White Paper

IBM Watson NLP Performance with Intel Optimizations

Intel® Xeon® Scalable processors deliver increased performance for IBM Watson NLP customers.

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

Lokendra Uppuluri

Cloud Software Architect, Intel Corporation

Preethi Venkatesh

AI Customer Engineering Manager, Intel Corporation

Waleed Q. Khan

Software Developer, IBM Research, IBM

Table of Contents

Abstract

In our modern world, taking advantage of Artificial Intelligence (AI) to gain insights from data is becoming more prevalent day by day. Graphical Processing Unit (GPU) systems use multiple cores to perform parallel processing, running select workloads to decrease processing times. Compared to GPUs, Central Processing Units (CPUs) have fewer cores; previously, this resulted in less capacity for parallelized processing. To move beyond this limitation, Intel has released new hardware that runs typical AI mathematical computations more efficiently on the CPU, and has also released libraries with hardware optimizations that enable an additional increase in performance. This white paper analyzes the performance impact of Intel® Optimized Libraries on AI inference workloads running on the CPU. To test the performance impact, we utilized IBM Watson NLP (Natural Language Processing)—an IBM InnerSource project delivered throughout IBM's software portfolio to products such as IBM Watson Natural Language Understanding (NLU)—in combination with Intel Optimized Libraries on supported hardware, and observed upwards of 35% improvement in overall function throughput in NLP tasks.

Introduction

AI applications are widely used to gain insights from data, assist users with queries, create content, and more. Workloads for AI applications are resource-intensive, requiring complex mathematical computations.

Typically, GPUs contain thousands of processing cores. Distributing a process to run in parallel over these multiple cores can decrease processing times due to the overhead involved. CPUs, on the other hand, contain considerably fewer cores, limiting the effectiveness of parallelization.

AI workloads are comprised of mathematical operations with effective performance that can be increased by running in parallel. Deploying them on a GPU over multiple cores improves performance. Studies of similar workloads running on both a GPU and a CPU have shown that the GPU performed 4-5 times faster than the CPU.[1]

However, the accelerated performance provided by GPUs comes with an expensive price tag. GPUs are in high demand for activities such as high-definition PC gaming, cryptomining, digital media editing, and other performance-critical applications. In addition, AI products that initiate inference requests need to deliver fast reaction times; if users experience delays in getting results for an inference request from an AI model, that can negatively impact the way they view the product. For these reasons, the demand for GPUs has outstripped the supply, resulting in shortages in the market.

Intel® Xeon® Scalable processors feature integrated AI accelerators through Intel® Deep Learning Boost technology, which enables better performance running AI workloads when compared to previous generations of CPUs. To leverage these new hardware capabilities, Intel has also released optimized versions of popular Python packages with new instruction sets. IBM Watson NLP is a popular Natural Language Processing (NLP) library used internally across IBM's software portfolio for text-based applications. This paper analyzes the performance gains achieved by Watson NLP when using Intel Optimized Libraries on supported hardware.

Intel Optimizations

Hardware Optimizations

Intel Xeon Scalable processors provide the foundation for a powerful data center platform that delivers an evolutionary leap in agility and scalability. Disruptive by design, this innovative processor enables new levels of platform convergence and enhanced capabilities across compute, storage, memory, network, and security.

Intel optimizations for AI leverage the latest processors with improved hardware features, improved instruction sets (Intel® Advanced Vector Extensions (Intel® AVX-512) and Vector Neural Network Instructions (VNNI)), as well as better memory and thread management. They also take advantage of out-of-the-box software features from widely popular Machine Learning and Deep Learning frameworks.

Intel AVX-512 boosts performance and throughput for the most demanding computational tasks in applications such as modeling and simulation, data analytics and machine learning, data compression, visualization, and digital content creation. Intel® Deep Learning Boost (Intel® DL Boost) with VNNI acceleration is built in specifically to run complex AI workloads on the same hardware as existing workloads.

Software Optimizations

Intel® AI Analytics Toolkit (Intel® AI Toolkit)

The Intel AI Toolkit offers high-performance, deep learning training on Intel® XPUs. It integrates fast inference into the AI development workflow with Intel-optimized deep learning frameworks for TensorFlow and PyTorch, pretrained models, and low-precision tools. The toolkit also delivers drop-in acceleration for data pre-processing and machine learning workflows with compute-intensive Python packages (NumPy, SciPy, Modin, scikit-learn) and XGBoost, all optimized for Intel XPUs. Optimizations explored in this study include NumPy and SciPy, as well as Intel optimizations for TensorFlow through the Intel® oneAPI Deep Neural Network Library (oneDNN).

