



Workstations powered by Intel can play a vital role in CPU-intensive AI developer tasks

In three AI development workflows, Intel processor-powered workstations delivered strong performance, without using their GPUs, making them a good choice for this part of the AI process



Characterizing documents

then adding them to a database and indexing them



Analyzing a portrait

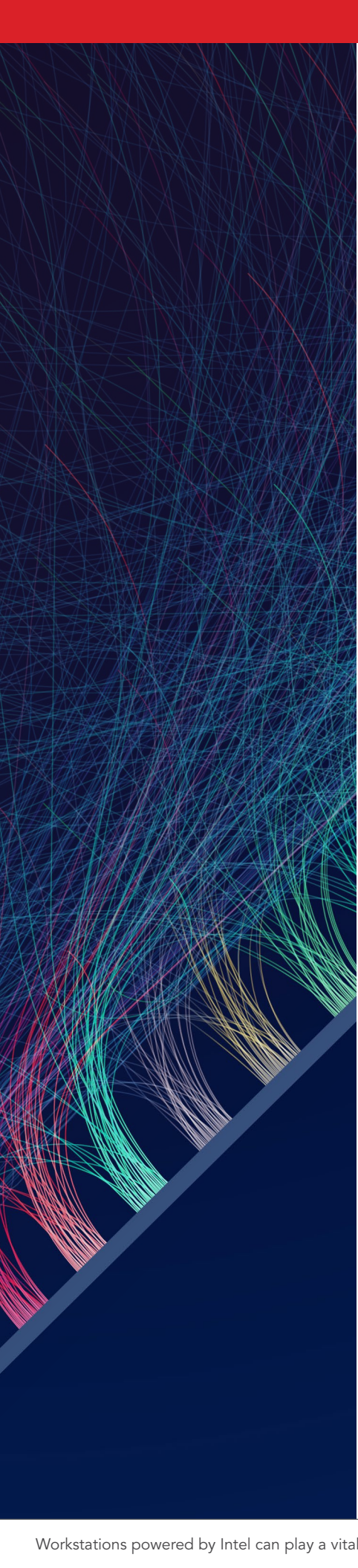
by asking a local LLM to describe it



Standardizing images

and creating new images suitable for testing or training

As the adoption of artificial intelligence (AI) has exploded in recent years, much attention has focused on graphical processing units (GPUs) and cloud-based platforms that support many AI model training and inferencing functions. However, CPU-only workstations can be a cost-effective alternative for many parts of the AI development workflow, and Intel has developed new hardware and software optimized for these kinds of tasks. In this report, we discuss the proof-of-concept testing Principled Technologies (PT) conducted, in which we executed three AI development workflows using only the CPU cores in tower and mobile workstations.



Hypothesis: Workstations with Intel processors are a strong option for certain AI tasks

While applications and services that run in the cloud or on expensive purpose-built GPUs may be useful for certain parts of an AI workflow, they can also pose obstacles in the areas of cost, privacy, and security. By shifting some common development or prototyping AI tasks to on-site workstations, organizations can mitigate these concerns.

PT conducted proof-of-concept testing to determine whether workstations with Intel processors provide an appropriate environment for carrying out data-intensive AI tasks using only the CPU cores. We tested both tower and mobile workstations.

We created three representative AI development workflows using different data sources. One workflow involved characterizing documents, adding them to a vector database running on the system, and indexing this content. A second workflow combined the disparate data sources in a multi-modal large language model (LLM) to determine the characteristics of a painting by Leonardo da Vinci. A third workflow standardized images to a common scale and precision, and used an ML k-means approach to find features in the images.

We executed the workflows on two sets of systems, all with Intel processors: three similarly configured mobile workstations from different vendors and three similarly configured tower workstations from the same vendors. We measured the time necessary—and in some cases the system memory required—to perform tasks that manipulate data. We also experimented with using the same data at different precision levels to determine how they affected performance. Our goal was not to compare the performance of the various workstations, but to demonstrate that these two categories of devices are well-suited to these tasks, regardless of the vendor you choose.

About the hardware and software environments we tested

For our proof-of-concept study, we created three custom AI development workflows and measured the time to complete them. Although all the systems we tested contained GPUs, we installed and used the CPU-only versions of the Python libraries and did not install the drivers needed to access GPU compute functions. This ensured that the workflows did not use those GPUs and all of the compute power came from the Intel CPU cores.

