intel XEON

White Paper

IBM

Improving IBM Watson with Intel Optimizations

Intel® Xeon® Scalable processors deliver built-in acceleration for AI workloads, enabling organizations to get more from their technology investment.

This paper is a joint effort by the engineering teams of Intel Corporation and IBM.

Executive Summary

Today, organizations recognize that AI undeniably reduces the time it takes them to extract profitable insights from their valuable data—*if* AI is implemented correctly. To help achieve that, they are adding more and more compute-intensive AI workloads at a rapid pace. That means they must develop effective AI models and deploy them at scale in production environments, and do so without slowing other workloads—and without generating excessive infrastructure costs.

In short, they must maximize the effectiveness and efficiency of their underlying infrastructures to run more workloads more quickly, and to scale existing workloads as needed.

This white paper is intended to show purchase decision makers, data scientists, and DevOps personnel the importance of using optimized libraries in their IT infrastructures. We also demonstrate how easy it is to use Intel Optimizations with Machine Learning workloads using IBM Watson Machine Learning (WML) on general purpose hardware equipped with Intel® Xeon® Scalable processors. Optimized software from Intel enables organizations to take full advantage of the built-in acceleration for AI workloads these powerful processors deliver.

IBM Watson Machine Learning

With advancements in compute, algorithms, and data access, enterprises are adopting deep learning to extract and scale insight through speech recognition, natural language processing, and image classification. Deep learning can interpret text, images, audio, and video at scale, generating patterns for recommendation engines, sentiment analysis, financial risk modeling, and anomaly detection. IBM Watson Machine Learning Accelerator, a deep learning capability in IBM Watson Studio on IBM Cloud Pak for Data, helps businesses scale compute and apps dynamically across any cloud…manage and unify large datasets and models transparently…adapt models continuously with real-time data from edge to cloud…optimize cloud and AI investments with faster training and inferencing…and much more.

IBM Watson Machine Learning helps provide real results for businesses:[1]

- 11x faster GPU performance with accelerated ML library
- 94% scaling efficiency for training from 6 to 48 GPU
- 45% inference throughput enhancement vs. open source

Table of Contents

Use cases

A wide range of enterprises and organizations can take advantage of IBM Watson Machine Learning to increase the value they draw from their data. For instance:

- **Image classification** can aid in disease diagnostics, public safety, and social media.
- **Speech-to-text recognition** can improve call center management, and can enable improved automated transcripts and mobile apps.
- **Optical Character Recognition (OCR)** can help businesses detect fraud and anomalies, validate documents automatically, and enhance cybersecurity.
- **Financial risk modeling** helps banks streamline regulatory compliance, assess credit applications, and manage client portfolios.
- **Natural Language Processing (NLP)** assists marketers in sentiment analysis, tone analysis, and brand monitoring.
- **Recommendation engines** enable retailers to predict customer behavior, customize offers and campaigns, and determine the best actions to take next.
- **Video analysis** helps agencies and organizations improve public safety, theft prevention, worker safety, and inventory management.
- And much, much more….

Intel and Artificial Intelligence

Intel has built an unparalleled AI development and deployment ecosystem, combined with a heterogeneous portfolio of AI-optimized hardware. Intel's goal is to make it as seamless as possible for every developer, data scientist, researcher, and data engineer to accelerate their AI journey from the edge to the cloud.

Through built-in hardware acceleration and optimizations for popular software tools, Intel supports an AI workflow that is streamlined from data ingestion to deployment at scale. For innovators using AI to take on great challenges, Intel is clearing the path forward to scale AI everywhere. With hardware and software to optimize AI data, modeling, and the deployment lifecycle, as well as streamline analytics at every stage, Intel can help accelerate your time to insight.

By extending Intel's AI software ecosystem, the latest software optimizations deliver performance gains of 10x to 100x[2] for popular frameworks and libraries in Deep Learning, Machine Learning, and Big Data Analytics, including TensorFlow, PyTorch, scikit-learn, XGBoost, Modin, and Apache Spark. Intel AI software enables developers to take advantage of end-to-end data science and AI workflows by accessing a rich suite of optimized libraries, frameworks, and tools for AI, such as data preparation, training, inference, and scaling. Plus, Intel AI software delivers unmatched productivity and performance. And because Intel tools are built on an open, standards-based, unified oneAPI programming model, developers can deploy them across a diverse range of systems.