NumPy/SciPy

Intel versions of NumPy and SciPy are optimized with the Intel oneAPI Math Kernel Library (oneMKL), replacing Eigen's compute math with oneMKL calls, providing efficient access to native FFT optimizations from a range of NumPy and SciPy functions. NumPy automatically maps operations on vectors and matrices to the BLAS and LAPACK functions wherever possible. Since oneMKL supports these de facto interfaces, NumPy can benefit from oneMKL optimizations through simple modifications to the NumPy scripts. One of the great benefits with oneMKL optimizations is the performance boost gained from leveraging SIMD and multithreading in NumPy's UMath arithmetic and transcendental operations across the range of Intel CPUs, from Intel® Core™ processors to Intel Xeon Scalable processors.

TensorFlow with oneDNN Optimizations

Intel optimizes deep learning frameworks, including TensorFlow and PyTorch, with the oneDNN library. As an open-source, cross-platform performance library of basic building blocks for deep learning applications, oneDNN uses new hardware features and accelerators available on Intel hardware. These optimizations are designed to accelerate key performance-intensive operations such as convolution, matrix multiplication, and batch normalization. oneDNN also leverages graph mode computation by fusing ops that are compute- and memory-bound to further accelerate computation.

Starting with TensorFlow release version 2.9, oneDNN optimizations are available by default. Google introduced the environment flag *TF_ENABLE_ONEDNN_OPTS* in TensorFlow 2.5 to help enable oneDNN optimizations, and users were expected to set this to "1".

Below are the key building blocks that oneDNN optimizes:

- Convolution
- Matrix multiplication
- Pooling
- Batch normalization
- Activation functions
- Recurrent Neural Network (RNN) cells
- Long Short-Term memory (LSTM) cells

Note that oneDNN Just-In-Time (JIT) compiles the operators at runtime, leveraging the later vector instruction set available on your hardware (Intel® AVX2, Intel AVX-512, or Intel AVX-512 VNNI instruction sets).

In optimizing the network, oneDNN rewrites graphs with 1:1 mapping or replaces them with custom ops based on the execution mode and custom ops support.

- **Graph mode:** oneDNN graph rewrite pass can do either 1:1 op mapping or replace a subgraph of standard TF operations with a single, fused oneDNN custom op (e.g., Conv2D+FusedBatchNorm+Relu).
- **Eager mode:** TensorFlow processes one operation at a time in eager execution. oneDNN eager op rewrite pass only performs 1:1 operation mappings (replacing the operation with its corresponding oneDNN operation, if one exists).

IBM Watson NLP

IBM Watson NLP is a state-of-the-art, text-based Natural Language Processing solution. It has been widely adopted within IBM and is bundled in more than 20 IBM products. Some examples of product use cases for IBM Watson NLP include:

- IBM Watson Natural Language Understanding (NLU) uses Watson NLP to extract meaning and metadata from unstructured text and data.
- IBM Watson Discovery uses Watson NLP for search and text-analytics from data.
- IBM Watson Studio is a data science platform that directly exposes capabilities from the Watson NLP library to users.

IBM Watson NLP Tasks

IBM Watson NLP takes advantage of the concept of "NLP tasks" to divide algorithms by the schema of their responses. An example of this is the "classification" task, which can be implemented by a wide variety of algorithms. All algorithm implementations for an NLP task adhere to a base Application Programming Interface (API). This makes the utilization of different algorithms easier for the user, as algorithms can be swapped without changes to the consuming code. An algorithm can support multiple languages, but another algorithm in the same task category may not support the same number of languages. The Watson NLP documentation website lists all the supported languages for each algorithm available.

Experiments

IBM Watson NLP provides a performance test suite developed for inference testing for each algorithmic implementation. The input for each inference request is curated by analyzing customer usage, so testing is done on real-world scenarios.

Performance data collection experiments run on two machines with identical specifications. Table 1 provides the hardware and Operating System (OS) specifications of the machines running the performance tests.

For performance testing, Watson NLP was built and runs inside a docker image. This ensures environment consistency between tests. The resources for the docker container are limited to 1 CPU and 20 GB of RAM. All performance tests run in a sequential manner, and no parallelism is invoked. The performance run collects metrics of interest while the tests are running and outputs the results in a comma-separated values (CSV) file for easy consumption.

OS	Ubuntu 18.04.5 / GNU Linux
Kernel Version	4.15.0-135-generic
Platform	x86_64
CPU(s)	40
Byte Order	Little Endian
CPU Model	Intel® Xeon® Silver 4210 CPU
Thread(s) Per Core	2
CPU Speed	2.20 GHz
RAM	791 GB

Table 1. Performance machine specifications

Collected Metrics

The performance test suite collects a variety of metrics for analyzing Watson NLP's performance over a broader scope. However, the key metric assessed to measure the impact on performance is *Function Throughput (Kcodepoints/sec)*.

