ARTIFICIAL INTELLIGENCE 501

Lesson 8
Software

LEGAL NOTICES & DISCLAIMERS

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon and others are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.

Learning Objectives

You will be able to:

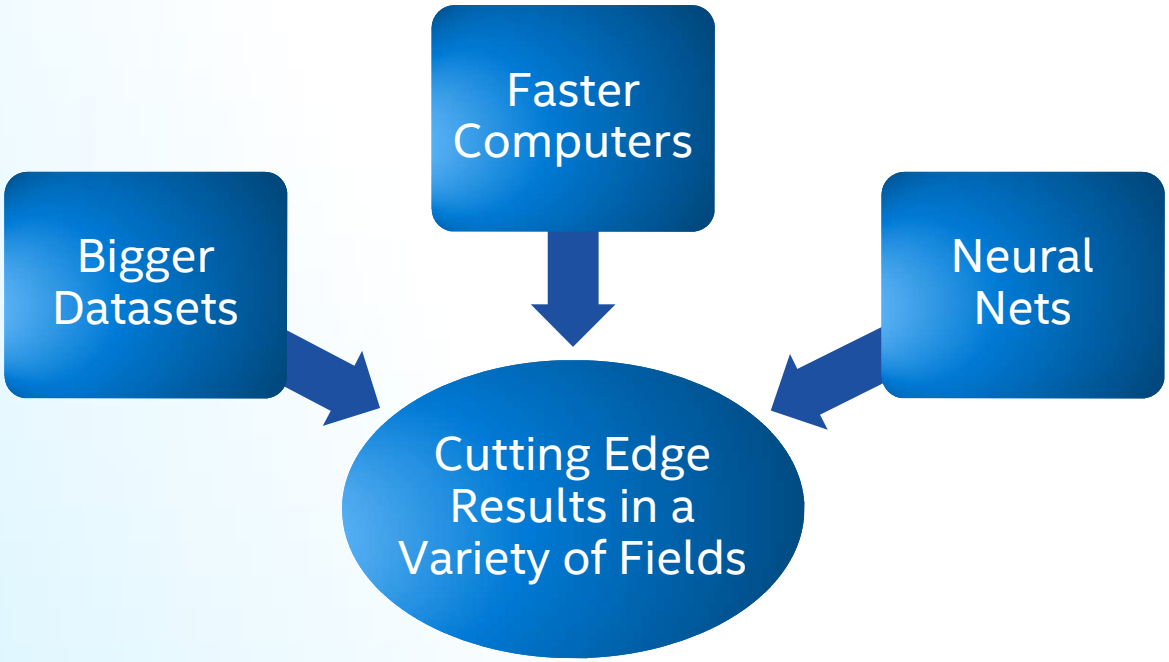
- Differentiate between Intel® optimized libraries and frameworks
- Describe the interplay of Intel® hardware, libraries, and frameworks
- Explain the usefulness of Intel® Nervana™ graph
- Explain the impact of big data on software
- Evaluate the application of Apache Spark*

Intel® AI Portfolio

Frameworks


intel
AI DEVELOPER

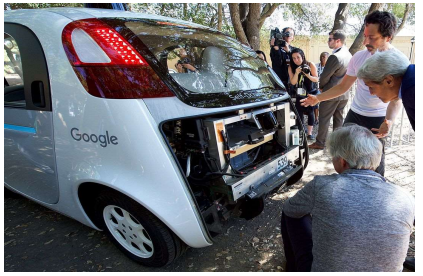
How Is This Era of AI Different?



Bigger Datasets

In 2020, it is expected that:

- The average internet user will generate ~1.5 GB of traffic per day.
- A smart hospital will generate 3,000 GB/day.
- Self-driving cars are each generating over 4,000 GB/day.
- Connected planes will generate 40,000 gigabytes per day.
- A connected factory will generate 1 million gigabytes per day.



Artificial Intelligence (AI) Software

As datasets get bigger the need for optimized software becomes increasingly important.

- The computation required increases with the size of the dataset.
- Software needs to be optimized for the underlying hardware to take full advantage of faster computers.
- The software and hardware need to be optimized for specific mathematical operations for optimal performance.

AI Software

It is usually inefficient for AI developers to write software to manage and analyze this data themselves.

- Developing software optimized for hardware and mathematical operations is time consuming.
- As methods become increasingly sophisticated the code for mathematics become more error prone.

AI Developer

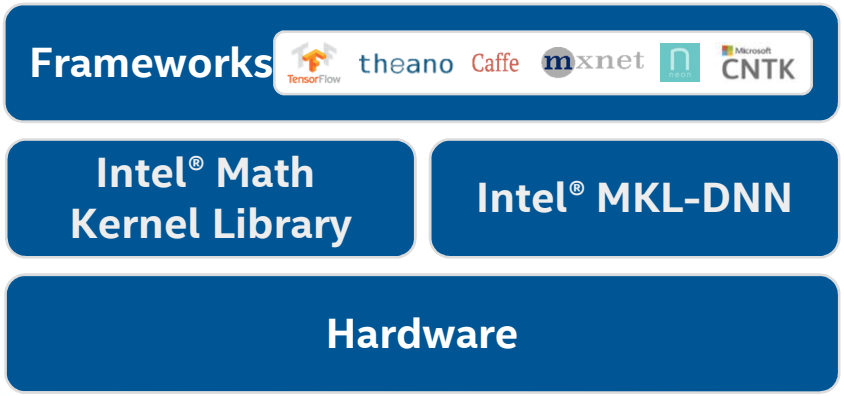
An AI developer needs to leverage existing code so they can focus on the big picture and overall system framework.

- Use code that's optimized for the underlying hardware since AI tasks require a large amount of computation to complete.
- Simplify model development and training by providing high-level primitives for complex and error-prone mathematical transformations.
- Understand and use algorithms by stringing together API calls.
- Take advantage of software libraries and frameworks.

AI Developer

Intel® has developed highly optimized math libraries on top of hardware.

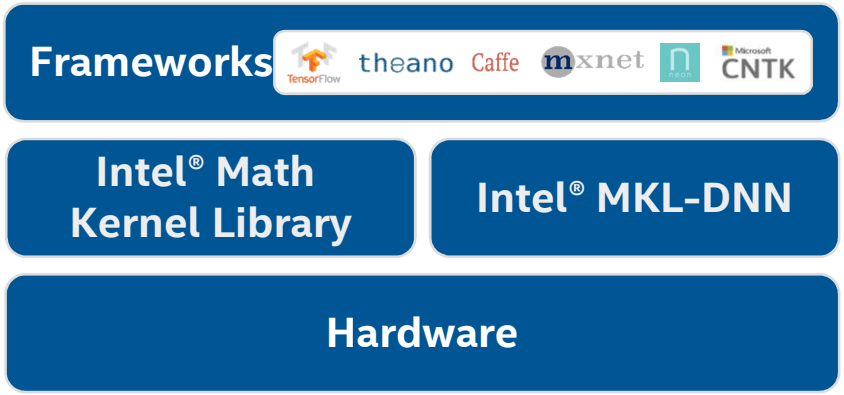
- These libraries make efficient use of Intel® processors.
- Optimized deep learning and machine learning frameworks built on top of Intel libraries.
- Libraries can be used to build frameworks that efficiently run deep learning on a variety of platforms



AI Developer

Throughout this lesson we will cover the existing software pieces that can be leveraged.

- Background on software libraries and frameworks.
- Libraries optimized for underlying hardware.
- Frameworks built on top of these libraries.




intel
LIBRARIES & FRAMEWORKS

Software Library

A **software library** is a set of code that provides functionality that can be called from your own code.

- Using libraries is advantageous for developers since they can use others' code without needing the knowledge to write it themselves.
- Encourages modular code development, where distinct groups of functionality are grouped together into distinct libraries.
- Due to libraries' usefulness, all languages include functionality to use code from external libraries within a program

Software Frameworks

A **software framework** provides generic functionality that can act as a skeleton architecture to accomplish a particular task.

- A framework dictates the flow of control and defines the overall nature of the program where common design patterns can be reused.
- Frameworks can be extended by the users to provide specific functionality and customizations.
- Frameworks provide a standard way to build and deploy applications.
- Multiple frameworks can use the same underlying libraries.

Deep Learning Frameworks

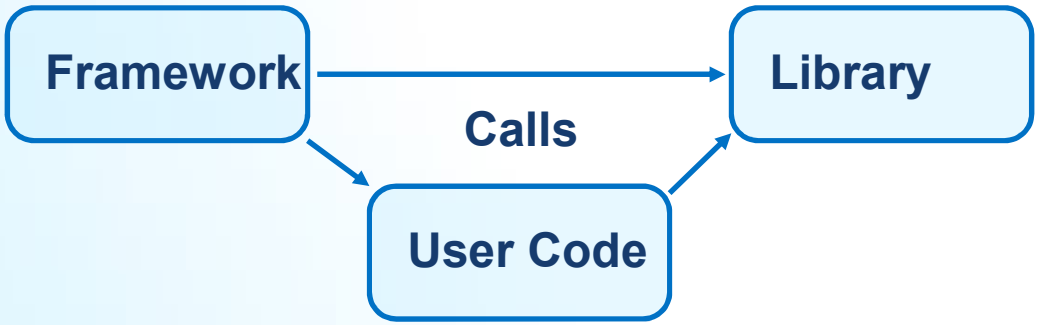
Deep Learning frameworks vary in their level of functionality.

