intel AI

How to make the right hardware choice for

DEEP LEARNING
TRAINING AND INFERENCE APPLICATIONS

There are many factors to bear in mind when addressing deep learning training and inference. Here's a quick guide to help you identify the optimal tool to meet your specific needs.

BEFORE YOU GET STARTED, CONSIDER THESE QUESTIONS:

1 How often do you need to train your model, both initially and after it has been deployed?

2 What kind of data are you using as your training set?

3 Can you train during off-peak hours?

4 What are your cost constraints?

5 What type of hardware architecture do you already have in your workstations or server rack?

6 Can you deal with the software complexities of multiple architectures?

TRAINING

Though only 10 to 15% of a typical workflow, training a model is a key step in harnessing the power of artificial intelligence. In this phase, the algorithm seeks to learn features and patterns from the data you feed it, later applying that knowledge to unseen data. Think of showing millions of examples of bone images to an algorithm that will be trained to identify bone density for radiologists. Training can take several hours to several days.

CPUs

For many applications—such as high-definition-, 3D-, and non-image-based deep learning on language, text, and time-series data—CPUs shine. This is especially true for memory-intensive data, including massive amounts of unstructured data as well as sparse data. Infrequent training (fewer than 10 times per year) may also be a factor in staying on CPUs.

Pros

1. AI applications can be run side by side with other applications, maximizing hardware utilization
2. Larger, memory-intensive models can be trained with greater ease
3. Data can be accessed from the same infrastructure on which you train, saving the time it would take to port data from one architecture to another
4. Utilization can be maximized, which can contribute to improved total cost of ownership
5. Public cloud instances of CPUs are typically far less expensive than GPUs, especially when training for extended periods on large models
6. If your training workload is not time sensitive, CPUs may be a viable alternative to GPUs

Cons

1. If you are training models frequently, the lower speed can potentially cost you valuable time
2. The same training will typically require more cycles with a CPU than when using a GPU

GPUs

For deep learning training with several neural network layers or on massive sets of certain data, like 2D images, GPUs or other accelerators can be your best bet. GPUs also tend to be the better choice for fast deep learning as the simple matrix math calculations greatly benefit when computations can be done in parallel.

Pros

1. Fast training on certain types of data

Cons

1. Hardware may not be utilized to full capacity
2. Memory limitations require you to break down images or your model, resulting in more work and/or subpar results
3. Data must be ported from one architecture to another when transferred from GPUs to CPUs in the data center
4. Can add cost, complexity, and operational expenses

INFERENCE

After your model is trained, you'll put it to work with inference, or the inferring of something about data it has never seen before. Inference cycles already surpass those of training by anywhere from 10x to 1000x, depending on the application. Today, inference primarily runs on CPUs, and a typical deployment will use a mix of neural networks and other types of compute power. Facebook uses CPUs for 100% of its current inference applications.

As deep learning has been adopted more broadly, there has been a clear shift in the ratio between cycles of inference and training. Our conservative internal estimate predicts a move from 1:1 in the early days of deep learning to potentially more than 10:1 by 2020. With inference taking a large majority of the workflow, it is critical to use hardware architectures well suited to those needs, meaning low latency and often low power.

CPUs

Recent hardware changes have enhanced the naturally good inference performance of CPUs. Coupled with all-new software optimizations and tools, this means the CPU has never been more performant for AI inference. There is also unprecedented diversity in acceleration hardware for robust and sustained inference applications. As inference is not as resource heavy as training, CPUs are economical as well.

Pros

1. Real-world applications have increasingly stricter latency, a CPU strength
2. The newest CPUs can support much more of the system memory required for complex models
3. CPUs can extend further to the edge, including devices, unlike GPUs, allowing a similar architecture from training through deployment
4. CPUs are well suited to new options like using pretrained models or transfer learning
5. CPUs already power laptops, workstations, robotics, some phones and smart speakers, and vehicles, etc.

Cons

Given today's inference demands and the capabilities of current technology there are no significant downsides to using a CPU for inference.

GPUs

GPUs excel at performing matrix operations that relied on by graphics, AI, and many scientific algorithms. Due to their parallel computing capabilities, GPUs can be useful for inference. But as AI models continue to grow in complexity, inference for real-world deployments increasingly favors CPUs.

Pros

1. When inference speed is a bottleneck, GPUs can provide financial and time gains
2. By design, GPUs can work well for tasks like image recognition
3. GPUs may be a cheaper option today

Cons

1. GPUs have inherent memory constraints
2. GPUs can be power hungry, which conflicts with the needs of edge devices
3. GPUs are less common in infrastructure that may already need to be utilized and upgraded to support AI applications

Though we focus here on CPUs and GPUs, there are certain inference environments that are not especially well suited to CPUs and GPUs, such as very small devices. Even for data center or cloud-based inference, sometimes the workloads are simply too sustained or of too high a volume for CPUs or GPUs to make economical sense. In these cases, other tools may be preferred, including vision processing units (VPUs), application-specific integrated circuits (ASICs), and field-programmable gate arrays (FPGAs).

