Introduction to AIXPRT

AIXPRT is a benchmark tool that makes it easier to evaluate a system's machine learning inference performance by running common image-classification, object-detection, and recommender system workloads.

March 26, 2020

BenchmarkXPRT

BenchmarkXPRT Development Community

Table of contents

Introduction

This paper provides an overview of how to access, configure, and use AIXPRT for testing. AIXPRT is an AI benchmark tool that lets you evaluate a system's machine learning (ML) inference performance by running common image-classification, object-detection, and recommender system workloads.

In this document, you will find details on AIXPRT toolkits and workloads, system requirements, choosing and downloading an installation package, adjusting test parameters, using alternative test configuration files, and understanding and submitting AIXPRT results.

The toolkits and workloads

AIXPRT includes support for the Intel OpenVINO, TensorFlow, and NVIDIA TensorRT toolkits to run image-classification and object-detection workloads with the ResNet-50 and SSD-MobileNet v1 networks, as well as a Wide and Deep recommender system workload with the Apache MXNet toolkit. The OpenVINO, TensorFlow, and TensorRT packages are available for Windows and Ubuntu, while the MXNet package is available for only Ubuntu. The test reports FP32, FP16, and INT8 levels of precision.

PACKAGE SELECTOR TOOL

AI workloads are now relevant to all types of hardware, from servers to laptops to IOT devices, so we intentionally designed AIXPRT to support a wide range of potential hardware, toolkit, and workload configurations. This approach provides AIXPRT testers with a tool that is flexible enough to adapt to a variety of environments. The downside is that the number of options makes it fairly complicated to determine which AIXPRT download package suits your needs.

To help testers navigate this complexity, we've developed an interactive package selector tool. Testers select options in five categories: operating system, host hardware, toolkit, target hardware, and workload. They can proceed in any order but must make a selection for every category. Because not all combinations work together, each selection the tester makes eliminates some options in the remaining categories. Figure 1 shows the operating system category.