With Intel AI software, developers have the flexibility to choose from bundles that include all optimized frameworks, simplifying implementation and increasing effectiveness.

Intel® AI Analytics Toolkit

The Intel AI Analytics Toolkit (AI Kit) gives data scientists, AI developers, and researchers familiar Python tools and frameworks to accelerate end-to-end data science and analytics pipelines on Intel architectures. The components are built using Intel libraries for low-level compute optimizations. This toolkit maximizes performance from pre-processing through machine learning, and provides interoperability for efficient model development.

Using this toolkit, you can:

- Deliver high-performance, deep-learning training on Intel XPUs and integrate fast inferencing into your AI development workflow with Intel technology-optimized, deep-learning frameworks for TensorFlow and PyTorch, pre-trained models, and low-precision tools.
- Achieve drop-in acceleration for data pre-processing and machine-learning workflows with compute-intensive Python packages, Modin, scikit-learn, and XGBoost, optimized for Intel.
- Gain direct access to analytics and AI optimizations from Intel to ensure that your software works together seamlessly.

Intel® Xeon® Scalable processors

Intel Xeon Scalable processors are the product of decades of innovation by the world's leading microprocessor manufacturer. They feature a proven, balanced architecture optimized for a variety of workloads. Ideal for Machine Learning and AI, Intel Xeon Scalable processors have the power to tackle tough data analytics and AI challenges, thanks to Intel-optimized software, such as the Intel® Distribution of the OpenVINO™ toolkit, the Intel® Neural Compressor, Intel® Deep Learning Boost, Intel® Advanced Vector Extensions 512, Intel® Speed Select technology, and more. Plus, Intel Xeon Scalable processors provide advanced security capabilities through Intel® Software Guard Extensions (Intel® SGX).

This built-in AI acceleration and advanced security help organizations get the most from their technology investment. With optimized performance, scale, and efficiency from the edge to the cloud, they provide customers the freedom to run workloads efficiently, effectively, and securely, so they can unlock valuable insights quickly and cost-effectively.

Intel Xeon Scalable processors are:

- Optimized for cloud, enterprise, HPC, network, security, and IoT workloads with 8 to 40 powerful cores and a wide range of frequencies, features, and power levels.
- Delivered with Intel Crypto Acceleration, enhancing data protection and privacy by increasing the performance of encryption-intensive workloads, including SSL web serving, 5G infrastructure, and VPN/firewalls, while reducing the performance impact of pervasive encryption.
- The only data center CPU with built-in AI acceleration, end-to-end data science tools, and an ecosystem of smart solutions.
- Engineered for the demands of cloud workloads across a wide range of XaaS environments.
- Built with Intel SGX, which helps protect data and application code while in use from the edge to the data center and in the multi-tenant public cloud.

Assessing Performance Benefits on Intel Hardware with Optimized Libraries

Environment used for performance testing

We executed tests on a CPD 3.5.4 cluster deployed on the Intel environment with the following configuration with cephfs storage:

All nodes have the following CPU type: Intel® Xeon® Gold 6248R processor @ 3.00GHz.

Node type	Number of servers	CPU / Node	RAM / Node
Master	3	8 cores	32 GB
Worker	24	32 cores	128 GB

On CPD 3.5.4 there are two XGBoost environments available:

1. Python 3.7 Classic/Legacy, which includes:

- scikit-learn 0.23.1
- XGBoost 0.90

2. Python 3.7 Default (OpenCE), which includes:

- scikit-learn 0.23.2
- XGBoost 1.3.3

Tests were executed on both these environments. WML version is 3.5.6.

Selecting candidate workloads, AI algorithm, and dataset to showcase benefits

First, we analyzed the WML software stack and identified core packages for training and deploying Machine Learning models. We selected Gradient Boosting and Random Forest, which are among the top ten most-used kernels[3] implemented using XGBoost and scikit-learn—which are themselves among the top five machine learning packages.[4]

The native WML stack contains sklearn (stock version) from Anaconda channel, one of the key components of Intel AI Analytics Toolkit. We used Intel Extension for sklearn to optimize the stock version for sklearn.