- Some allow you to define a neural network of arbitrary complexity from the most basic building blocks
- Others act as drivers or wrappers aimed at boosting developer productivity but are limited in their functionality due to the higher level of abstraction.

Inversion of Control

Inversion of control refers to the flow of control being inverted when working with frameworks.

- Typically, custom code calls libraries for tasks.
- Frameworks invert control and call custom code for customized or specific tasks.




INTEL® LIBRARIES

Intel® Math Kernel Library

Intel® Math Kernel Library (Intel® MKL) is a library of math routines that are optimized specifically for Intel® processors.

- Highly-optimized performance primitives for Deep Learning (DL) applications, as well as functions that have been highly optimized for single-core vectorization and cache memory utilization.
- Features highly-optimized, threaded and vectorized functions to maximize performance on Intel® architecture and compatible processors.
- Uses standard C and Fortran APIs for compatibility with BLAS, LAPACK and FFTW functions

Intel® MKL for Deep Neural Networks

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) is a library of deep neural network primitives for rapid integration into DL frameworks and optimized for Intel architectures.

- Based on Intel® MKL library.
- Common DNN APIs across all Intel® architecture.
- Includes highly vectorized and threaded building blocks for implementing convolutional neural networks with C and C++ interfaces.
- Implemented in C++ and provides both C++ and C APIs.
- Allows the functionality to be used from a wide range of high-level languages, such as Python* or Java*.

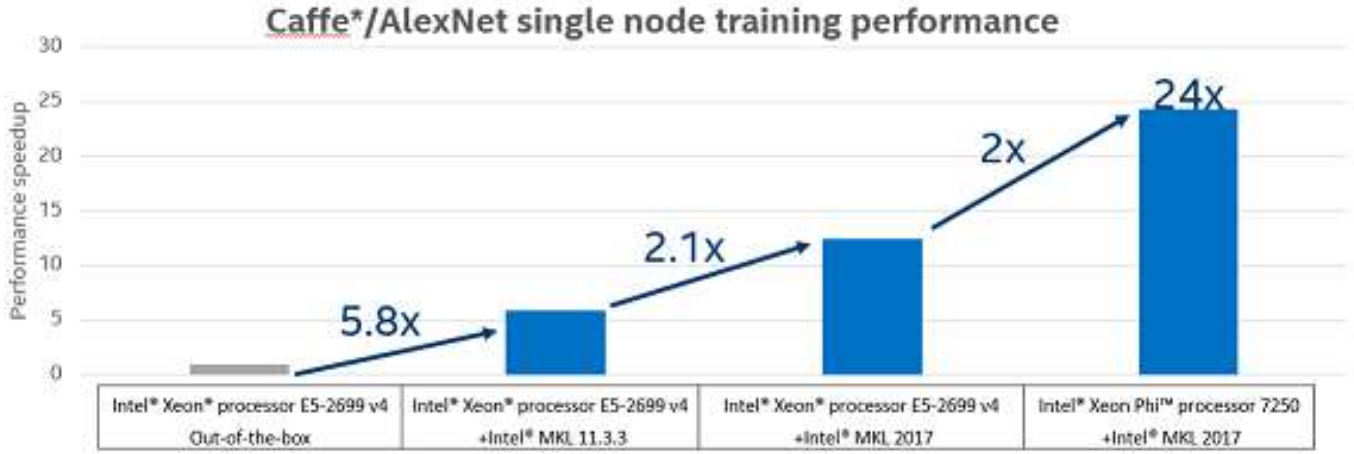
Intel® DNN Performance Benchmarks

The AlexNet* DL topology is typically used as a DL benchmark.

- Intel's version uses MKL-DNN primitives on Intel® Xeon® Processors
- The following graph (on the next page) compares the standard vs Intel® optimized Caffe* implementation on the Intel Xeon processors.

Intel® DNN Performance Benchmarks

Better performance in Deep Neural Network workloads with Intel® Math Kernel Library (Intel® MKL)



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 information go to http://www.intel.com/performance. *Other names and brands may be property of others
Configurations:

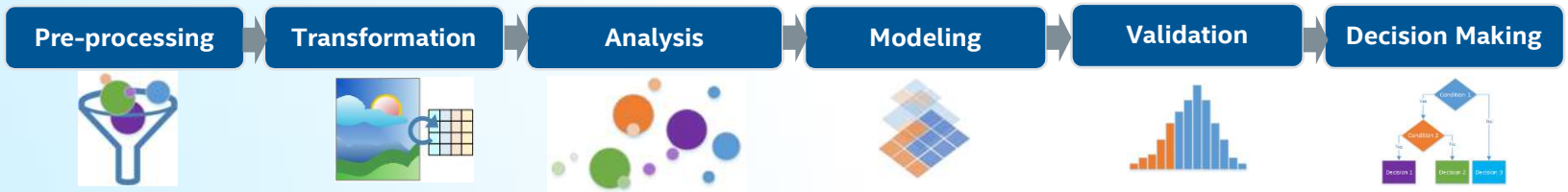
- 2 Socket system with Intel® Xeon Processor E5-2699 v4 (22 Cores, 2.2 GHz), 128 GB memory, Red Hat* Enterprise Linux 6.7, BVLC Caffe, Intel Optimized Caffe framework, Intel® MKL 11.3.3, Intel® MKL 2017
- Intel® Xeon Phi™ Processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM), 128 GB memory, Red Hat* Enterprise Linux 6.7, Intel® Optimized Caffe framework, Intel® MKL 2017

All numbers measured without taking data manipulation into account.

Intel® Data Analytics Acceleration Library

Intel® Data Analytics Acceleration Library (Intel® DAAL) provides highly optimized algorithmic building blocks for data analytics and machine learning.

- Includes building blocks for all data analytics states.
- Pre-processing, transformation, analysis, modeling, validation, and decision making.



Intel® Data Analytics Acceleration Library

Designed to use with popular data platforms including Hadoop*, Spar*, R*, and MATLAB* for highly efficient data access.

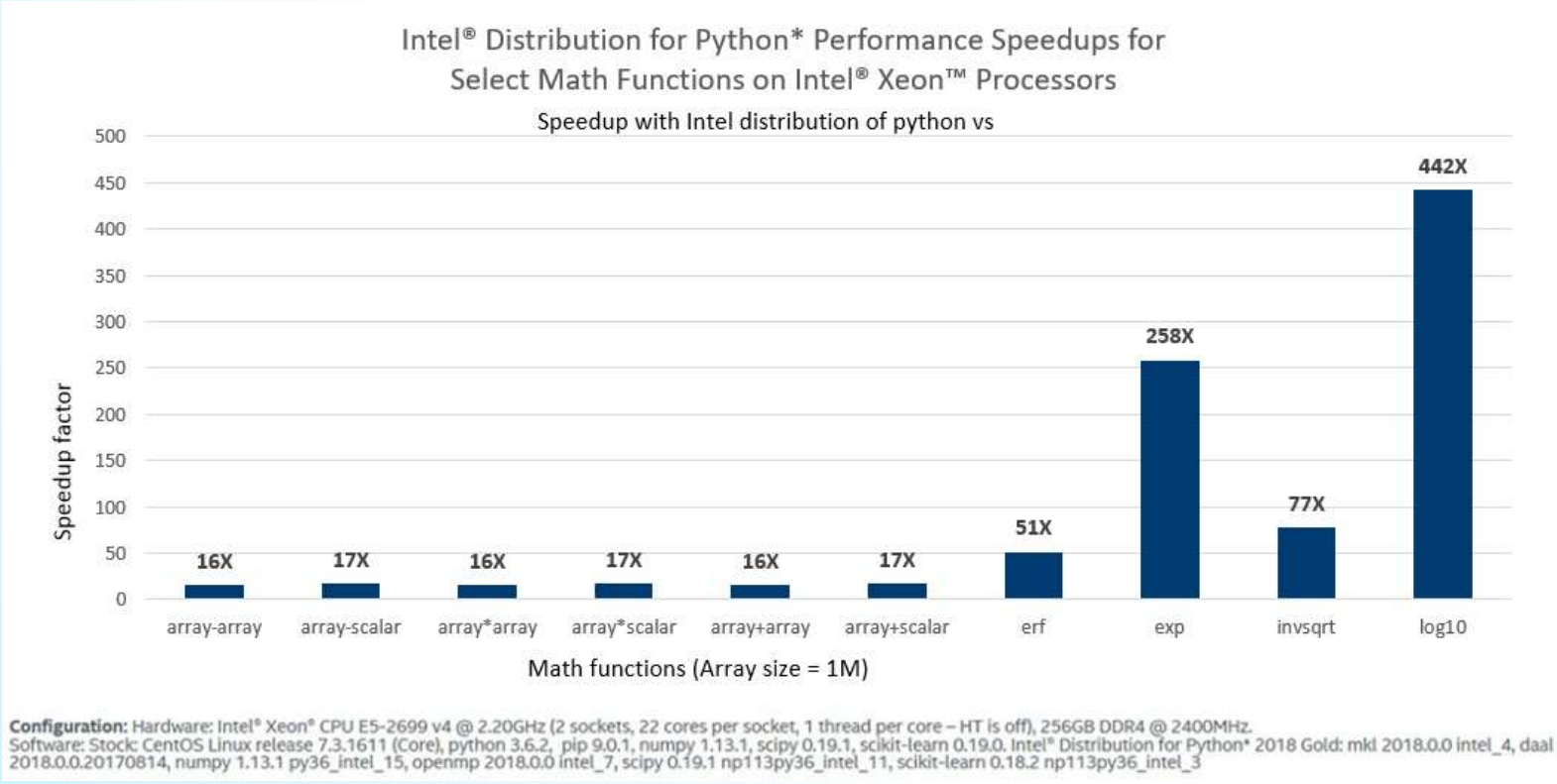
- Features highly tuned functions for DL, classical Machine Learning (ML), and data analytics performance across spectrum of Intel® architecture devices.
- Includes Python*, C++, and Java* APIs.
- Includes connectors to popular data sources including Spark* and Hadoop® and a range of data formats including CSV and SQL.