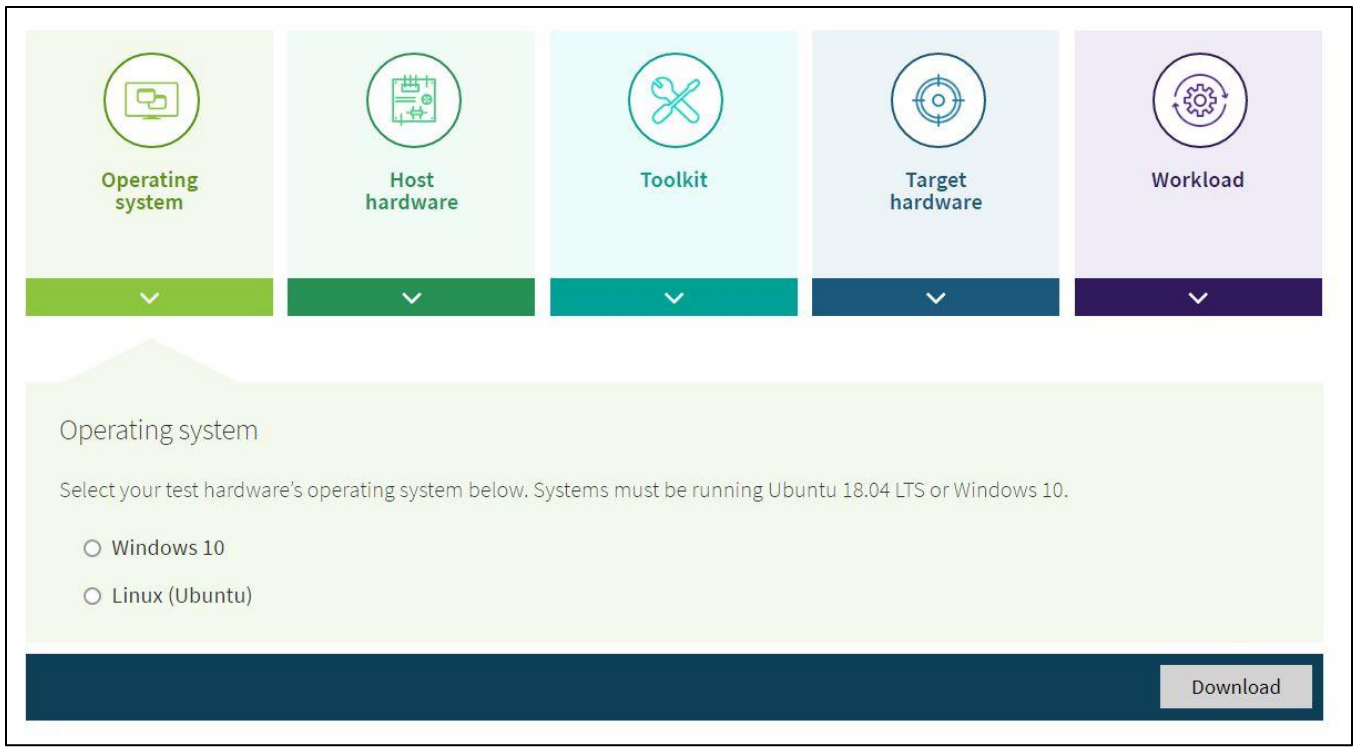


Figure 1: Designating the operating system with the AIXPRT package selector tool.

After a tester selects an option, a check mark appears on the category icon, and the selection they have made appears in the category box (e.g., TensorFlow in the Toolkit category). This shows testers which categories they've completed and the selections they've made. After a tester completes more than one category, a Start over button appears in the lower-left corner. Clicking this button clears all selections and gives testers a clean slate. Figure 2 shows the toolkit category, with the operating system and host hardware categories having already been completed.

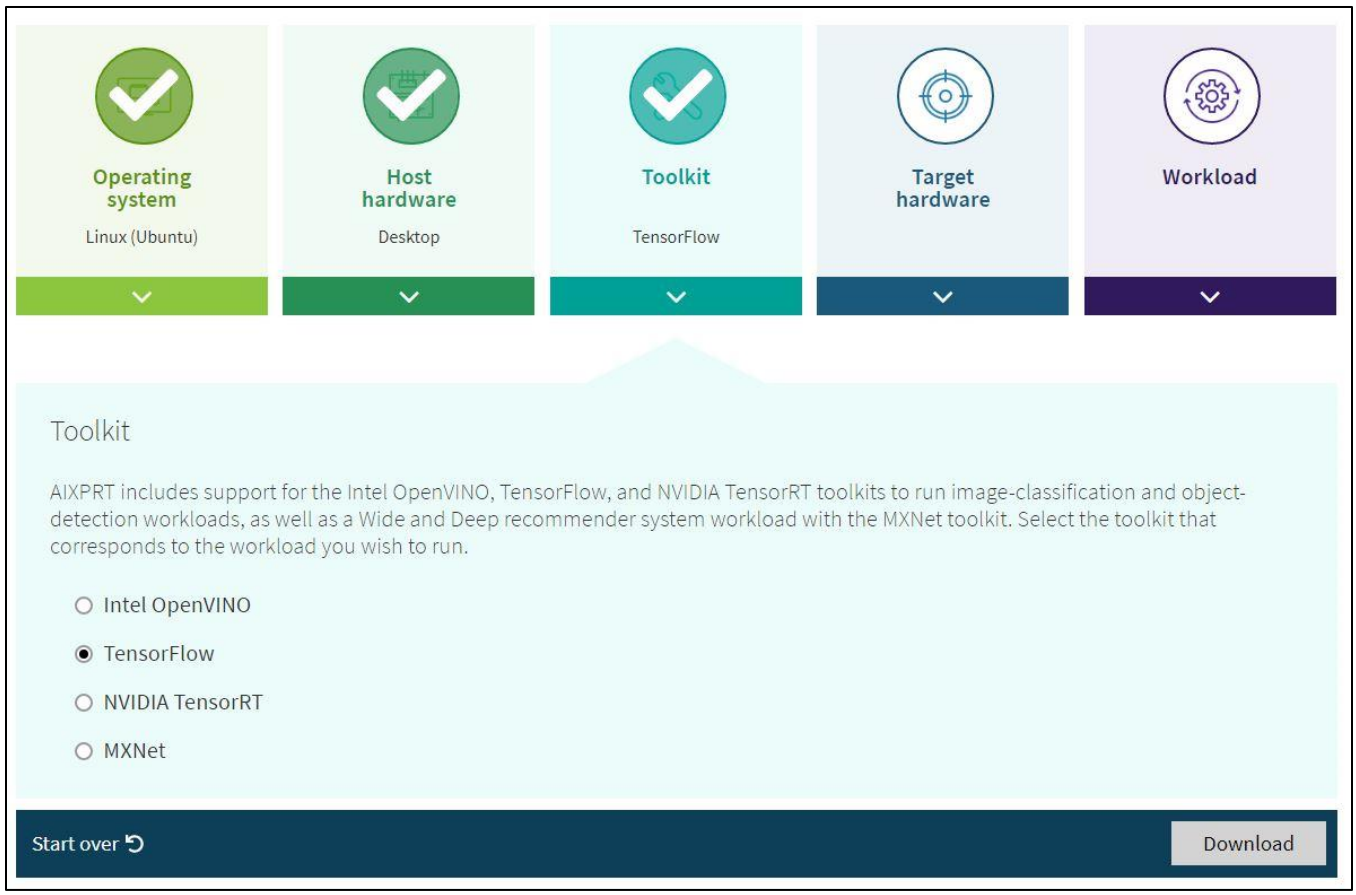


Figure 2: Choosing a toolkit with the AIXPRT package selector tool.