For all algorithms tested, the size of the raw text sent for inference is tracked, as is the time required by the algorithm to process the input. *Function Throughput (Kcodepoints/sec)* represents the total size of raw text in Kcodepoints sent to the algorithm over the time needed to process that input. It is important to note that the recorded time used for calculating Function Throughput is the processing time for the algorithm under test; any time spent pre-processing the raw text into the correct format is recorded in a separate variable. Additionally, if multiple inference requests are made to the algorithm as part of the tests, the Kcodepoints size and duration are cumulative.

The package test-suite also collects metrics for CPU and RAM utilization. Since the tests are run sequentially in a docker container, analyzing the CPU and RAM statistics of the container results in a close approximation of the resource utilization for each algorithm. Figures for CPU and RAM utilization are collected by running "docker stats" on the container at a two-second interval while the algorithm's test is running.

To run the experiments, the following two builds/branches are tested and compared against one another.

1. **Base Build** is the base experiment, using Watson NLP.
2. **Intel Build** is the optimized build, using Watson NLP with oneDNN optimizations enabled in TensorFlow.

Table 2 specifies the versions of key software and packages used in the builds.

Python	3.8.8
NumPy	1.21.4
SciPy	1.6.2
TensorFlow	2.8

Table 2. Software and package version

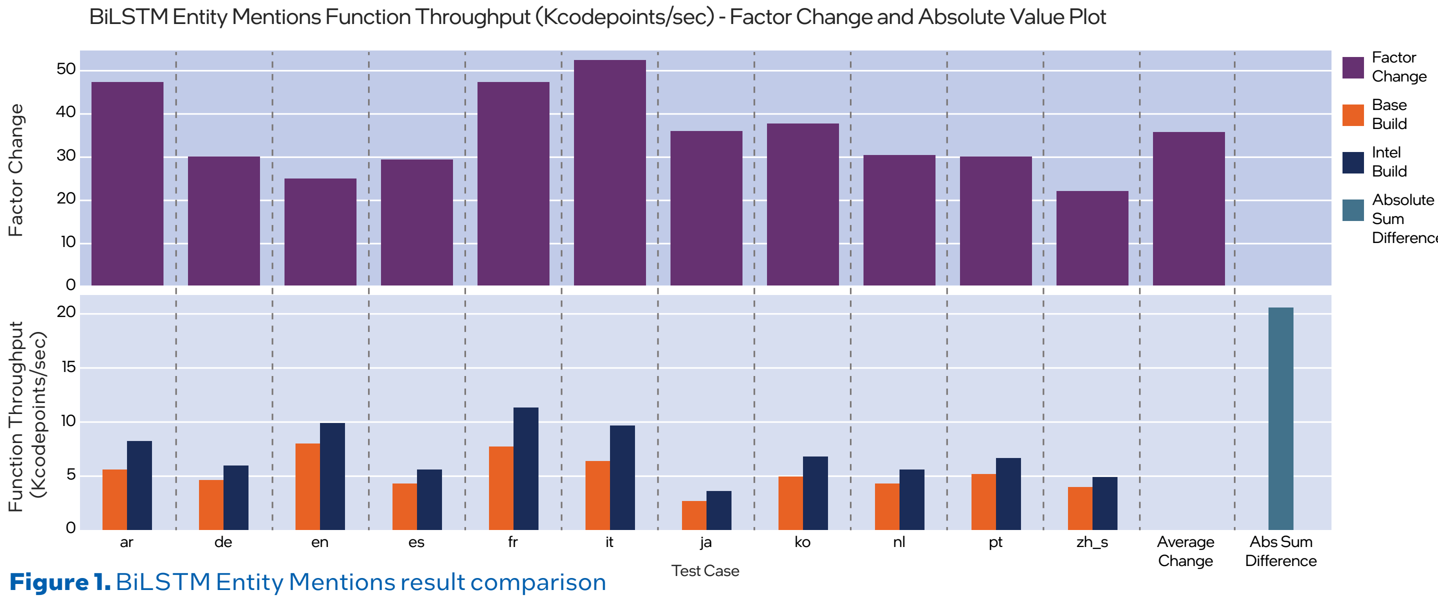

Figure 1. BiLSTM Entity Mentions result comparison

Results and Analysis

BiLSTM Entity Mentions

BiLSTM Entity Mentions is a Bidirectional Long Short-Term Memory (BiLSTM) neural network model that leverages character- and word-level representations to extract entities (e.g., people, organizations, and dates) from text. Representations are modeled in both forward and backward directions, and are eventually processed by additional statistical modeling techniques to ensure that the tag assigned to each word in a sequence takes into account the tags of its neighbors. This typically enables more cohesive results. Entity extraction is useful for a wide variety of tasks, such as query understanding in search engines and chatbot interactions with humans.

The *Function Throughput* results are shown in Figure 1. The model supports multiple languages (such as English, French, and Spanish), marked along the x-axis. The plot shows that the model performs faster for all supported languages when using the Intel Build.