Intel® Distribution for Python*

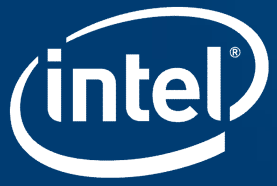
Easy, out-of-the-box access to high performance Python* that is closer to native code speeds.

- Drop-in replacement for your existing Python (no code changes).
- Ready access to set of tools and techniques for high performance on Intel® architecture.
- Accelerated Python packages – NumPy, SciPy, Pandas, Scikit-learn*, Jupyter*, Matplotlib*, and Mpi4py.
- Integrated with Intel® MKL, Intel® DAAL and PyDAAL*, Intel® MPI library, and Intel® TBB.
- Manage packages and Jupyter notebooks easily with Conda*, Anaconda* cloud, and PIP*.

Intel® Distribution for Python*



Configuration: Hardware: Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz (2 sockets, 22 cores per socket, 1 thread per core – HT is off), 256GB DDR4 @ 2400MHz.
Software: Stock: CentOS Linux release 7.3.1611 (Core), python 3.6.2, pip 9.0.1, numpy 1.13.1, scipy 0.19.1, scikit-learn 0.19.0. Intel® Distribution for Python* 2018 Gold: mkl 2018.0.0 intel_4, daal 2018.0.0.20170814, numpy 1.13.1 py36_intel_15, openmp 2018.0.0 intel_7, scipy 0.19.1 np113py36_intel_11, scikit-learn 0.18.2 np113py36_intel_3



TENSORFLOW*

TensorFlow*

TensorFlow*: Python* based DL framework designed for ease of use and extensibility on modern deep neural networks.

- Open-sourced by Google in November 2015.
- As of May 2017, it now integrates optimizations for Intel® Xeon® processors.
- According to technology site KDNuggets*, as of 2017 TensorFlow is the most popular deep learning framework
- Based on defining static computation graphs and then having multidimensional arrays through the graph.

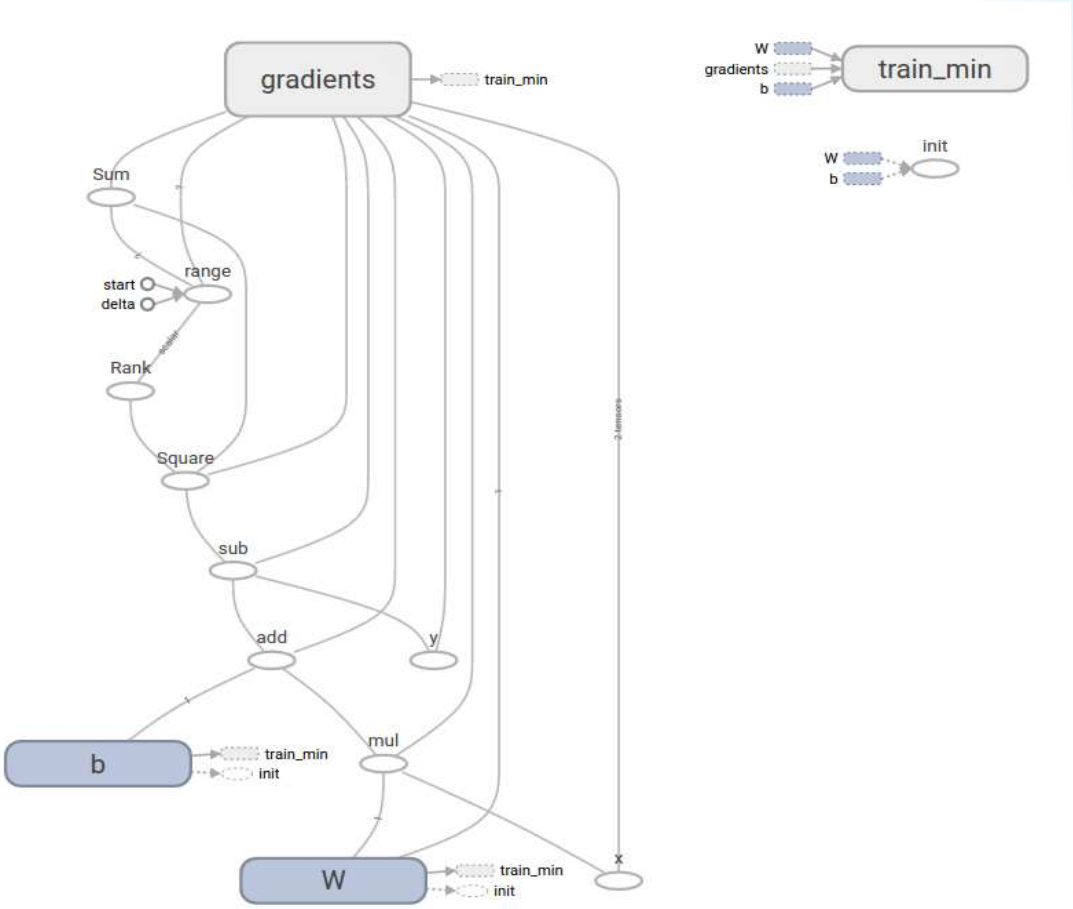
TensorFlow*

TensorFlow* is based on the idea of defining static computational graphs and flowing data through them.

- Edges represent numerical data flowing through the graph.
- Nodes represent computations.
- Data is represented as tensors that are typed multidimensional arrays.
- These can be used to represent words, images, or any other data flowing through the network.
- TensorFlow uses “optimizers”, such as gradient descent, Adam, and RMSProp, to train neural networks.

TensorFlow*

Computational graph for linear regression. Source: TensorFlow*



Initial Performance Gains on Intel® Xeon® Scalable Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-Alexnet	Images / sec	128	33.52	84.2	524	1696	15.6x	20.2x

Optimized performance using TensorFlow with Intel optimizations and built with bazel build --config=mkl --copt="-DEIGEN_USE_VML"

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

Initial Performance Gains on Intel® Xeon® Scalable Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-GoogleNet v1	Images / sec	128	16.87	49.9	112.3	439.7	6.7x	8.8x

Optimized performance using TensorFlow with Intel optimizations and built with bazel build --config=mkl --copt="-DEIGEN_USE_VML"

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

Initial Performance Gains on Intel® Xeon® Scalable Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-VGG	Images / sec	64	8.2	30.7	47.1	151.1	5.7x	4.9x

Optimized performance using TensorFlow with Intel optimizations and built with bazel build --config=mkl --copt="-DEIGEN_USE_VML"

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

Initial Performance Gains on Intel® Xeon Phi™ Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-Alexnet	Images / sec	128	12.21	31.3	549	2698.3	45x	86.2x

Optimized performance using TensorFlow with Intel optimizations and built with `bazel build --config=mkl --copt="-DEIGEN_USE_VML"`

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

Initial Performance Gains on Intel® Xeon Phi™ Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-GoogleNet v1	Images / sec	128	5.43	10.9	106	576.6	19.5x	53x

Optimized performance using TensorFlow with Intel optimizations and built with bazel build --config=mkl --copt="-DEIGEN_USE_VML"

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

Initial Performance Gains on Intel® Xeon Phi™ Processor

Baseline using TensorFlow* 1.0 release with standard compiler knobs.

Benchmark	Metric	Batch Size	Baseline Performance Training	Baseline Performance Inference	Optimized Performance Training	Optimized Performance Inference	Speedup Training	Speedup Inference
ConvNet-VGG	Images / sec	64	1.59	24.6	69.4	251	43.6x	10.2x

Optimized performance using TensorFlow with Intel optimizations and built with bazel build --config=mkl --copt="-DEIGEN_USE_VML"

Configuration Details: Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2. AlexNet, GoogleNet v1 and VGG benchmarks: https://github.com/soumith/convnet-benchmarks

TensorFlow* 501 Course

Intel® AI student kits includes a TensorFlow* 501 course.