Once you've completed all five categories, a Download button appears in the lower-right corner. When you click this, a popup appears that provides a link for the correct download package and associated readme file.

PACKAGE DOWNLOAD TABLE

Testers who know exactly which package they need can bypass the tool and go directly to the download table.

Operating system	Toolkit	Target hardware	Install package	Documentation
Windows 10	OpenVINO (bundled)	CPUs, Intel processor graphics	Download	Readme
Windows 10	TensorFlow	CPUs, NVIDIA GPUs	Download	Readme
Linux (Ubuntu)	OpenVINO	CPUs, Intel processor graphics	Download	Readme
Linux (Ubuntu)	TensorFlow	CPUs, AMD GPUs, NVIDIA GPUs	Download	Readme
Linux (Ubuntu)	TensorRT in TensorFlow	NVIDIA GPUs	Download	Readme

System requirements and installation

AIXPRT test systems must be running either Ubuntu 18.04 LTS or Windows 10. The minimum CPU and GPU requirements vary by toolkit. Testers can find the package-specific system requirements and installation instructions for each framework in the readme files included in the AIXPRT install package. The readme files for each respective framework in the download packages are located here:

- AIXPRT_1.0.1_OpenVINO_Windows\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_OpenVINO_Ubuntu\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_Tensorflow_Windows\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_Tensorflow_Ubuntu\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_TensorRT_Windows\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_TensorRT_Ubuntu\AIXPRT\Modules\Deep-Learning
- AIXPRT_1.0_MXNet_Ubuntu\AIXPRT\Modules\Deep-Learning

You can also access the readme files in the AIXPRT Resources GitHub repository. Please be sure to read the "Known Issues" section in the readme, as there may be issues relevant to your specific configuration.

NOTE: The OpenVINO on Windows package includes a precompiled version of OpenVINO for easy installation on Windows via a few quick commands, and a script that installs the necessary OpenVINO dependencies.

Test parameters

AIXPRT allows testers to adjust four key test parameters: batch size, level of precision, number of concurrent instances, and default number of requests. Below, we discuss each parameter and describe how to adjust it with pre-test configuration steps.

BATCH SIZE

Note: The term "batch size" means one thing in ML inference and something different in ML training. AIXPRT tests inference, so we'll discuss on that meaning of the term.

In ML inference, batch size refers to the number of combined input samples (e.g., images) that the tester wants the algorithm to process simultaneously. When testing inference performance, testers adjust batch size to achieve an optimal balance between latency (speed) and throughput (the total amount processed over time).

Because of the lighter demands of processing one image at a time, Batch 1 often produces the fastest latency times, and can be a good indicator of how a system handles near-real-time inference demands from client devices. Larger batch sizes (8, 16, 32, 64, or 128) can result in greater throughput on test hardware that is capable of completing more inference work in parallel. However, this increased throughput can come at the expense of latency. Running concurrent inferences via larger batch sizes is a good way to gauge the maximum throughput a server can handle.

Configuring inference batch size in AIXPRT

A good practice when starting to explore batch size is to match it to the number of cores under test (e.g., Batch 8 for eight cores). To adjust batch size in AIXPRT, testers edit the configuration files located in AIXPRT/Config. To represent a spectrum of common tunings, AIXPRT tests Batches 1, 2, 4, 8, 16, and 32 by default.

The screenshot below shows part of a sample config file. The numbers in the lines immediately below "batch_sizes" indicate the batch size. This test configuration would run tests using both Batch 1 and Batch 2. To change batch size, simply replace those numbers and save the changes.

LEVELS OF PRECISION

Another key test variable is the level of precision. In the context of ML inference, this refers to the computer number format (FP32, FP16, or INT8) representing the weights (parameters) a network model uses when performing the calculations necessary for inference tasks.

Higher levels of precision help decrease the number of false positives and false negatives, but can increase the amount of time, memory bandwidth, and computational power necessary to achieve accurate results. Lower levels of precision typically (but not always) enable the model to process inputs more quickly while using less memory and processing power. However, they can allow a degree of inaccuracy that is unacceptable for certain real-world applications.

