custom_extract_and_lookup.py