Machine Learning 501

Get an overview of the fundamentals of machine learning on modern Intel® architecture. (12 weeks)

Get Started

Previously named Machine Learning 101

Deep Learning 501

Learn the basic techniques and foundations of deep learning on modern Intel® architecture. (12 weeks)

Get Started

Previously named Deep Learning 101

TensorFlow* 501

Master the basics of using TensorFlow* with Intel® architecture. (8 weeks)

Get Started

Previously named TensorFlow 101


intel
OTHER FRAMEWORKS

Caffe*

Caffe* is a deep learning framework written in C++.

- Developed at the University of California Berkeley in 2014.
- Designed to run fast on GPUs, the most popular DL solution at that time.
- The Intel® optimized branch of Caffe is one of the most popular frameworks for image recognition, with improved performance on Intel® Xeon® and Intel® Xeon Phi™ processors.

Caffe2*

Caffe2* is an open source DL framework announced by Facebook in April 2017 with new features.

- Deployed at Facebook to help researchers train large machine learning models and deliver AI on mobile devices.
- Built with expression, speed, and modularity in mind.
- Integrating Intel® Math Kernel Library functions into Caffe2 for optimal inference performance on CPUs.

Theano*

Theano* is a Python* numerical computation library.

- Developed at the University of Montreal.
- Designed to run fast on GPUs.
- Improved performance on Intel® Xeon® and Intel® Xeon Phi™ processors with this fork of the popular Python library.
- Announced in September 2017 that major development will stop.

MXNet*

MXNet* is an open-source, deep learning framework.

- Apache* project originally developed within Carnegie Mellon University.
- Includes built-in support for the Intel® Math Kernel Library (Intel® MKL).
- Now includes Intel, Microsoft, and multiple universities as contributors.
- Characterized by nearly linear scaling with the number of GPUs used (doubling GPUs leads to nearly double the speed).
- Supports both imperative and symbolic programming.

neon™ Framework

neon™ is a deep learning solution optimized for Intel® architecture.

- Nervana™ Systems, founded in 2014, open-sourced neon™ in 2015.
- Nervana Systems joined Intel in August 2016.
- Posted top benchmarks on both GPU and CPU architectures.
- Used assembly level optimizations to produce faster speeds.


intel
BIG DATA FRAMEWORKS

Big Data

By 2020 the Internet of Things (IoT) will include:

- A projected 200 billion smart-and-connected devices.
- The data produced is expected to double every two years to total 40 zettabytes (40 trillion gigabytes).
- New frameworks are needed to process this data

Apache Spark*

Apache Spark* is the most widely used framework for big data analytics.

- Open source big data cluster processing framework.
- Built around speed, ease of use, and advanced analytics.
- Big ecosystem that includes Spark Core*, SQL, Streaming, Mlib, and GraphX*.
- Integrates with cluster management and distributed storage.
- In-memory processing instead of saving data to hard disks.
- No major DL capabilities for Apache Spark.

BigDL

BigDL is a distributed deep learning library for Apache Spark*.

- Developed internally at Intel® and open-sourced December 2016.
- Efficiently scale-out to “Big Data Scale” using Apache Spark.
- High performance powered by Intel® MKL and multi-threaded programming, and implemented as a standalone library on Spark.
- Run DL applications as standard Spark™ programs, that can directly run on top of existing Spark or Hadoop* clusters.
- Feature parity with popular DL frameworks, such as Caffe* and TensorFlow*.
- Load pre-trained Caffe or Torch* models into Spark programs using BigDL.

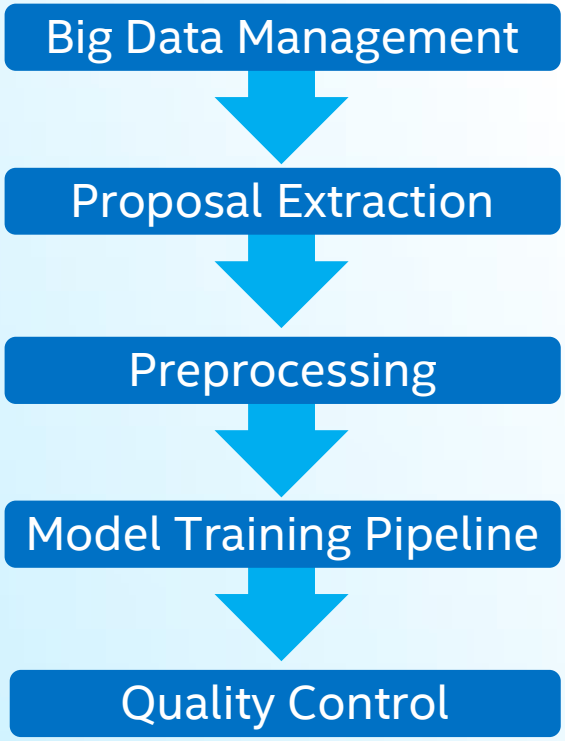
BigDL: A Case Study

Intel worked with one of the largest steel producers to improve their end-to-end manufacturing pipeline.

- Goal: steel surface defect recognition for steel production and quality control from images.
- High resolution images were collected with cameras mounted on assembly line.
- Two part problem:

1. Detect the defect in the image.
2. Classify the defect type.

BigDL: Steel Defects Application



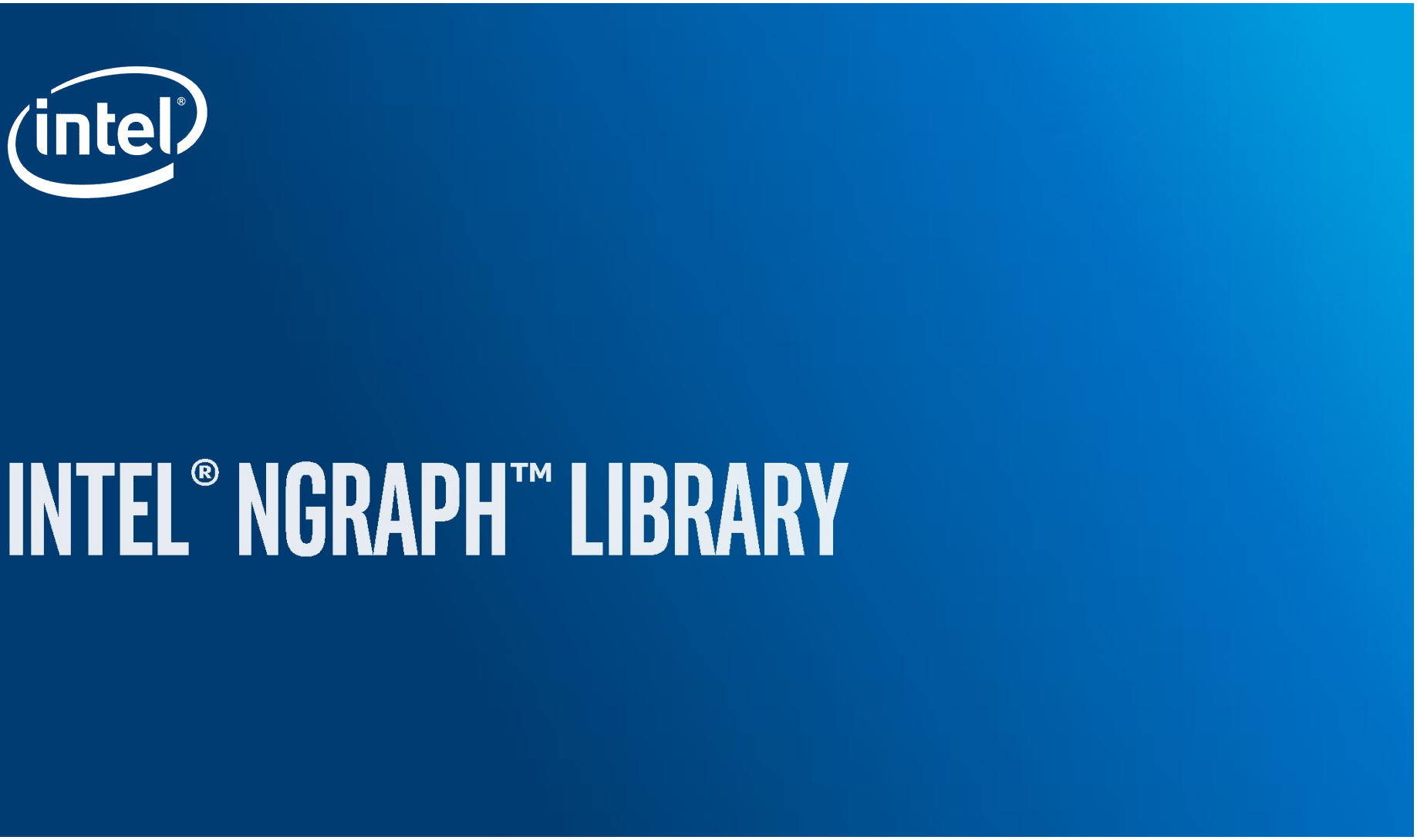
Large volume of high resolution images
of common defects and classifications

Extract proposals for each defect
KeystoneML* framework running on Apache Spark*

Preprocess raw images for model training
Convert to grayscale, resize, normalizing pixels