Test systems

We viewed the tower workstations in our testing as potentially solid choices for data scientists working with AI workflows with larger ML and AI models and more data. We tested the following tower workstations:

- **Dell™ Precision™ 7960 tower workstation** with the Intel® Xeon® w7-3455 processor, 128 GB of RAM, and a 1TB PCIe NVMe® solid-state drive (SSD)
- **HP Z8 Fury G5 tower workstation** with the Intel Xeon w7-3455 processor, 128 GB of RAM, and a 1TB PCIe NVMe SSD
- **Lenovo® ThinkStation® P7 tower workstation** with the Intel Xeon w9-3495X processor, 128 GB of RAM, and a 1TB PCIe NVMe SSD

About the Intel processors in the tower workstations we tested

All three of the tower workstations we tested feature processors from the Intel® Xeon® W-3400 processor collection. Both the Dell Precision 7960 and the HP Z8 Fury G5 feature the 24-core Intel Xeon w7-3455 processor, and the Lenovo ThinkStation P7 features the 56-core Intel® Xeon® w9-3495X processor. According to Intel, platforms featuring these processors deliver “the ultimate workstation solution for professional creators, delivering outstanding performance, security, and reliability along with expanded platform capabilities for VFX, 3D rendering, complex 3D CAD, and AI development & edge deployments.”¹

We viewed the mobile workstations as well-suited to development environments where data scientists explore and attempt to improve smaller AI models. Many data scientists and AI developers train very small data sets on their local workstations to reduce the cost of exploration. A cost-effective approach to development is to experiment on local clients first and then scale to multiple servers or the cloud. We tested the following mobile workstation systems:

- **Dell Precision 7780 mobile workstation** with the 13th Gen Intel Core™ i7-13850HX processor, 64 GB of RAM, and a 1TB PCIe NVMe SSD
- **HP ZBook Fury 16 G10 mobile workstation** with the 13th Gen Intel Core i7-13850HX processor, 32 GB of RAM, and a 512GB PCIe NVMe SSD
- **Lenovo ThinkPad® P16 G2 mobile workstation** with the 13th Gen Intel Core i9-13980HX processor, 64 GB of RAM, and a 1TB PCIe NVMe SSD

About the Intel processors in the mobile workstations we tested

As we noted earlier, the Dell Precision 7780 and the HP ZBook Fury 16 G10 we tested both feature the 13th Gen Intel Core i7-13850HX processor. This processor has 20 cores, 5.30 GHz maximum turbo frequency, and 30 MB Intel Smart Cache. The Lenovo ThinkPad P16 G2 we tested features the 13th Gen Intel Core i9-13980HX processor with 24 cores, 5.60 GHz maximum turbo frequency, and 36 MB Intel Smart Cache.

Workflow environment and analytic tools

We used the Ubuntu 22.04 Linux operating system. For our scripting language, we used Python, which is well suited for data ingestion, manipulation, and exploration. Higher-level functions that support AI are available as add-on packages to Python. Many of these are efficient because they use optimized libraries to perform their calculations in CPU-only or with the assistance of GPUs. We used the CPU-only versions.

We performed all of the tasks using Python scripts. Another option we could have used is Python in notebooks, a browser-based visual interface to sequences of Python commands. Notebooks allow a data scientist to perform tasks repeatedly as well as explore many “what-if” ideas. Users can save notebooks and reuse the code in them. Visualizing data is an important part of data science. We used Python plotting packages to display results inside a notebook.

We also used some AI functions from Python that a data scientist might use to find better ways to represent data for AI (e.g., to test the efficiency of an embedding model, or determine which data fields to use to index documents). We used local AI models, though a data scientist might also wish to use server cluster- or cloud-based ones via standard API calls. Again, we used Python to effect these steps.

Finally, we used the Intel Python distribution and some of the optimized versions of Python packages it provides for ML and AI. These are part of Intel AI Tools (see box below).