For example, a high precision level might be appropriate for computer vision applications in the medical field, where the benefits of hyper-accurate object detection and classification far outweigh the benefit of saving a few milliseconds. On the other hand, a low precision level could work well for vision-based sensors in the security industry, where alert time is critical and monitors simply need to know that an animal or a human triggered a motion-activated camera.

FP32, FP16, and INT8

In AIXPRT, we can instruct the network models to use one of three levels of precision: FP32, FP16, or INT8.

- FP32 refers to single-precision (32-bit) floating point format, a number format that can represent an enormous range of values with a high degree of mathematical precision. Most CPUs and GPUs handle 32-bit floating point operations very efficiently, and many programs that use neural networks, including AIXPRT, use FP32 precision by default.
- FP16 refers to half-precision (16-bit) floating point format, a number format that uses half the number of bits as FP32 to represent a model's parameters. A lower level of precision than FP32, FP16 still provides a great enough numerical range to successfully perform many inference tasks. FP16 uses less memory than FP32 and is often faster.
- INT8 refers to the 8-bit integer data type. INT8 data, which has a smaller numeric range than floating point data, is a good choice for certain types of calculations. Depending on the model, using INT8 precision can significantly improve latency and throughput over floating point precision, but accuracy can decrease. This is not always the case, however. Researchers have shown that a process called quantization, which involves approximating continuous values with discrete counterparts, can enable some networks, such as ResNet-50, to run INT8 precision without any significant loss of accuracy.

Configuring the level of precision

The screenshot below shows part of the same sample file we used in the batch size section. The value in the "precision" row indicates the precision setting. This test configuration would run tests using INT8. To change the precision, a tester simply replaces that value with "fp32" or "fp16" and saves the changes.

```
{
    "delayBetweenWorkloads": 5,
    "isDemo": false,
    "iteration": 3,
    "module": "Deep-Learning",
    "runtype": "performance",
    "workloads_config": [
        {
            "batch_sizes": [
                1
            ],
            "concurrent_instances": 1,
            "hardware": "cpu",
            "name": "ResNet-50",
            "precision": "int8",
            "runtype": "performance",
            "total_requests": 200
        },
{
            "batch_sizes": [
                2
            ],
            "concurrent_instances": 1,
            "hardware": "cpu",
            "name": "ResNet-50",
            "precision": "int8",
            "runtype": "performance",
            "total_requests": 200
```


Note that while decreasing precision from FP32 to FP16 or INT8 often improves throughput and inference speed overall, this is not always true. Many other factors can affect ML performance, including (but not limited to) the complexity of the model, the presence of specific ML optimizations for the hardware under test, and any inherent limitations of the target CPU or GPU.

CONCURRENT INSTANCES

In the context of ML inference, the number of concurrent instances refers to how many instances of the network model (ResNet-50, SSD-MobileNet, etc.) the benchmark runs simultaneously.

By default, AIXPRT toolkits run one instance at a time and distribute the compute load according to the characteristics of the CPU or GPU under test, as well as any relevant optimizations or accelerators in the toolkit's reference library. By increasing the number of concurrent instances, a tester can use multiple CPUs or GPUs to run multiple instances of a model simultaneously, usually to increase throughput.

With multiple concurrent instances, a tester can leverage additional compute resources to achieve potentially higher throughput while maintaining latency targets.

In the current version of AIXPRT, all available toolkits let testers run multiple concurrent instances. OpenVINO and TensorRT automatically allocate hardware for each instance and don't let testers make manual adjustments. TensorFlow and MXNet require testers to manually bind instances to specific hardware.

Setting the number of concurrent instances

In our sample config file, the value in the "concurrent instances" row indicates how many concurrent instances will be operating during the test. In this example, the number is one. To change that value, a tester simply replaces it with the desired number and saves the changes.

DEFAULT NUMBER OF REQUESTS

The final key test configuration variable in AIXPRT is the default number of requests, which appears in the same config file that contains the batch size, level of precision, and number of concurrent instances settings. The default setting for this variable differs depending on the AIXPRT test package you choose.

The total_requests variable specifies how many inference requests AIXPRT will send to a network (e.g., ResNet-50) during one test iteration at a given batch size (e.g., Batch 1, 2, 4, etc.). This simulates the inference demand that the end users place on the system. Because we designed AIXPRT to run on different types of hardware, we set the default number of requests for each test package to suit the most likely hardware environment on which that package will run.