Train and predict CNN on Apache Spark
Tune parameters and proposal extraction algorithms

Effective BigDL management of system


INTEL® NGRAPH™ LIBRARY

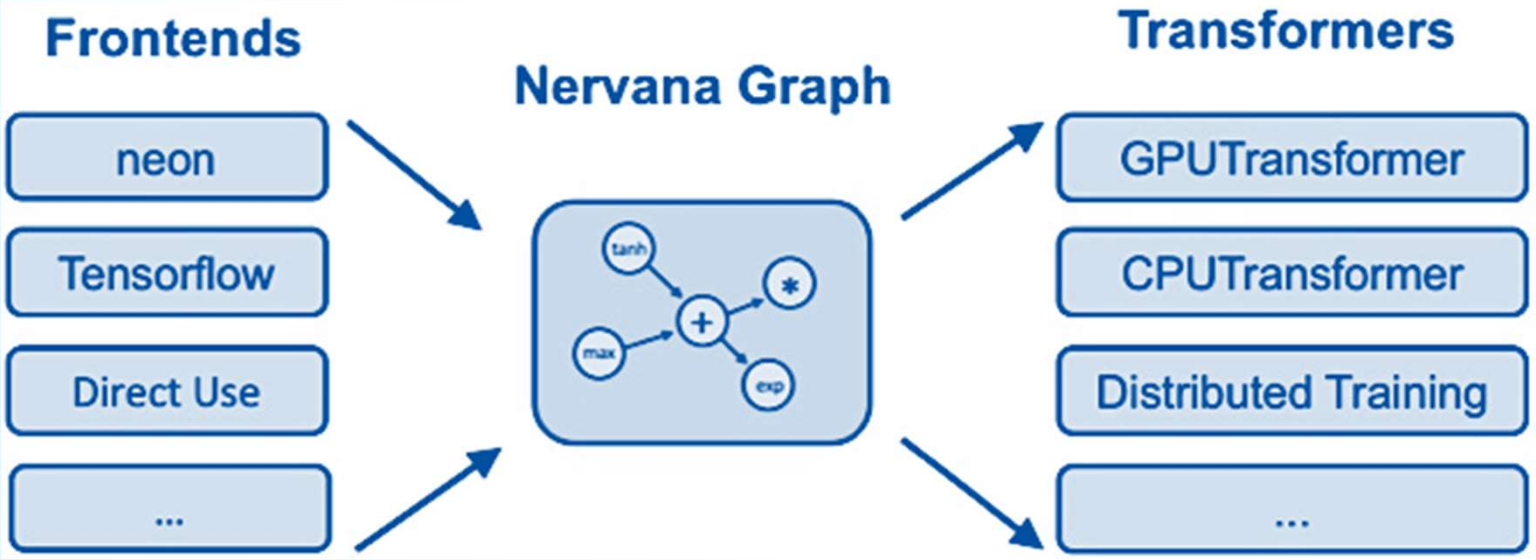
Intel® nGraph™ Library: Motivation

As deep learning advances the modern data scientist needs:

- The flexibility to choose the right front-end for their specific problem.
- To use existing frameworks.
- To write new frameworks.
- Mix and match models across different frameworks.
- Use new hardware by plugging into an existing system, without writing new libraries.
- Efficiently execute across a wide variety of hardware.

Intel® nGraph™ Library: Motivation

nGraph™ is Nervana's library for developing frameworks that can efficiently run DL computations on a variety of compute platforms.



Intel® nGraph™ Library

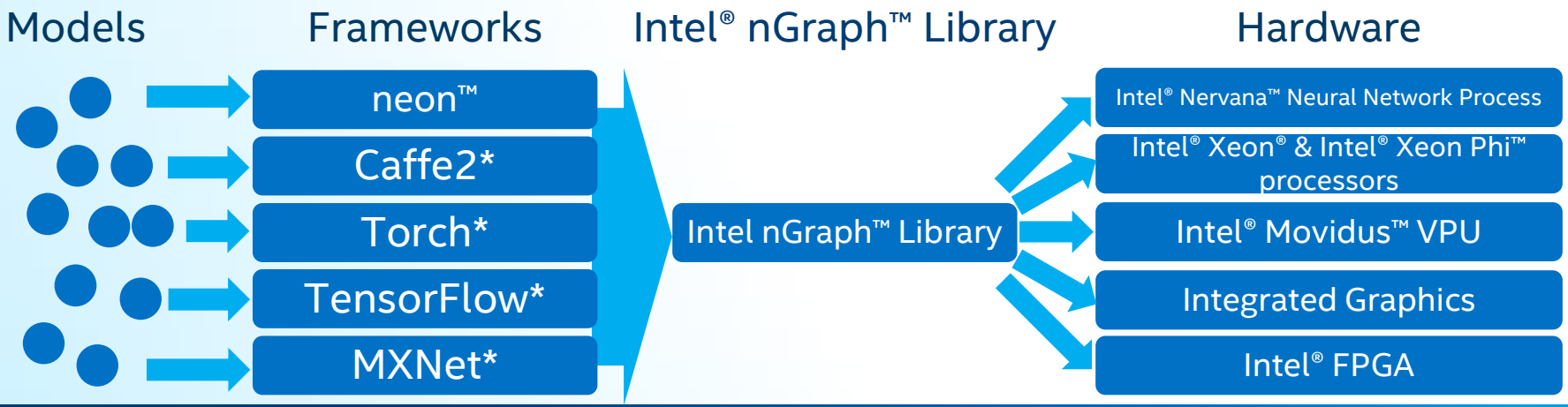
It consists of three primary API components:

- An API for creating computational graphs.
- Two higher-level front-end APIs utilizing the graph API for common DL workflows.
- A transformer API for compiling these graphs and executing them.

Intel® nGraph™ Library

Creates an intermediate representation (IR) for a model that allows for execution across current and future platforms.

- Offers connectors to popular frameworks and back-ends for compiling and executing this IR on IA, GPUs and Intel® hardware.

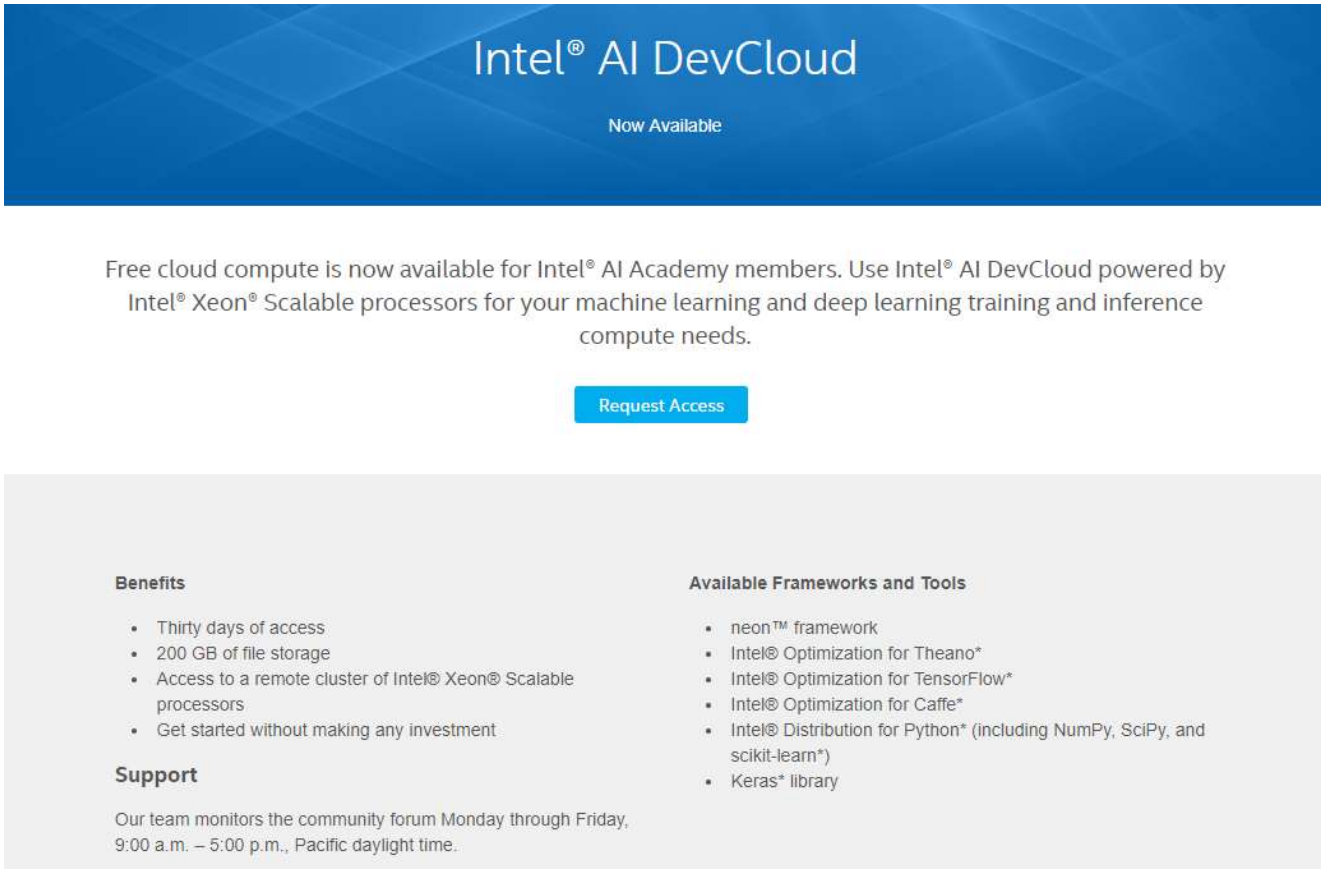



ADDITIONAL INTEL® RESOURCES

Intel® AI DevCloud Overview

Intel AI DevCloud™ is a server cluster consisting of Intel® Xeon® scalable processors.