Benchmarking setup

Online scoring

- Concurrent scoring requests were made to simulate multiple users submitting scoring requests on the same deployment.
- Scoring requests were submitted by each simulated user repeatedly in a loop, using an established connection (HTTP Keep-Alive) for the entire test duration.
- The load on the deployed model was increased by adding more simulated users.
- We are reporting the best throughput (requests/sec or TPS) achieved at optimal response time for each test.

Batch scoring

WML supports batch scoring for various runtimes using data reference or inline data. For batch scoring, WML creates a pod based on the hardware specification mentioned in the deployment definition or in the job definition.

Usage and implementation of Intel-optimized libraries with WML

XGBoost:

For this testing, we created the XGBoost model using the HIGGS dataset with gradient booster and random forest algorithms. The tests were conditioned using XGBoost 0.90, XGBoost 1.3.3, and Intel Scikit-learn Extension for XGBoost 1.3.3.

Random Forest:

For this testing, we trained and deployed two versions of Random Forest models for HIGGS dataset, one with stock scikit-learn and the other with scikit-learn enabled with the Intel extension for sklearn.

Stock sklearn setup and run

We trained a HIGGS dataset on the native WML software stock utilizing the Jupyter notebook interface, and we deployed the trained model to score real and batch prediction. Here are a few code snippets that use sklearn to train and persist the HIGGS dataset model:

Snippet to train the RF model

```
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier

# Create our random forest classifier
clf = RandomForestClassifier(criterion=params["criterion"],
        n _ estimators=params["num _ trees"],
        max _ depth=params["max _ depth"],
        max _ features=params["max _ features"],
        min _ samples _ split=params["min _ samples _ split"],
        max _ leaf _ nodes=params["max _ leaf _ nodes"],

min _ impurity _ decrease=params["min _ impurity _ decrease"],
        bootstrap=params["bootstrap"],
        random _ state=params["seed"])
fit _ time, _ = measure _ function _ time(clf.fit, X _ test[0:100000], y _ test[0:100000], params=params)
```

Persist the model

This step requires defining the metadata with the software spec and the classifier model to persist the model prior to deployment. We used the "default_py3.7_opence" Python image and "scikit-learn_0.23" version to persist the classifier model.

```
software _ spec _ uid =
client.software _ specifications.get _ uid _ by _ name("default _ py3.7 _ opence")
metadata = {
        client.repository.ModelMetaNames.NAME: "Random forest stock on Higg dataset",
        client.repository.ModelMetaNames.TYPE: "scikit-learn _ 0.23",
        client.repository.ModelMetaNames.SOFTWARE _ SPEC _ UID : software _ spec _ uid,
}
model _ details = client.repository.store _ model(clf, metadata)
```

Deploy the model

Next, we defined the meta props, such as online/batch inference, and deployed the persisted model.

```
meta _ props = {
        client.deployments.ConfigurationMetaNames.NAME: "Predict higgs with random forest stock",
        client.deployments.ConfigurationMetaNames.ONLINE: {}
}
deployment _ details = client.deployments.create(model _ uid,meta _ props)
```

Optimized sklearn setup and run

As mentioned earlier, we added Intel Extension for sklearn to the existing sklearn package in WML. Intel Extension for sklearn is targeted to dispatch optimized calls at run time without the need to modify large amounts of code for drop-in acceleration.

Snippet to train the model

Comparing this with the stock snippet above, Intel extension for sklearn requires only importing and patching the optimizations with a few code changes. All the optimizations are out-of-box, without modifications to the actual code.

```
**********conda changes**********
from sklearnex import patch _ sklearn
patch _ sklearn()
global sklearn
**********
from sklearn.ensemble import RandomForestClassifier
```

Persist model

As explained above, model persistence requires defining the software spec. IBM WML provides support to extend the existing software spec, with additional package extensions required to run predictions. In this case, we add Intel Extension for sklearn as package extensions. More information on package extensions can be found here.