For example, testing with OpenVINO on Windows aligns more closely with a consumer device (desktop or laptop) scenario than testing with OpenVINO on Ubuntu, which is more typical of server/datacenter testing. Those testing consumer devices require a much lower inference demand than those testing servers, and the default total_requests settings for the two packages reflect that. The default for the OpenVINO on Windows package is 500, while the default for the OpenVINO on Ubuntu package is 5,000.

Setting the number of requests so low that a system finishes each workload in less than 1 second can produce high run-to-run variation, so our default settings represent a lower boundary that will work well for common test scenarios.

Finding the optimal combination of machine learning variables for each scenario is often a matter of trial and error, and the following default settings represent what we think is a reasonable starting point for each test package:

- MXNet: 1,000
- OpenVINO on Ubuntu: 5,000
- OpenVINO on Windows: 500
- TensorFlow on Ubuntu: 100
- TensorFlow on Windows: 10
- TensorRT on Ubuntu: 5,000
- TensorRT on Windows: 500

Testers can adjust these variables in the config file to meet their needs. To do so, first locate and open the JSON test configuration file in the AIXPRT/Config directory. Below, we show a section of the default config file (CPU_INT8.json) for the OpenVINO on Windows test package (AIXPRT_1.0.1_OpenVINO_Windows.zip). For each batch size, the total_requests setting appears at the bottom of the list of configurable variables. In this case, the default setting Is 500. Change the total_requests numerical value for each batch size in the config file, save your changes, and close the file.

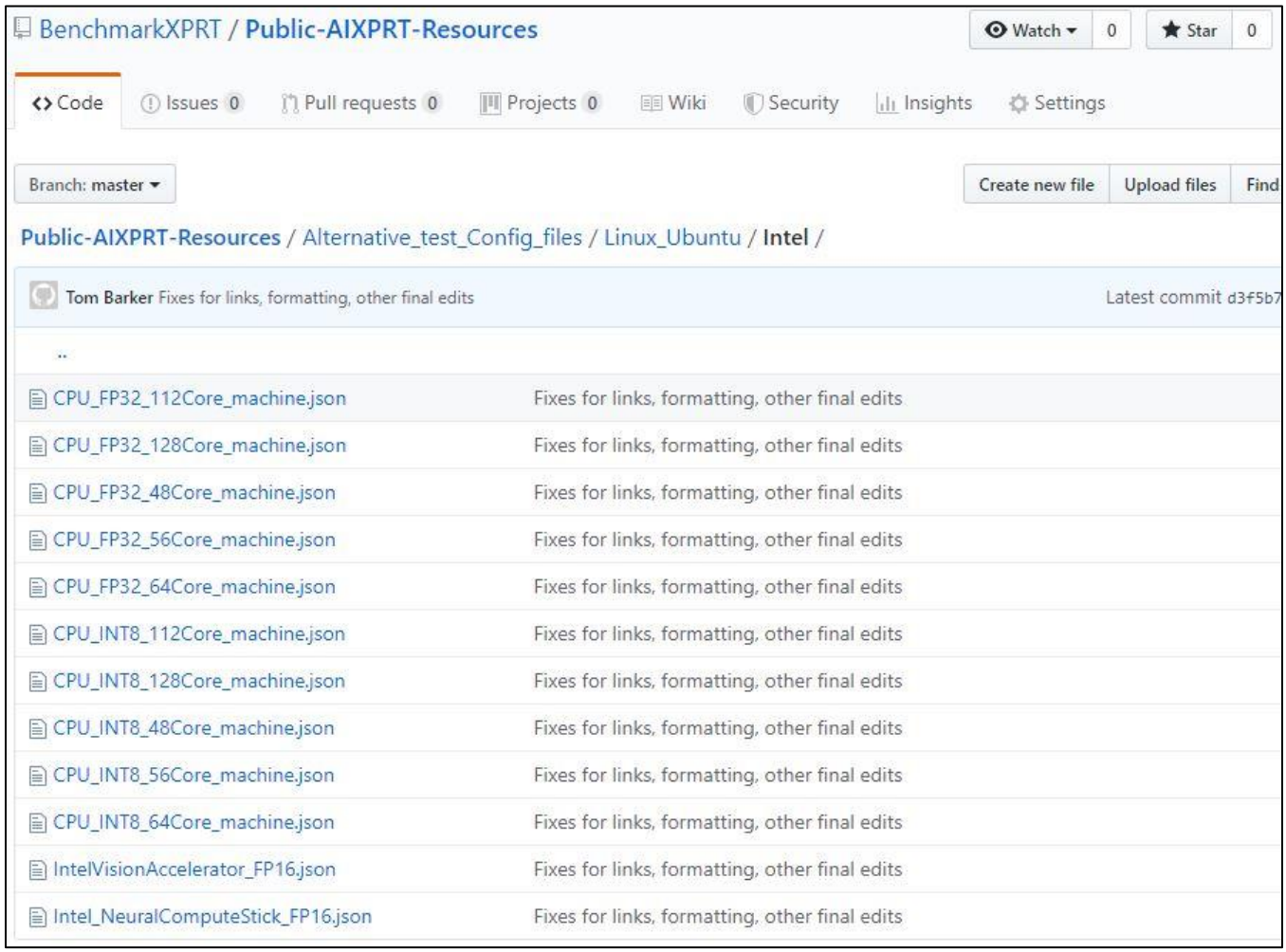
Note that if you are running multiple concurrent instances, OpenVINO and TensorRT automatically distribute the number of requests among the instances. MXNet and TensorFlow testers must manually allocate the instances in the config file. You can find an example of how to structure manual allocation here. We hope to make this process automatic for all toolkits in a future update.