About Intel AI Tools

AI Tools from Intel, formerly known as the Intel® AI Analytics Toolkit, aim to maximize performance at all stages of the AI pipeline, from preprocessing through machine learning, and support efficient model development through interoperability.² According to Intel, AI Tools “give data scientists, AI developers, and researchers familiar Python* tools and frameworks to accelerate end-to-end data science and analytics pipelines on Intel® architecture.”

AI Tools allow users to “train on Intel® CPUs and GPUs and integrate fast inference into your AI development

workflow with Intel®-optimized, deep learning frameworks for TensorFlow* and PyTorch*, pretrained models, and model optimization tools; achieve drop-in acceleration for data preprocessing and machine learning workflows with compute-intensive Python packages, Modin*, scikit-learn*, and XGBoost; and gain direct access to analytics and AI optimizations from Intel to ensure that your software works together seamlessly.”³

Learn more at <https://www.intel.com/content/www/us/en/developer/tools/oneapi/ai-analytics-toolkit.html>.

Our findings

Workflow 1: Characterizing documents, adding them to a vector database running on the system, and indexing them

For this workflow, we began with a single clean data source of unstructured text documents. We used a local embedding model to automatically characterize the documents, add them to a local vector Redis database, and index them. This is typically part of the process for setting up an AI-assisted chatbot. Once the corpus of documents is in a vector database, chatbots can efficiently search them. If you don't care about the general information that might be in a large language model (LLM) AI such as GPT, you can get the answer from this corpus of data and use the LLM's ability to recognize language to read the query and come up with a good answer.

Our workflow's scripts analyzed the document and categorized the various parts of it. It pulled out tables as well as the summary and the text, turned it into a format that is easy to search, and put into a Redis database that has vector database capabilities.

Figure 1 shows the average time the three tower workstations needed to complete the tasks in Workflow 1. Table 1 breaks down the time across the five phases of the workflow. As they show, loading and database uploading were the most time-intensive phases of the workflow.

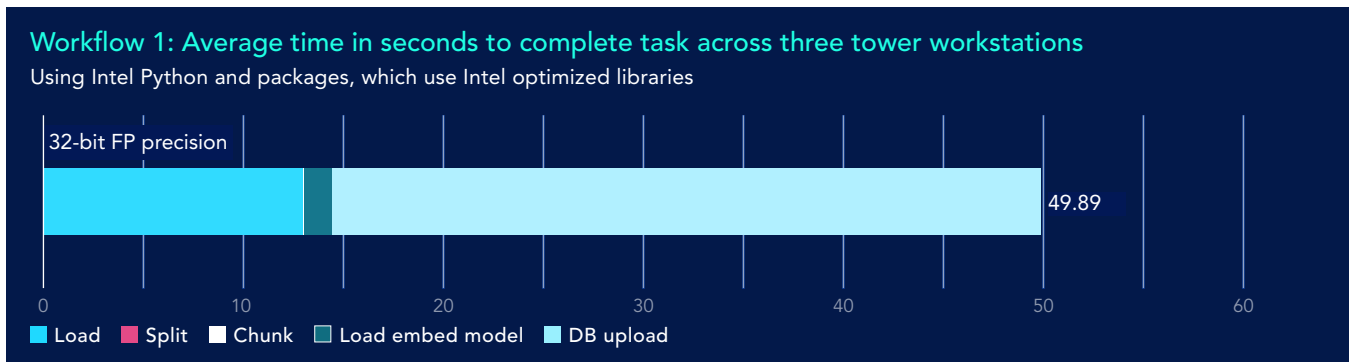


Figure 1: The average time to execute Workflow 1 across the three tower workstations using Intel AI tools. Note that the Split and Chunk phases took so little time that they do not appear in the chart. Source: Principled Technologies.

Table 1: The amount of time, in seconds, each phase of Workflow 1 took using Intel Python and libraries (average of three tower workstations). Asterisk indicates less than 10 milliseconds. Source: Principled Technologies.

Task	Load	Split	Chunk	Load embed model	DB upload	Total
Time to complete (seconds)	12.99	*	0.02	1.42	35.44	49.89

Completing Workflow 1 on the mobile workstations took roughly twice as long as on the tower workstations (see Figure 2 and Table 2). Again, loading and database uploading were the most time-intensive phases of the workflow. We consider these times acceptable given the tasks, indicating that these workstations are appropriate choices for this workflow.