- Storage, compute needed for DL.
- Pre-loaded with several Intel® optimized DL packages:
- neon™, Theano*, TensorFlow*, Caffe*, Keras*.
- Intel® Distribution for Python*.



https://software.intel.com/en-us/ai-academy/tools/devcloud

Other Resources: AI Student Kits

Intel® AI student kits includes courses on machine learning and deep learning.

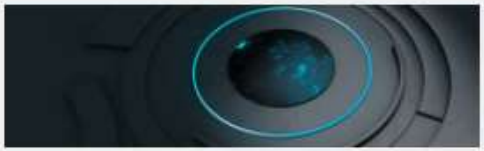
- All three contain concrete examples of implementing AI techniques.

Machine Learning 501

Get an overview of the fundamentals of machine learning on modern Intel® architecture. (12 weeks)

Get Started

Previously named Machine Learning 101



Deep Learning 501

Learn the basic techniques and foundations of deep learning on modern Intel® architecture. (12 weeks)

Get Started

Previously named Deep Learning 101

TensorFlow* 501

Master the basics of using TensorFlow* with Intel® architecture. (8 weeks)

Get Started

Previously named TensorFlow 101


intel
HOMEWORK EXERCISE

Get Access to Hardware

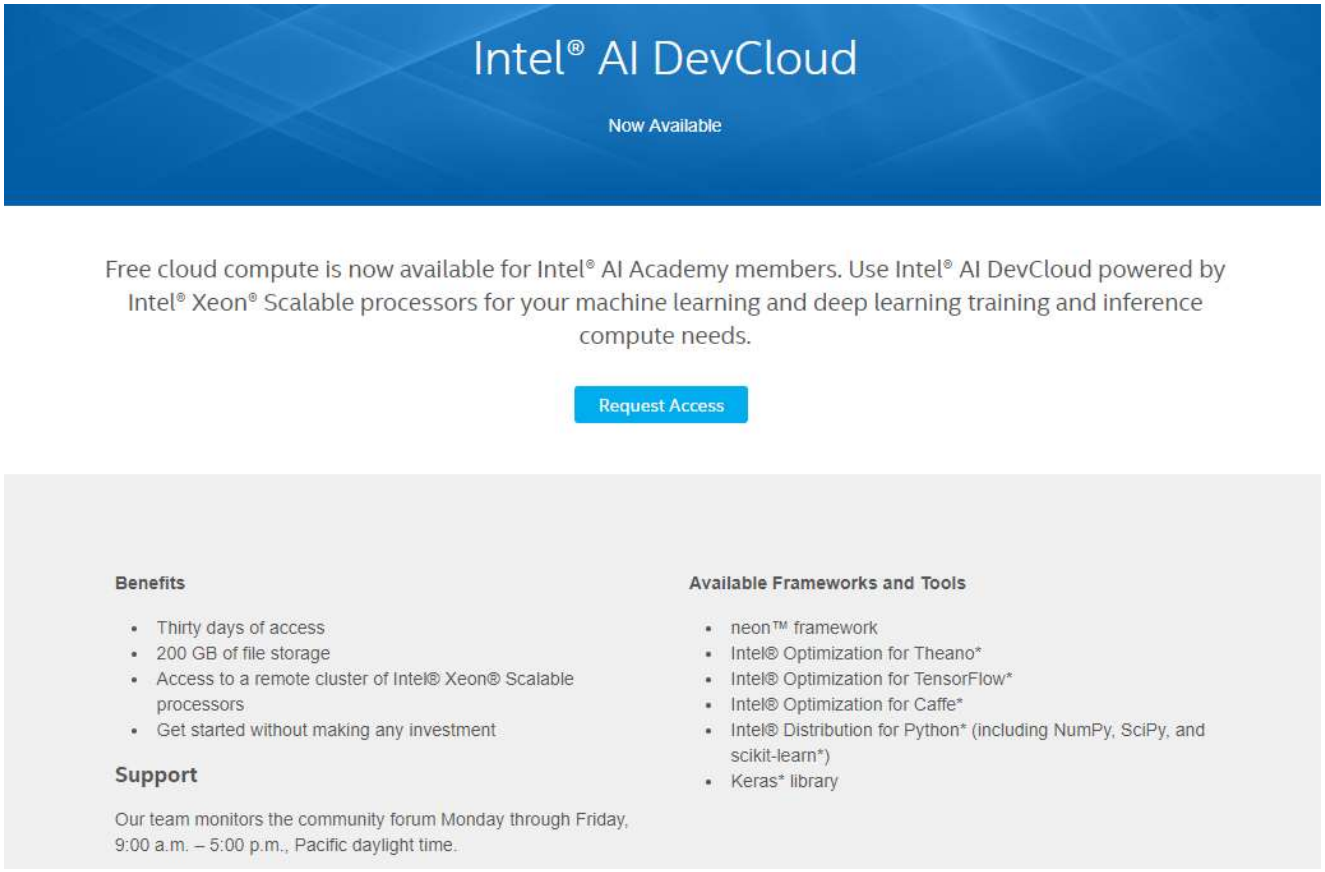
Sign up for Intel® AI DevCloud access

- Understand the tools & libraries that are available to use via remote access
- Sign up & receive access to start work on the Jupyter* notebooks from the Artificial Intelligence 501, Machine Learning 501, Deep Learning 501 or TensorFlow 501 student kits.

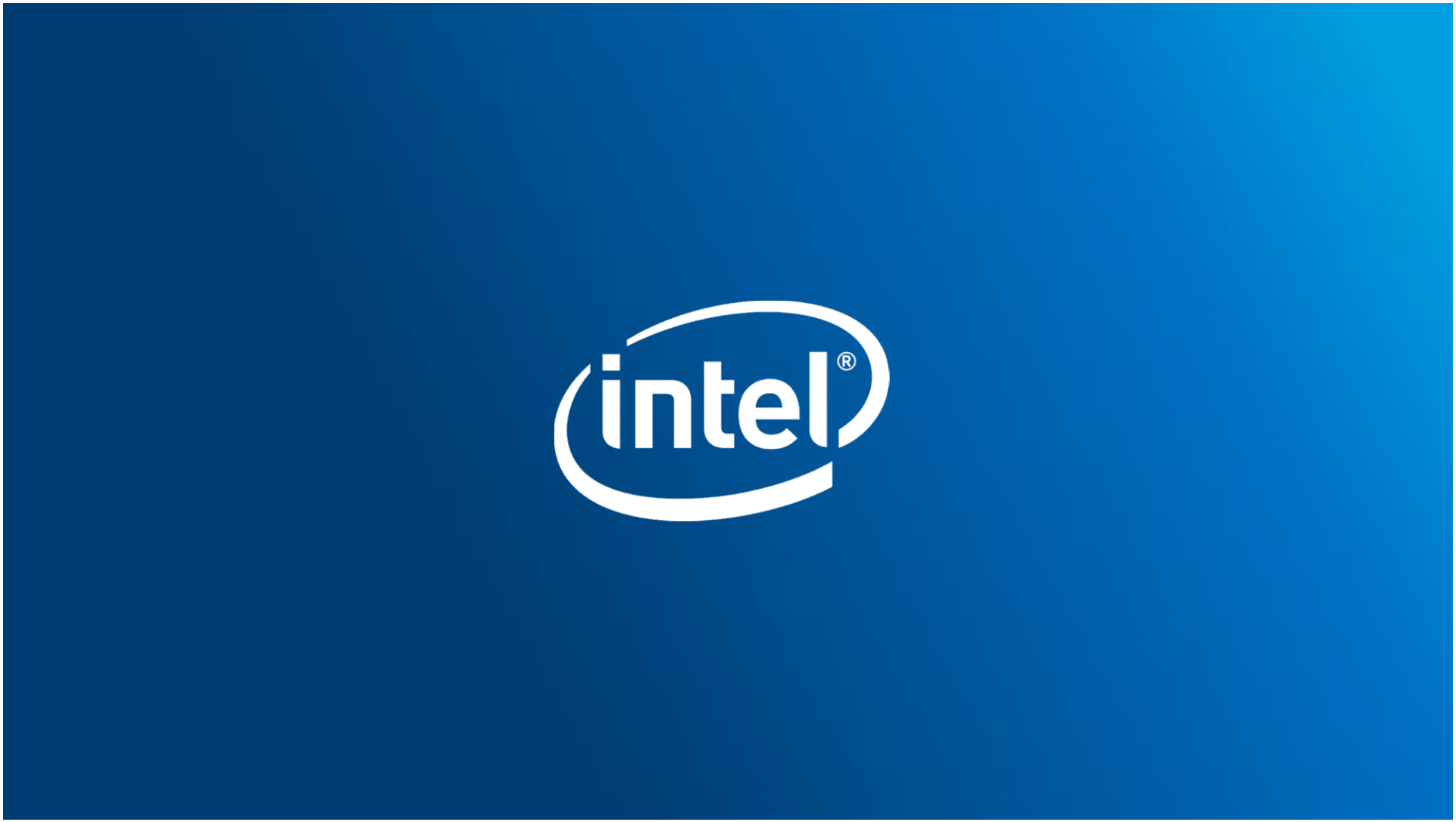
Intel® AI DevCloud Overview

Intel® AI DevCloud is a server cluster consisting of Intel® Xeon® scalable processors.