Using alternate test configuration files

While AIXPRT testers can adjust key variables by editing the JSON file in the AIXPRT/Config directory, editing the variables manually can take some time, and testers don't always know the appropriate values for their system. To address both issues, we provide a selection of alternative config files in the AIXPRT Resources GitHub repository. Testers can download these files and drop them into the AIXPRT/Config directory to quickly and easily change the parameters of a test.

In the GitHub repository, we've organized the available config files first by operating system (Linux_Ubuntu and Windows) and then by vendor (All, Intel, and NVIDIA). Within each section, we offer preconfigured JSON files set up for several scenarios, such as running with multiple concurrent instances on a system's CPU or

GPU, running with FP32 precision instead of FP16, etc. The screenshot below shows the preconfigured files that are currently available for systems running Ubuntu on Intel hardware.



Because potential AIXPRT use cases cut across a wide range of hardware segments, including desktops, edge devices, and servers, not all AIXPRT workloads and configs will be applicable to each segment. In many cases, the ideal combination of test configuration variables remains an open question for ongoing research. However, we hope the alternative configuration files will help by giving testers a starting place.

Please note that each alternative test configuration file you want to run should replace the existing default config file. If multiple config files are present, AIXPRT will run all of the configurations and generate a separate result for each. More information about the config files, including detailed instructions for handling the files, is available in the EditConfig.md document in the GitHub repository.

After running the benchmark

UNDERSTANDING AIXPRT RESULTS

To understand AIXPRT results at a high level, it's important to revisit the core purpose of the benchmark. The AIXPRT bundled toolkits measure inference latency (the speed of processing) and throughput (the number of inputs a system processes in a given time period) for image recognition (ResNet-50), object detection (SSD-MobileNet v1), and recommender system (Wide and Deep) tasks. Testers have the option of adjusting variables such as batch size (the number of input samples to process simultaneously) to try and increase throughput, but greater throughput can come at the expense of increased per-task latency. In real-time or near real-time use cases such as performing image recognition on individual photos a camera is capturing, lowering latency improves the user experience. In other cases, such as performing image recognition on a large library of photos, increasing throughput might be more beneficial; designating larger batch sizes or running concurrent instances can allow the overall workload to complete more quickly.

The dynamics of these performance tradeoffs ensure that no single score could represent good performance on all machine learning scenarios. Some testers might prioritize lower latency, while others would sacrifice latency to achieve the greater throughput that their use cases demand.

For each completed run, testers can find two files in the AIXPRT/Results folder: (1) a JSON results file with latency and throughput numbers and (2) a CSV raw results file that includes values for each AI task configuration (e.g., ResNet-50, Batch1, on CPU). Parsing and consolidating the raw data can take some time, and we're developing a results file parsing tool to make the job much easier.

Currently, the results parsing tool is available in only the AIXPRT OpenVINO on Windows package, though we hope to add it to more packages soon. The tool produces a summary (example below) that makes it easier to quickly identify relevant comparison points such as maximum throughput and minimum latency.

RESULT SUMMARY:		
ResNet-50 Maximum Inference Throughput	63.79	images/sec
ResNet-50 Minimum Inference Latency per image (90th percentile)	26.441401	milliseconds
SSD-MobileNet-v1 Maximum Inference Throughput	154.665	images/sec
SSD-MobileNet-v1 Minimum Inference Latency per image (90th percentile)	8.068	milliseconds

In addition to the summary, the tool displays the throughput and latency results for each AI task configuration the benchmark tested. AIXPRT runs each AI task multiple times and reports the average inference throughput and corresponding latency percentiles.