For simplicity, we have truncated the numbers in the tables and charts. Consequently, numbers may not sum to the total. Untruncated numbers appear in [the science behind the report](#). For Workflow 2, we report the maximum time it took to complete any phase as well as the maximum time to complete all phases. The times we report come from different images, which vary widely in complexity. As a result, in two cases, the sum of the maximum times for the tasks exceeded the maximum time to process an image.

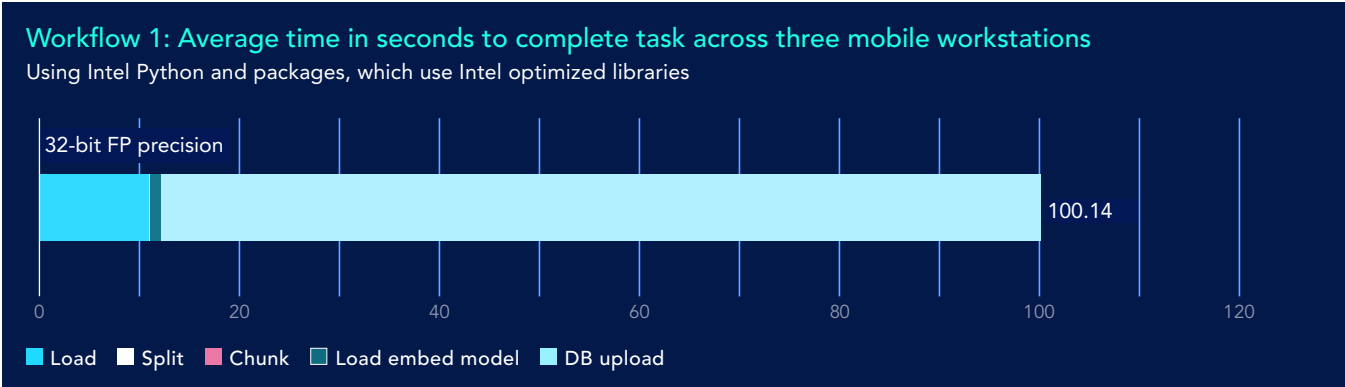


Figure 2: The average time to execute Workflow 1 across the three mobile workstations using Intel AI tools. Note that the Split and Chunk phases took so little time that they do not appear in the chart. Source: Principled Technologies.

Table 2: The amount of time, in seconds, each phase of Workflow 1 took using Intel Python and libraries (average of three mobile workstations). Asterisk indicates less than 10 milliseconds. Source: Principled Technologies.

Task	Load	Split	Chunk	Load embed model	DB upload	Total
Time to complete (seconds)	11.03	*	0.02	1.18	87.90	100.14

Workflow 2: Combining disparate data sources

In this workflow, we used the disparate data sources in a multi-modal large language model (LLM) to determine the characteristics of a painting by Leonardo.

We used a multiprocessing model where we sent images from the main process to the other cores, which performed the image segmentation and returned the results to the main process. This allowed us to keep the cores busy; if an image required more time because it had complicated features, it tied up only one core. The other cores processed their images independently.

Table 3 shows the average time the three tower workstations needed to complete Workflow 2. Processing made up the bulk of the time, with very brief pre- and post-processing phases.

Table 4 shows the average time the three mobile workstations needed to complete Workflow 2. As we saw with Workflow 1, the mobile workstations as a group took roughly twice as long as the tower workstations did to execute the tasks. Again, we believe these times are acceptable for these tasks, and these workstations are a solid option for this workflow.

Table 3: The amount of time, in seconds, each phase of Workflow 2 took using Intel Python and libraries (average of three tower workstations). Source: Principled Technologies.

Task	Pre-processing	Processing	Post-processing	Total
Time to complete (seconds)	0.62	136.55	0.30	137.48

Table 4: The amount of time, in seconds, each phase of Workflow 2 took using Intel Python and libraries (average of three mobile workstations). Source: Principled Technologies.