VPUs – VPUs are ultralow-power computer vision engines and offer capabilities for inference for low-power devices such as smart cameras and network video recorders.

ASICs – Technically, a GPU is an ASIC used for processing graphics algorithms. A custom ASIC is dedicated to performing fixed operations extremely fast since the entire chip's logic area can be focused on a narrow set of functions, making them suitable for a high degree of parallelism.

FPGAs – With particular value for high-throughput, low-latency inference applications, FPGAs offer a unique blend of flexibility, performance, and extensibility unmatched by custom ASICs or GPUs.

2nd Gen Intel® Xeon® Scalable processor platform performance

AI includes a wide variety of machine learning and analytics techniques, but its connection with deep learning has most captured the interest and imagination of today's innovators.

Intel® Xeon® Scalable processors are optimized specifically to run high-performance deep learning inference. Most deep learning, including computer vision and inference, already runs on Intel® Xeon® processors as they are the foundation of many of the world's data centers. The hardware performance of AI applications can further benefit from software optimizations.

Our new 2nd Generation Intel Xeon Scalable processors have AI built in, offering significant leaps in inference performance, memory, and bandwidth that accelerate complex AI applications. The processors have been enhanced with substantial improvements in software optimizations and hardware instructions, giving you the flexibility you need for both AI and the vast range of data-centric applications.

Intel® Deep Learning Boost technology – This cross-platform tool features a model optimizer and inference engine to streamline and simplify model deployment. Access a new set of embedded accelerators (vector neural network instructions, or VNNIs) to speed up the dense computations of convolutional neural networks (CNNs) and deep neural networks (DNNs). The low-precision integer operations deliver up to 30x improvement in inference performance.[2]

Intel® Optane™ DC persistent memory – Achieve up to triple the maximum storage per node and enable more memory closer to the CPU so data can be sustained even throughout power cycles or system maintenance.

This is AI on Intel.

UP TO **277X**[1] INFERENCE THROUGHPUT

achieved using an Intel® Xeon® Platinum 8180 processor running Intel-optimized Caffe* GoogLeNet* v1 with Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN), vs. an Intel® Xeon® processor E5-2699 v3 with BVLC-Caffe*

UP TO **30X**[2] INFERENCE THROUGHPUT IMPROVEMENT

on Intel® Xeon® Platinum 9282 processor with Intel® Deep Learning Boost (Intel® DL Boost)

UP TO **241X**[1] TRAINING THROUGHPUT

achieved using an Intel Xeon Platinum 8180 processor running Intel-optimized Caffe AlexNet* with Intel MKL-DNN, vs. an Intel Xeon processor E5-2699 v3 with BVLC-Caffe[1]

To boost performance, we've also optimized the software tools and frameworks widely used today for Intel Xeon Scalable processor-based platforms:

TensorFlow | mxnet | Caffe2 | PaddlePaddle

The best results are achieved when the right tool is used for the job. Today's CPUs and GPUs each boast distinct benefits, but neither is perfect for every environment or goal. To unlock the greatest possible impact for your deep learning application, ensure that you are using the optimal solution throughout training and inference.

Learn more about deep learning, and how Intel is powering AI across an exciting set of industry use cases, by visiting intel.ai.

If you are a developer, find tools and resources at software.intel.com.

For more information on Intel Enterprise Solutions for AI, please contact us today.

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information, visit intel.com/performance.

Intel does not control or audit the design or implementation of third-party benchmark data or websites referenced in this document. Intel encourages all of its customers to visit the referenced websites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase.

Optimization notice: Intel® compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel® microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel® microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Performance results are based on testing as of 06/15/2018 (v3 baseline), 05/29/2018 (241x), and 06/07/2018 (277x) and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Up to 30X AI performance with Intel® Deep Learning Boost (Intel DL Boost) compared to Intel® Xeon® Platinum 8180 processor (July 2017). Tested by Intel as of 2/26/2019. Platform: Dragon rock 2 socket Intel® Xeon® Platinum 9282(56 cores per socket), HT ON, turbo ON, Total Memory 768 GB (24 slots/ 32 GB/ 2933 MHz), BIOS: SE5C620.86B.0D.01.0241.112020180249, Centos* 7 Kernel 3.10.0-957.5.1.el7. x86_64, Deep Learning Framework: Intel® Optimization for Caffe* version: https://github.com/intel/caffe d554cbf1, ICC 2019.2.187, MKL DNN version: v0.17 (commit hash: 830a10059a018cd-2634d94195140cf2d8790a75a), model: https://github.com/intel/caffe/blob/master/models/intel_optimized_models/int8/resnet50_int8_full_conv.prototxt, BS=64, No datalayer DummyData: 3x224x224, 56 instance/2 socket, Datatype: INT8 vs Tested by Intel as of July 11th 2017: 2S Intel® Xeon® Platinum 8180 cpu @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS* Linux release 7.3.1611 (Core), Linux kernel* 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC). Performance measured with: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (https://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50),. Intel C++ compiler ver. 17.0.2 20170213, Intel® Math Kernel Library (Intel® MKL) small libraries version 2018.0.20170425. Caffe run with "numactl -l".