DETAILED RESULTS (Median in case of multiple iterations):							
Workload	Inference Throughput	Inference Throughput Units	Inference Latency(50th percentile time)	Inference Latency(90th percentile time)	Inference Latency(95th percentile time)	Inference Latency(99th percentile time)	Inference Latency Units
ResNet-50_Batch 1_cpu_fp32	43.281	images/sec	21.8837	26.441401	26.46085	26.47641	milliseconds
ResNet-50_Batch 2_cpu_fp32	55.602	images/sec	35.016101	36.495399	38.70585	40.47421	milliseconds
ResNet-50_Batch 4_cpu_fp32	60.196	images/sec	65.6038	66.448003	70.73485	74.36273	milliseconds
ResNet-50_Batch 8_cpu_fp32	61.505	images/sec	128.087699	135.831505	137.7693	138.1383	milliseconds
ResNet-50_Batch 16_cpu_fp32	63.467	images/sec	249.898598	258.590013	258.6647	258.7244	milliseconds
ResNet-50_Batch 32_cpu_fp32	63.79	images/sec	498.136789	507.01791	508.4374	509.2079	milliseconds
SSD-MobileNet-v1_Batch 1_cpu_fp32	125.31	images/sec	7.9636	8.068	8.1243	8.16934	milliseconds
SSD-MobileNet-v1_Batch 2_cpu_fp32	144.972	images/sec	13.7862	13.8688	14.09595	14.27767	milliseconds
SSD-MobileNet-v1_Batch 4_cpu_fp32	154.665	images/sec	25.660699	26.147701	26.50425	26.73021	milliseconds
SSD-MobileNet-v1_Batch 8_cpu_fp32	139.745	images/sec	52.643701	72.641	73.7118	74.56844	milliseconds
SSD-MobileNet-v1_Batch 16_cpu_fp32	144.33	images/sec	109.848797	113.956802	115.6115	116.9353	milliseconds
SSD-MobileNet-v1_Batch 32_cpu_fp32	139.705	images/sec	222.8771	243.39059	254.1891	262.5254	milliseconds

Detailed instructions for using the tool, which requires running a single command, are in the AIXPRT OpenVINO on Windows readme.

BROWSING THE RESULTS TABLE

At AIXPRT.com, interested parties can view test results published by the BenchmarkXPRT Development Community. The following tips will help visitors navigate the results viewer:

- Click the tabs at the top of the table to switch from ResNet-50 network results to SSD-MobileNet or Wide and Deep network results.
- Click the header of any column to sort the data on that variable. One click sorts A-Z and two clicks sort Z-A.
- Click the link in the Source column to visit a detailed page on that result. The page contains additional test configuration and system hardware information and lets you download results files.
- You can filter results in categories such as framework, target hardware, batch size, and precision, and can designate minimum throughput and maximum latency scores. When you select a value from a drop-down menu or enter text, the results change immediately to reflect the filter.
- You can search for variables such as processor vendor or processor speed.
- The viewer displays eight results per page by default. You can change this to 16, 48, or Show all.

SUBMITTING RESULTS

We invite and encourage all AIXPRT testers to submit results from their testing for inclusion in the public results viewer. Please follow the process below to prepare results for submission:

1. After a benchmark run completes, locate the XML results file the benchmark generates at C:\Program Files (x86)\HDXPRT\Reports\Name of Run\Name of Run_Results.xml.
2. Make a copy of the results file.
3. Create an email message to the BenchmarkXPRT Community Administrator, using the address BenchmarkXPRTsupport@principledtechnologies.com and the subject "AIXPRT Results Submission."
4. Attach the results file to the message.

5. In the body of the message, specify the name of your company and the person who conducted the test.
6. Be sure that the email reply-to address you specify is a valid reply address inside your organization.

Before publishing the results to the public database, we will verify the tester's identity and validate the results. We will notify you if we publish your results.

Accessing the source code

The AIXPRT source code is available to the public via GitHub. As we've discussed in the past, publishing XPRT source code is part of our commitment to making the XPRT development process as transparent as possible. With other XPRT benchmarks, we've made the source code available to only community members. With AIXPRT, we have released the source code more widely. By allowing all interested parties, not just community members, to download and review our source code, we're taking tangible steps to improve openness and honesty in the benchmarking industry and are encouraging the kind of constructive feedback that helps to ensure that the XPRTs continue to contribute to a level playing field.