Task	Pre-processing	Processing	Post-processing	Total
Time to complete (seconds)	1.30	265.71	0.51	267.53

Workflow 3: Standardizing images to a common scale and precision

This workflow involved taking image data and processing it into a form suitable for either training a neural net to find features or using it against a trained neural net to categorize its content. We used an open-source medical data image set consisting of images from lung CAT scans. Our workflow involved the tasks prior to the compute-intensive AI analysis, which would be more appropriate to execute using clusters of systems with GPUs. Our workflow prepared the data

for training or inference or segmentation. The systems identified images that were of low quality, e.g., had missing data. We standardized the images to a standard scale and precision (dynamic range). Next, we created new images from the existing ones that would be useful for testing or training on image data, e.g., rotating, cropping, flipping, rescaling, and changing the color of parts. Finally, we compressed and recoded the images in preparation for further AI analysis.

Tower workstations

The processors in the tower workstations support both 16-bit floating point (FP) precision and 32-bit FP precision. As Figure 3 shows, using 16-bit precision, the workstations completed the workflow in markedly less time, a savings of 26.7 percent.

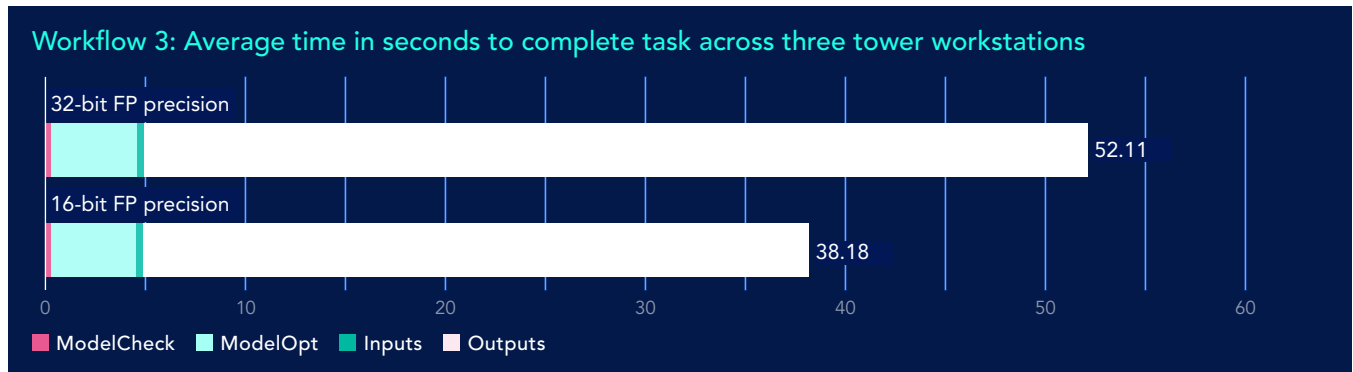
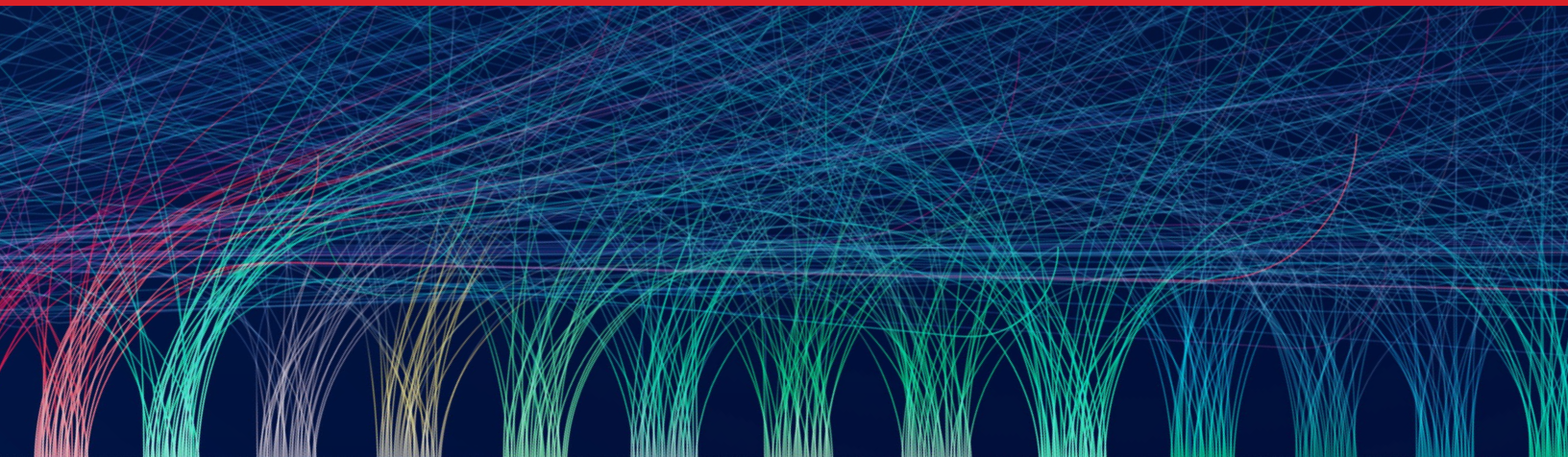