1. INFERENCE using FP32 Batch Size Caffe* GoogLeNet* v1 128 AlexNet* 256.

   Configurations for inference throughput:

   Tested by Intel as of 06/07/2018: Platform: 2-socket Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz/28 cores HT ON; turbo: ON, total memory 376.28 GB (12 slots/32 GB/2666 MHz), four instances of the framework, CentOS* Linux*-7.3.1611-Core, SSD sda RS3WC080 HDD 744.1 GB, sdb RS3WC080 HDD 1.5 TB, sdc RS3WC080 HDD 5.5 TB, deep learning framework Caffe* version: a3d5b022fe026e9092fc7abc7654b1162ab9940d; topology: GoogLeNet* v1 BIOS*: SE5C620.86B.00.01.0004.071220170215 MKL-DNN: version: 464c268e544bae26f9b85a2acb9122c766a4c396; NoDataLayer. Measured: 1449 imgs/sec vs. tested by Intel as of 06/15/2018; Platform: 2S Intel® Xeon® CPU E5-2699 v3 @ 2.30 GHz (18 cores), HT enabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 64 GB DDR4-2133 ECC RAM. BIOS: SE5C610.86B.01.01.0024.021320181901, CentOS Linux-7.5.1804 (Core) kernel 3.10.0-862.3.2.el7.x86_64, SSD sdb INTEL SSDSC2BW24 SSD 223.6 GB. Framework: BVLC-Caffe: github.com/BVLC/caffe, inference and training measured with "Caffe time" command. For "ConvNet" topologies, dummy data set was used. For other topologies, data was stored on local storage and cached in memory before training. BVLC Caffe (github.com/BVLC/caffe), revision 2a1c552b66f026c7508d390b526f2495ed3be594.

   Configuration for training throughput:

   Tested by Intel as of 05/29/2018: Platform: 2-socket Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz/28 cores HT ON; turbo: ON, total memory 376.28 GB (12 slots/32 GB/2666 MHz), four instances of the framework, CentOS* Linux*-7.3.1611-Core, SSD sda RS3WC080 HDD 744.1 GB, sdb RS3WC080 HDD 1.5 TB, sdc RS3WC080 HDD 5.5 TB, deep learning framework Caffe* version: a3d5b022fe026e9092fc7abc765b1162ab9940d; topology: AlexNet BIOS: SE5C620.86B.00.01.0004.071220170215 MKLDNN: version: 464c268e544bae26f9b85a2acb9122c766a4c396; NoDataLayer. Measured: 1257 imgs/sec vs. tested by Intel as of 06/15/2018; platform: 2S Intel® Xeon® CPU E5-2699 v3 @ 2.30 GHz (18 cores), HT enabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 64 GB DDR4-2133 ECC RAM. BIOS: SE5C610.86B.01.01.0024.021320181901, CentOS Linux-7.5.1804 (Core) kernel 3.10.0-862.3.2.el7.x86_64, SSD sdb INTEL SSDSC2BW24 SSD 223.6 GB. Framework: BVLC-Caffe: github.com/BVLC/caffe, inference and training measured with "Caffe time" command. For "ConvNet" topologies, dummy data set was used. For other topologies, data was stored on local storage and cached in memory before training. BVLC Caffe (github.com/BVLC/caffe), revision 2a1c552b66f026c7508d390b526f2495ed3be594.

   Performance results are based on testing as of 06/15/2018 (v3 baseline), 05/29/2018 (241x) and 6/07/2018 (277x) and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

2. Tested by Intel as of 02/26/2019. Platform: Dragon rock 2 socket Intel® Xeon® Platinum 9282 processor (56 cores per socket), HT ON, turbo ON, Total Memory 768 GB (24 slots/32 GB/2933 MHz), BIOS:SE5C620.86B.0D.01.0241.112020180249, CentOS* 7 Kernel 3.10.0-957.5.1.el7.x86_64, Deep Learning Framework: Intel® Optimization for Caffe* version: https://github.com/intel/caffe d554cbf1, ICC 2019.2.187, MKL-DNN version: v0.17 (commit hash: 830a10059a018cd2634d94195140cf2d8790a75a), model: https://github.com/intel/caffe/blob/master/models/intel_optimized_models/int8/resnet50_int8_full_conv.prototxt, BS=64, No data layer syntheticData:3x224x224, 56 instance/2 socket, Datatype: INT8 vs. Tested by Intel as of 07/11/2017: 2S Intel® Xeon® Platinum 8180 processor CPU @ 2.50 GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384 GB DDR4-2666 ECC RAM. CentOS Linux* release 7.3.1611 (Core), Linux* kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD Data Center S3700 Series (800 GB, 2.5in SATA 6Gb/s, 25nm, MLC). Performance measured with: Environment variables: KMP AFFINITY= "granularity=fine, compact," OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, synthetic dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models(ResNet-50). Intel® C++ Compiler ver. 17.0.2 20170213, Intel® Math Kernel Library (Intel® MKL) small libraries version 2018.0.20170425. Caffe run with "numactl -l."

Intel® technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com/benchmarks.

intel