The customlibrarypyod.yaml contains the additional install instructions used to create the package extension. In this case, the .yaml file contains the command to install the sklearn extension.

```
conda install scikit-learn-intelex -c intel
```

Create metadata for package extensions.

```
base _ id = client.software _ specifications.get _ id _ by _ name('default _ py3.7 _ opence')

pe _ metadata = {
        client.package _ extensions.ConfigurationMetaNames.NAME: 'pyod custom library',
        client.package _ extensions.ConfigurationMetaNames.TYPE: 'conda _ yml'
}
pe _ asset _ details = client.package _ extensions.store(meta _ props=pe _ metadata,
file _ path='/project _ data/data _ asset/customlibrarypyod.yaml')

pe _ asset _ id = client.package _ extensions.get _ id(pe _ asset _ details)
```

Create the metadata for software specs.

```
metadata = {
        client.software _ specifications.ConfigurationMetaNames.NAME: 'Intel _ sklearn _ extension',
        client.software _ specifications.ConfigurationMetaNames.DESCRIPTION: 'Adding pyod as custom library',
        client.software _ specifications.ConfigurationMetaNames.BASE _ SOFTWARE _ SPECIFICATION: {'guid': base _ id}
}
# store the software spec
ss _ asset _ details = client.software _ specifications.store(meta _ props=metadata)
# get the id of the new asset
asset _ id = client.software _ specifications.get _ id(ss _ asset _ details)
```

Add the package extension.

```
client.software _ specifications.add _ package _ extension(asset _ id, pe _ asset _ id)
```

Now persist the model by defining the extended software spec and the classifier model.

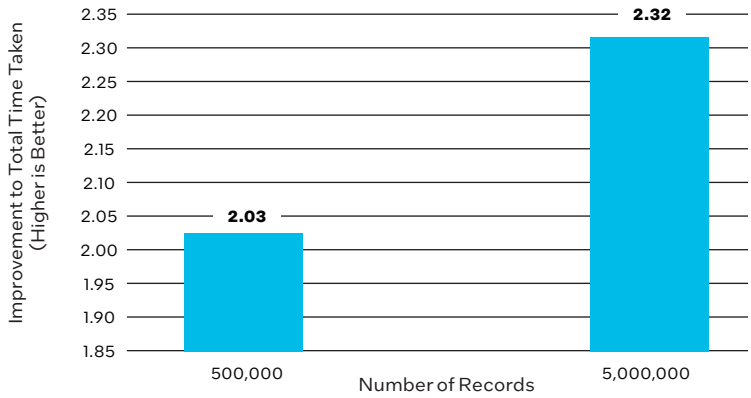
```
metadata _ model = {
        client.repository.ModelMetaNames.NAME: "Random forest on Higgs dataset opt",
        client.repository.ModelMetaNames.TYPE: "scikit-learn _ 0.23",
        client.repository.ModelMetaNames.SOFTWARE _ SPEC _ UID : asset _ id
}
model _ details = client.repository.store _ model(clf, metadata _ model)
Model deployment is similar to the stock version and can be found here
```

Results

Following are the findings of our benchmarking of key Watson Machine Learning workloads optimized for Intel architecture. As shown in these charts, customers can experience significant performance boosts in a number of different scenarios—from 2.3x improvements for XGBoost to 5x improvements for Random Forest. Performance enhancements like these are important validation that selecting Intel architecture for IBM Watson Machine Learning can have a dramatic effect on customer TCO.

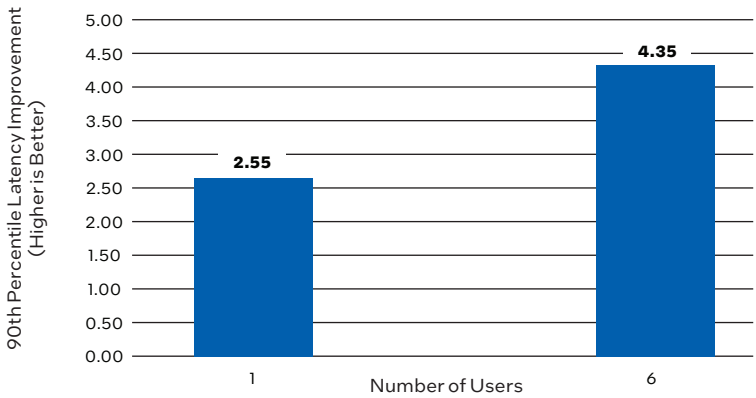
- Intel optimizations for XGBoost show up to a **2.32x** performance improvement (total time) for gradient boosting batch predictions.