Traditional open-source models encourage developers to change products and even take them in new and different directions. Because benchmarking requires a product that remains static to enable valid comparisons over time, we allow people to download the source code and submit potential workloads for future consideration, but we reserve the right to control derivative works. This discourages a situation where someone publishes an unauthorized version of the benchmark and calls it an "XPRT."

We encourage you to download and review the source and send us any feedback you may have. Your questions and suggestions may influence future versions of AIXPRT.

Future development

With four separate machine learning toolkits on their own development schedules, three workloads, and a wide range of possible configurations and use cases, AIXPRT has more moving parts than any of the XPRT benchmark tools to date. Because there are so many different components, and because we want AIXPRT to provide consistently relevant evaluation data in the rapidly evolving AI and machine learning spaces, we anticipate a cadence of AIXPRT updates in the future that will be more frequent than the schedules we've used for other XPRTs in the past. With that expectation in mind, we want to let AIXPRT testers know that when we release an AIXPRT update, they can expect minimized disruption, consideration for their testing needs, and clear communication.

Each AIXPRT toolkit (Intel OpenVINO, TensorFlow, NVIDIA TensorRT, and Apache MXNet) is on its own development schedule, and we won't always have a lot of advance notice when new versions are on the way. Hypothetically, a new version of OpenVINO could release one month, and a new version of TensorRT just two months later. Thankfully, the modular nature of AIXPRT's installation packages ensures that we won't need to revise the entire AIXPRT suite every time a toolkit update goes live. Instead, we'll update each package individually when necessary. This means that if you only test with a single AIXPRT package, updates to the other packages won't affect your testing. For us to maintain AIXPRT's relevance, there's unfortunately no way to avoid all disruption, but we'll work to keep it to a minimum.

As we move forward, when software compatibility issues force us to update an AIXPRT package, we may discover that the update has a significant effect on results. If we find that results from the new package are no longer comparable to those from previous tests, we'll share the differences that we're seeing in our lab. As always, we will use documentation and versioning to make sure that testers know what to expect and avoid confusion about which package to use.

When we update any package, we'll make sure to communicate any updates in the new build as clearly as possible. We'll document all changes thoroughly in the package readmes, and we'll talk through significant updates in the XPRT blog.

Conclusion

We hope this paper has answered any questions you may have about AIXPRT. For more information, visit us at AIXPRT.com and BenchmarkXPRT.com. If you cannot find the answer to your question, you need help with AIXPRT, or you have suggestions on ways to improve AIXPRT, send an email to our team at BenchmarkXPRTsupport@principledtechnologies.com.

About the BenchmarkXPRT family

The BenchmarkXPRT tools are a set of apps that help you test how well devices do the kinds of things you do every day. In addition to AIXPRT, the BenchmarkXPRT suite currently comprises the following tools:

- CloudXPRT, a cloud benchmark accurately measures the performance of modern, cloud-first applications deployed on modern infrastructure as a service (IaaS) platforms, whether those platforms are on-premises, hosted elsewhere, or some combination of the two (hybrid clouds)
- WebXPRT, a browser benchmark that compares the performance of almost any web-enabled device
- HDXPRT, a benchmark to test how well Windows PCs handle real-world apps
- TouchXPRT, a Universal Windows Platform app to test the responsiveness of Windows 10 devices
- CrXPRT, an app to test the responsiveness and battery life of Chromebooks
- MobileXPRT, an app to test the responsiveness of Android devices

We designed the apps to test a wide range of devices on a level playing field. When you look at results from XPRTs, you get unbiased, fair product comparison information.

THE COMMUNITY MODEL

We built BenchmarkXPRT around a unique community model. Community membership is open to anyone, and there are many different ways to participate.

Members of the BenchmarkXPRT Development Community are involved in every step of the process. They give input on the design of upcoming versions, contribute source code, and help test the resulting implementation. Community members have access to the source code and access to early releases in the form of community previews.

The community helps us avoid the ivory tower syndrome. Diversity of input during the design process makes the tests more representative of real-world activity. Giving community members access to the source code both improves the implementation of the design and increases confidence in the code.

The community model differs from the open source model primarily by controlling derivative works. It is important that the BenchmarkXPRT benchmarks return consistent results. If the testing community calls different derivative works by the same name, the result would be that test results would not be comparable. That would limit, if not destroy, the tools' effectiveness.

If you are not currently a community member, we encourage you to join! Our community is open to everyone, from software developers to interested consumers. Not only will you get early releases of future XPRTs, but you will also be able to download the source code (available to members only) and influence the future of the tools. Register now, or for more information, see the BenchmarkXPRT FAQ.