- Storage, compute needed for DL.
- Pre-loaded with several Intel® optimized DL packages:
- neon™, Theano*, TensorFlow*, Caffe*, Keras*.
- Intel® Distribution for Python*.



https://software.intel.com/en-us/ai-academy/tools/devcloud


intel®

CONFIGURATION DETAILS (CONT'D)

2S Intel® Xeon® processor E5-2697A v4 on Apache Spark™ with MKL2017 up to 18x performance increase compared to 2S E5-2697 v2 + F2JBLAS machine learning training
BASELINE: Intel® Xeon® Processor E5-2697 v2 (12 Cores, 2.7 GHz), 256GB memory, CentOS 6.6*, F2JBLAS: https://github.com/fommil/netlib-java, Relative performance 1.0

Intel® Xeon® processor E5-2697 v2 Apache* Spark* Cluster: 1-Master + 8-Workers, 10Gbit/sec Ethernet fabric, Each system with 2 Processors, Intel® Xeon® processor E5-2697 v2 (12 Cores, 2.7 GHz), Hyper-Threading Enabled, 256GB RAM per System, 1-240GB SSD OS Drive, 12-3TB HDDs Data Drives Per System, CentOS* 6.6, Linux* 2.6.32-642.1.1.el6.x86_64, Intel® Intel® MKL 2017 build U1_20160808 , Cloudera Distribution for Hadoop (CDH) 5.7, Apache* Spark* 1.6.1 standalone, OMP_NUM_THREADS=1 set in CDH*, Total Java Heap Size of 200GB for Spark* Master and Workers, Relative performance up to 3.4x

Intel® Xeon® processor E5-2699 v3 Apache* Spark* Cluster: 1-Master + 8-Workers, 10Gbit/sec Ethernet fabric, Each system with 2 Processors, Intel® Xeon® processor E5-2699 v3 (18 Cores, 2.3 GHz), Hyper-Threading Enabled, 256GB RAM per System, 1-480GB SSD OS Drive, 12-4TB HDDs Data Drives Per System, CentOS* 7.0, Linux 3.10.0-229.el7.x86_64, Intel® Intel® MKL 2017 build U1_20160808 , Cloudera Distribution for Hadoop (CDH) 5.7, Apache* Spark* 1.6.1 standalone, OMP_NUM_THREADS=1 set in CDH*, Total Java Heap Size of 200GB for Spark* Master and Workers, Relative performance up to 8.8x

Intel® Xeon® processor E5-2697A v4 Apache* Spark* Cluster: 1-Master + 8-Workers, 10Gbit Ethernet/sec fabric, Each system with 2 Processors, Intel® Xeon® processor E5-2697A v4 (16 Cores, 2.6 GHz), Hyper-Threading Enabled, 256GB RAM per System, 1-800GB SSD OS Drive, 10-240GB SSDs Data Drives Per System, CentOS* 6.7, Linux 2.6.32-573.12.1.el6.x86_64, Intel® MKL 2017 build U1_20160808 , Cloudera Distribution for Hadoop (CDH) 5.7, Apache* Spark* 1.6.1 standalone, OMP_NUM_THREADS=1 set in CDH*, Total Java Heap Size of 200GB for Spark* Master and Workers, Relative performance up to 18x

Machine learning algorithm used for all configurations : Alternating Least Squares ALS Machine Learning Algorithm https://github.com/databricks/spark-perf

Intel® Xeon Phi™ Processor 7250 GoogleNet V1 Time-To-Train Scaling Efficiency up to 97% on 32 nodes
32 nodes of Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: flat mode), 96GB DDR4 memory, Red Hat* Enterprise Linux 6.7, export OMP_NUM_THREADS=64 (the remaining 4 cores are used for driving communication) MKL 2017 Update 1, MPI: 2017.1.132, Endeavor KNL bin1 nodes, export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2, Throughput is measured using "train" command. Data pre-partitioned across all nodes in the cluster before training. There is no data transferred over the fabric while training. Scaling efficiency computed as: (Single node performance / (N * Performance measured with N nodes))*100, where N = Number of nodes

Intel® optimized Caffe*: Intel internal version of Caffe

GoogLeNetV1: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf, batch size 1536

Intel® Xeon Phi ™ processor 7250 up to 400x performance increase with Intel Optimized Frameworks compared to baseline out of box performance
BASELINE: Caffe Out Of the Box, Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: cache mode), 96GB memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2, BVLC-Caffe: https://github.com/BVLC/caffe, with OpenBLAS, Relative performance 1.0

NEW: Caffe: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: cache mode), 96GB memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2, Intel® Caffe: : https://github.com/intel/caffe based on BVLC Caffe as of Jul 16, 2016, MKL GOLD UPDATE1, Relative performance up to 400x

AlexNet used for both configuration as per https://papers.nips.cc/paper/4824-Large image database-classification-with-deep-convolutional-neural-networks.pdf, Batch Size: 256

Intel® Xeon Phi™ Processor 7250, 32 node cluster with Intel® Omni Path Fabric up to 97% GoogleNetV1 Time-To-Train Scaling Efficiency
Caffe: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: flat mode), 96GB DDR4 memory, Red Hat* Enterprise Linux 6.7, Intel® Caffe: https://github.com/intel/caffe, not publically available yet

export OMP_NUM_THREADS=64 (the remaining 4 cores are used for driving communication)
MKL 2017 Update 1, MPI: 2017.1.132, Endeavor KNL bin1 nodes, export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2, Throughput is measured using "train" command. Split the images across nodes and copied locally on each node at the beginning of training. No IO happens over fabric while training.

GoogLeNetV1: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf, batch size 1536

Intel® Xeon Phi ™ processor Knights Mill up to 4x estimated performance improvement over Intel® Xeon Phi™ processor 7290
BASELINE: Intel® Xeon Phi™ Processor 7290 (16GB, 1.50 GHz, 72 core) with 192 GB Total Memory on Red Hat Enterprise Linux* 6.7 kernel 2.6.32-573 using MKL 11.3 Update 4, Relative performance 1.0

NEW: Intel® Xeon phi™ processor family – Knights Mill, Relative performance up to 4x

Intel® Arria 10 – 1150 FPGA energy efficiency on Caffe/AlexNet up to 25 img/s/w with FP16 at 297MHz
Vanilla AlexNet Classification Implementation as specified by http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf, Training Parameters taken from Caffe open-source Framework are 224x224x3 Input, 1000x1 Output, FP16 with Shared Block-Exponents, All compute layers (incl. Fully Connected) done on the FPGA except for Softmax, Arria 10-1150 FPGA, -1 Speed Grade on Altera PCIe DevKit with x72 DDR4 @ 1333 MHz, Power measured through on-board power monitor (FPGA POWER ONLY), ACDS 16.1 Internal Builds + OpenCL SDK 16.1 Internal Build, Compute machine is an HP Z620 Workstation, Xeon E5-1660 at 3.3 GHz with 32GB RAM. The Xeon is not used for compute.

Knights Mill performance : Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Optimization Notice: Intel's 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. Notice Revision #20110804
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: http://www.intel.com/performance Source: Intel measured everything except Knights Mill which is estimated as of November 2016

CONFIGURATION DETAILS (CONT'D)

Optimization Notice: Intel's 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. Notice Revision #20110804.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

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 http://www.intel.com/performance/datacenter. Tested by Intel as of 14 June 2016. Configurations:

Faster and more scalable than GPU claim based on Intel analysis and testing

Up to 2.3x faster training per system claim based on AlexNet* topology workload (batch size = 1024) using a large image database running 4-nodes Intel Xeon Phi processor 7250 (16 GB MCDRAM, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, 96GB DDR4-2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat Enterprise Linux* 6.7 (Santiago), 1.0 TB SATA drive WD1003FZEX-00MK2A0 System Disk, running Intel® Optimized DNN Framework (internal development version) training 1.33 million images in 10.5 hours compared to 1-node host with four NVIDIA "Maxwell" GPUs training 1.33 million images in 25 hours (source: http://www.slideshare.net/NVIDIA/gtc-2016-opening-keynote slide 32).