Figure 3: A comparison of the average time to execute Workflow 3 across the three tower workstations, using Intel AI tools, at 16-bit and 32-bit FP precision. Source: Principled Technologies.

Table 5 breaks down the time across the four phases of the workflow. As it shows, Outputs was the most time-intensive phase of the workflow.

Table 5: The amount of time, in seconds, each phase of Workflow 3 took using Intel Python and libraries (average of three tower workstations). Source: Principled Technologies.

Task	ModelCheck	ModelOpt	Inputs	Outputs	Total
Time to complete using 32-bit FP precision (seconds)	0.27	4.33	0.35	47.15	52.11
Time to complete using 16-bit FP precision (seconds)	0.30	4.25	0.35	33.27	38.18



In addition to measuring the time to complete this workflow, we also monitored memory usage during the tasks. As Figure 4 shows, using 16-bit precision not only allowed the workstations to execute the workflow in less time, but also reduced memory usage dramatically.

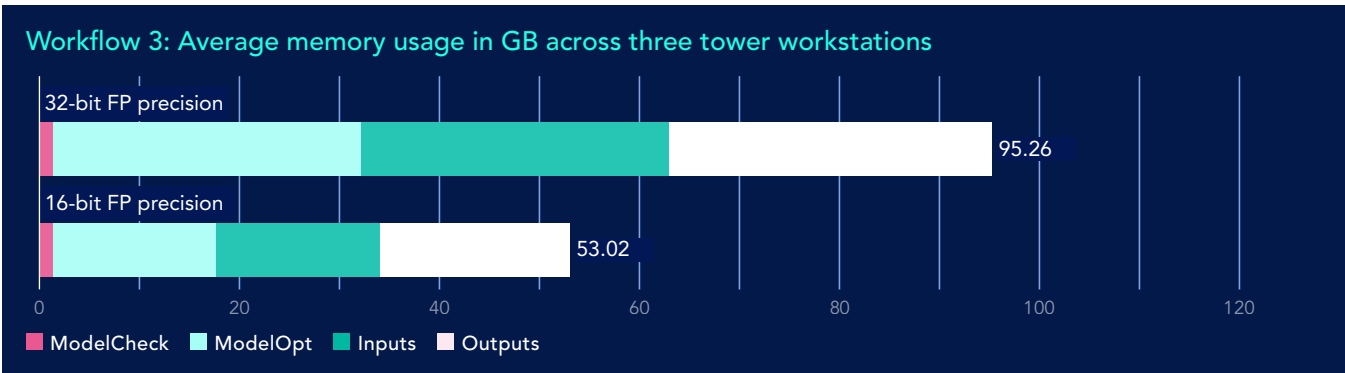


Figure 4: A comparison of the average memory usage in GB during Workflow 3 across the three tower workstations, using Intel AI tools, at 32-bit and 16-bit FP precision. Source: Principled Technologies.

Table 6 breaks down the memory usage across the four phases of the workflow. As it shows, ModelOpt, Inputs, and Outputs were the most memory-intensive phases of the workflow.

Table 6: The amount of memory each phase of Workflow 3 consumed using Intel Python and libraries (average of three tower workstations). Source: Principled Technologies.