Extreme Gradient Boosting Batch Performance in IBM Studio Using XGBoost 1.3.3 with Intel Optimizations vs. Stock XGBoost 0.9



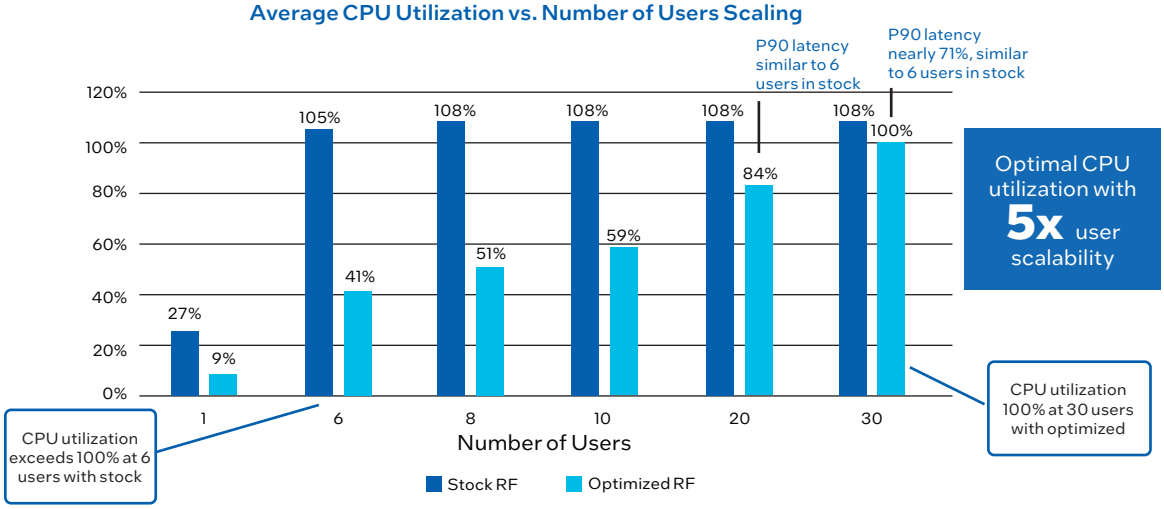
- Intel extension for scikit-learn shows up to a **4.35x** performance improvement at 90th percentile latency for real-time predictions using Random Forest.

Random Forest Real-time Inference Performance in IBM Watson Studio Using Intel-optimized Scikit-learn vs. Stock Scikit-learn



- Intel extension for scikit-learn shows optimal CPU utilization with **5x** user scaling for real-time predictions using Random Forest for small worker size configuration (CPU request: 500m. Memory request: 8GB. CPU limit: 2. Memory limit: 8GB).

Average CPU Utilization vs. Number of Users Scaling



To learn more about how your company can leverage the benefits of running IBM Watson Machine Learning on Intel architecture, please take a look at the links below...

IBM Watson Machine Learning
https://www.ibm.com/products/deep-learning-platform

Intel® Xeon® Scalable Processors
https://www.intel.com/content/www/us/en/products/details/processors/xeon/scalable.html

Optimize Data Science and Machine Learning on Intel® Xeon® Scalable Processors
https://www.intel.com/content/www/us/en/developer/videos/optimize-end-to-end-machine-learning-acceleration.html?wapkw=machine%20learning%20optimizations#gs.z9p9o3

Appendix: Definitions

- **Pod:** Kubernetes pod hosting the specific framework.
- **Concurrent Users:** The number of simulated users concurrently submitting scoring requests.
- **Response time:** The time taken for sending the scoring request from the client and the response received from the service. The following statistical values are captured.
- **Min:** Minimum scoring response time during the test duration.
- **Max:** Maximum scoring response time during the test duration.
- **Avg:** Average scoring response time for the entire test duration.
- **90%:** 90th percentile scoring response time (i.e., 90% of the requests during the test duration were completed within this time).
- **TPS:** Number of scoring requests/second.

intel XEON

1. Performance test estimate by IBM Systems and IBM Clients, 2020. https://www.ibm.com/downloads/cas/VXYXLYDY

2. Software AI accelerators: AI performance boost for free
https://www.intel.com/content/www/us/en/developer/articles/technical/software-ai-accelerators-ai-performance-boost-for-free.html#gs.88b3j8

3. State of Data Science and Machine Learning 2021, slide 32, https://www.kaggle.com/kaggle-survey-2021

4. State of Data Science and Machine Learning 2021, slide 32, https://www.kaggle.com/kaggle-survey-2021