Document BERT Sentiment

Document BERT Sentiment is a model that leverages the BERT (Bidirectional Encoder Representations from Transformers) language model for the task of document sentiment classification. BERT uses bidirectional representations in a masked context; it has been shown to be effective on a wide variety of NLP tasks, especially those where context is highly important. Document Sentiment Analysis is used for many practical applications, such as identifying unhappy customers and social media analysis.

The Document Bert Sentiment model supports more than 20 different languages. The results for *Function Throughput* can be viewed on Figure 2. The average percentage increase across the model is approximately 15% with the Intel Build—a substantial increase in performance for this class of use case workload.

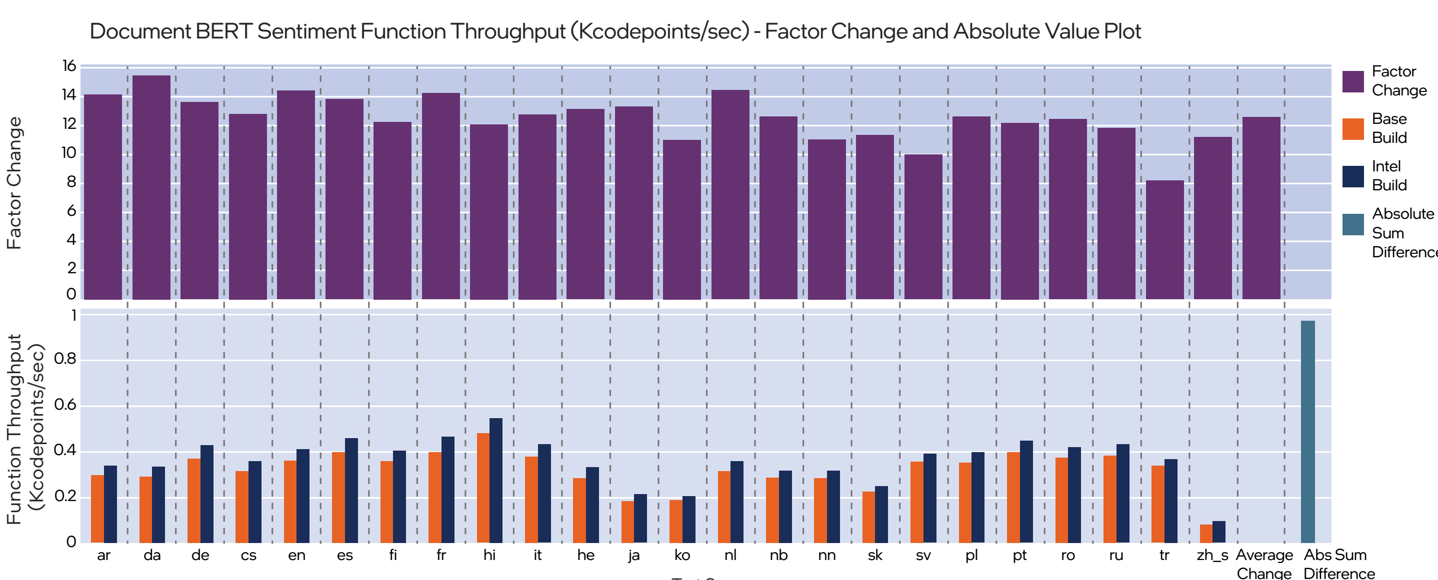

Figure 2. Document Bert Sentiment result comparison

Conclusion

This study presents an analysis of the performance of Watson NLP inference requests running on CPUs. The primary objective was to study the effect of Intel Optimized Libraries for ML/AI workloads on supported hardware. Two builds of Watson NLP were compared: one using Intel-optimized packages and the other without. Performance testing for inference requests was performed on these builds using the multiple algorithms available in Watson NLP. The tests were run multiple times to account for variability. When using Intel oneDNN TensorFlow optimizations, IBM Watson NLP exhibited an increase of up to 35% in function throughput for NLP tasks including text and sentiment classification, and embeddings. This performance improvement has a positive effect on the overall performance of IBM products such as IBM Watson Natural Language Understanding (NLU).

Incorporating Intel Optimized Libraries on supported hardware for applicable AI/ML workloads can increase performance. No discernable effect on CPU and memory utilization was recorded between the two builds.

Learn more at:
https://www.intel.com/content/www/us/en/partner/showcase/ibm/overview.html

References

1. E. Buber, B. Diri, Performance Analysis and CPU vs GPU Comparison for Deep Learning, in: 2018 6th International Conference on Control Engineering & Information Technology (CEIT), IEEE, 2018, pp. 1–6. https://www.researchgate.net/publication/334168063_Performance_Analysis_and_CPU_vs_GPU_Comparison_for_Deep_Learning