Task	ModelCheck	ModelOpt	Inputs	Outputs	Total
Average memory usage when using 32-bit FP precision (GB)	1.38	30.74	30.85	32.28	95.26
Average memory usage when using 16-bit FP precision (GB)	1.38	16.29	16.42	18.92	53.02

Mobile workstations

The mobile workstations we tested support only 32-bit FP precision. Note: We were unable to complete this workflow on one of the three mobile workstations we tested because the workflow required just over 32 GB of RAM to run efficiently and this workstation had only 32 GB of RAM. Consequently, these results are the average of the other two workstations. As Figure 5 shows, using 32-bit precision, the two mobile workstations we tested completed the workflow in an average of 223.80 seconds, more than four times as long as the tower workstations needed.

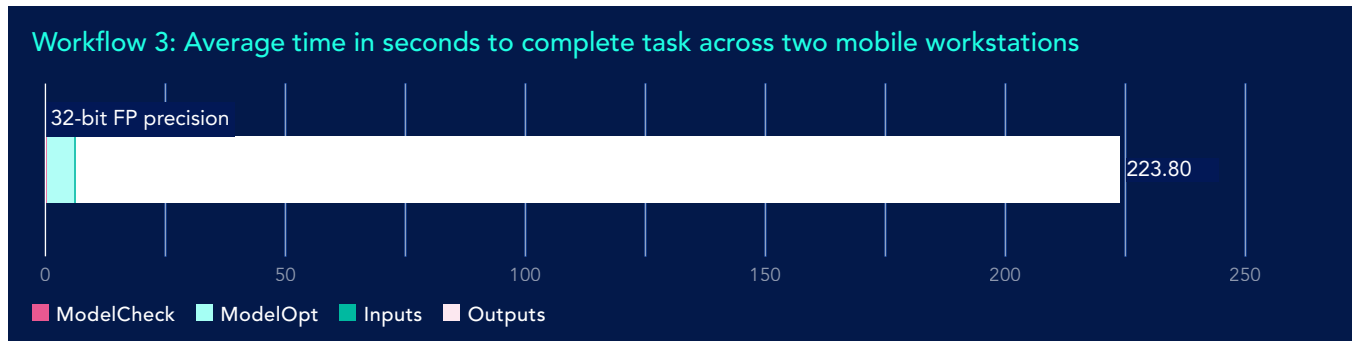


Figure 5: The average time to execute Workflow 3 across the two mobile workstations, using Intel AI tools, at 32-bit FP precision. Source: Principled Technologies.

Table 7 breaks down the time across the four phases of the workflow. As it shows, Outputs was the most time-intensive phase of the workflow. As was the case with Workflows 1 and 2, we judge these times to be acceptable for completing these tasks, and believe these workstations are good choices for this workflow.

Table 7: The amount of time, in seconds, each phase of Workflow 3 took using Intel Python and libraries (average of two mobile workstations). Source: Principled Technologies.

Task	ModelCheck	ModelOpt	Inputs	Outputs	Total
Time to complete (seconds)	0.35	5.70	0.37	217.36	223.80



Conclusion

We executed three AI development workflows on tower workstations and mobile workstations from three vendors, with each workflow utilizing only the Intel CPU cores, and found that these platforms were suitable for carrying out various AI tasks. For two of the workflows, we learned that completing the tasks on the tower workstations took roughly half as much time as on the mobile workstations. This supports the idea that the tower workstations would be appropriate for a development environment for more complex models with a greater volume of data and that the mobile workstations would be well-suited for data scientists fine-tuning simpler models. In the third workflow, we explored tower workstation performance with different precision levels and learned that using 16-bit floating point precision allowed the workstations to execute the workflow in less time and also reduced memory usage dramatically. For all three AI workflows we executed, we consider the time the workstations needed to complete the tasks to be acceptable, and believe that these workstations can be appropriate, cost-effective choices for these kinds of activities.

-
1. Intel, "Intel® Xeon® W Processors," accessed April 10, 2024, <https://www.intel.com/content/www/us/en/products/details/processors/xeon/w.html>.
 2. Intel, "AI Tools," accessed April 8, 2024, <https://www.intel.com/content/www/us/en/developer/tools/oneapi/ai-analytics-toolkit.html>.
 3. Intel, "AI Tools."

Read the science behind this report at <https://facts.pt/W6LrIMS> ►



Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc. All other product names are the trademarks of their respective owners. For additional information, review the science behind this report.

This project was commissioned by Intel.