Up to 38% better scaling efficiency at 32-nodes claim based on GoogLeNet deep learning image classification training topology using a large image database comparing one node Intel Xeon Phi processor 7250 (16 GB MCDRAM, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, DDR4 96GB DDR4-2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat* Enterprise Linux 6.7, Intel® Optimized DNN Framework with 87% efficiency to unknown hosts running 32 each NVIDIA Tesla* K20 GPUs with a 62% efficiency (Source: http://arxiv.org/pdf/1511.00175v2.pdf showing FireCaffe* with 32 NVIDIA Tesla* K20s (Titan Supercomputer*) running GoogLeNet* at 20x speedup over Caffe* with 1 K20).

Up to 6 SP TFLOPS based on the Intel Xeon Phi processor peak theoretical single-precision performance is preliminary and based on current expectations of cores, clock frequency and floating point operations per cycle. FLOPS = cores x clock frequency x floating-point operations per second per cycle

Up to 3x faster single-threaded performance claim based on Intel estimates of Intel Xeon Phi processor 7290 vs. coprocessor 7120 running XYZ workload.

Up to 2.3x faster training per system claim based on AlexNet* topology workload (batch size = 1024) using a large image database running 4-nodes Intel Xeon Phi processor 7250 (16 GB MCDRAM, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, 96GB DDR4-2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat Enterprise Linux* 6.7 (Santiago), 1.0 TB SATA drive WD1003FZEX-00MK2A0 System Disk, running Intel® Optimized DNN Framework, Intel® Optimized Caffe (internal development version) training 1.33 billion images in 10.5 hours compared to 1-node host with four NVIDIA "Maxwell" GPUs training 1.33 billion images in 25 hours (source: http://www.slideshare.net/NVIDIA/gtc-2016-opening-keynote slide 32).

Up to 38% better scaling efficiency at 32-nodes claim based on GoogLeNet deep learning image classification training topology using a large image database comparing one node Intel Xeon Phi processor 7250 (16 GB MCDRAM, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, DDR4 96GB DDR4-2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat* Enterprise Linux 6.7, Intel® Optimized DNN Framework with 87% efficiency to unknown hosts running 32 each NVIDIA Tesla* K20 GPUs with a 62% efficiency (Source: http://arxiv.org/pdf/1511.00175v2.pdf showing FireCaffe* with 32 NVIDIA Tesla* K20s (Titan Supercomputer*) running GoogLeNet* at 20x speedup over Caffe* with 1 K20).

Up to 50x faster training on 128-node as compared to single-node based on AlexNet* topology workload (batch size = 1024) training time using a large image database running one node Intel Xeon Phi processor 7250 (16 GB MCDRAM, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, 96GB DDR4-2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat Enterprise Linux* 6.7 (Santiago), 1.0 TB SATA drive WD1003FZEX-00MK2A0 System Disk, running Intel® Optimized DNN Framework, training in 39.17 hours compared to 128-node identically configured with Intel® Omni-Path Host Fabric Interface Adapter 100 Series 1 Port PCIe x16 connectors training in 0.75 hours. Contact your Intel representative for more information on how to obtain the binary. For information on workload, see https://papers.nips.cc/paper/4824-Large image database-classification-with-deep-convolutional-neural-networks.pdf.

Up to 30x software optimization improvement claim based on customer CNN training workload running 2S Intel® Xeon® processor E5-2680 v3 running Berkeley Vision and Learning Center* (BVLC) Caffe + OpenBlas* library and then run tuned on the Intel® Optimized Caffe (internal development version) + Intel® Math Kernel Library (Intel® MKL).

CONFIGURATION DETAILS (CONT'D)

Platform: 2S Intel® Xeon® Platinum 8180 processor @ 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

Deep Learning Frameworks:

- 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, 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 (GoogLeNet, AlexNet, and ResNet-50), https://github.com/intel/caffe/tree/master/models/default_vgg_19 (VGG-19), and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

- TensorFlow*: (https://github.com/tensorflow/tensorflow), commit id 207203253b6f8ea5e938a512798429f91d5b4e7e. Performance numbers were obtained for three convnet benchmarks: alexnet, googlenetv1, vgg(https://github.com/soumith/convnet-benchmarks/tree/master/tensorflow) using dummy data. GCC 4.8.5, Intel MKL small libraries version 2018.0.20170425, interop parallelism threads set to 1 for alexnet, vgg benchmarks, 2 for googlenet benchmarks, intra op parallelism threads set to 56, data format used is NCHW, KMP_BLOCKTIME set to 1 for googlenet and vgg benchmarks, 30 for the alexnet benchmark. Inference measured with --caffe time -forward_only -engine MKL2017option, training measured with --forward_backward_only option.

- MxNet: (https://github.com/dmlc/mxnet/), revision 5efd91a71f36fea483e882b0358c8d46b5a7aa20. Dummy data was used. Inference was measured with "benchmark_score.py", training was measured with a modified version of benchmark_score.py which also runs backward propagation. Topology specs from https://github.com/dmlc/mxnet/tree/master/example/image-classification/symbols. GCC 4.8.5, Intel MKL small libraries version 2018.0.20170425.

- Neon: ZP/MKL_CHWN branch commit id:52bd02acb947a2adabb8a227166a7da5d9123b6d. Dummy data was used. The main.py script was used for benchmarking , in mkl mode. ICC version used : 17.0.3 20170404, Intel MKL small libraries version 2018.0.20170425.

CONFIGURATION DETAILS (CONT'D)

Platform: 2S Intel® Xeon® processor E5-2697 v2 @ 2.70GHz (12 cores), HT enabled, turbo enabled, scaling governor set to "performance" via intel_pstate driver, 256GB DDR3-1600 ECC RAM. CentOS Linux* release 7.3.1611 (Core), Linux kernel 3.10.0-514.21.1.el7.x86_64. SSD: Intel® SSD 520 Series 240GB, 2.5in SATA 6Gb/s, 25nm, MLC.

Performance measured with: Environment variables: KMP_AFFINITY='granularity=fine, compact,1,0', OMP_NUM_THREADS=24, CPU Freq set with cpupower frequency-set -d 2.7G -u 3.5G -g performance

Deep Learning Frameworks:

- Caffe*: (http://github.com/intel/caffe/), revision b0ef3236528a2c7d2988f249d347d5fdae831236. 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 (GoogLeNet, AlexNet, and ResNet-50), https://github.com/intel/caffe/tree/master/models/default_vgg_19 (VGG-19), and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). GCC 4.8.5, Intel MKL small libraries version 2017.0.2.20170110.

CONFIGURATION DETAILS (CONT'D)

Intel® Caffe*: Intel internal version of Caffe

Intel® Arria 10 – 1150 FPGA energy efficiency on Caffe/AlexNet up to 25 img/s/w with FP16 at 297MHz
Vanilla AlexNet Classification Implementation as specified by http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf, Training Parameters taken from Caffe open-source Framework are 224x224x3 Input, 1000x1 Output, FP16 with Shared Block-Exponents, All compute layers (incl. Fully Connected) done on the FPGA except for Softmax, Arria 10-1150 FPGA, -1 Speed Grade on Altera PCIe DevKit with x72 DDR4 @ 1333 MHz, Power measured through on-board power monitor
(FPGA POWER ONLY), ACDS 16.1 Internal Builds + OpenCL SDK 16.1 Internal Build, Compute machine is an HP Z620 Workstation, Intel® Xeon E5-1660 processor at 3.3 GHz with 32GB RAM. The Intel Xeon processor is not used for compute.

Projected data comes from verified deterministic model of DLA IP implementation on FPGA

CONFIGURATION DETAILS (CONT'D)

INFERENCE using FP32 Batch Size Caffe GoogleNet v1 256 AlexNet 256.

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 http://www.intel.com/performance Source: Intel measured as of June 2017 Optimization Notice: Intel's 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.

Configurations 40, 41:

Platform: 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. Deep Learning Frameworks: 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, 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 (GoogLeNet, AlexNet, and ResNet-50), https://github.com/intel/caffe/tree/master/models/default_vgg_19 (VGG-19), and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

Platform: 2S Intel® Xeon® CPU E5-2699 v3 @ 2.30GHz (18 cores), HT enabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 256GB DDR4-2133 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.el7.x86_64. OS drive: Seagate* Enterprise ST2000NX0253 2 TB 2.5" Internal Hard Drive. Performance measured with: Environment variables: KMP_AFFINITY='granularity=fine, compact,1,0', OMP_NUM_THREADS=36, CPU Freq set with cpupower frequency-set -d 2.3G -u 2.3G -g performance. Deep Learning Frameworks: Intel Caffe: (http://github.com/intel/caffe/), revision b0ef3236528a2c7d2988f249d347d5fdae831236. 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 (GoogLeNet, AlexNet, and ResNet-50), https://github.com/intel/caffe/tree/master/models/default_vgg_19 (VGG-19), and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). GCC 4.8.5, MKLML version 2017.0.2.20170110. BVLC-Caffe: https://github.com/BVLC/caffe, Inference & Training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was st ored on local storage and cached in memory before training BVLC Caffe (http://github.com/BVLC/caffe), revision 91b09280f5233cafc62954c98ce8bc4c204e7475 (commit date 5/14/2017). BLAS: atlas ver. 3.10.1.

CONFIGURATION DETAILS (CONT'D)

Intel® and SURFsara* Research Collaboration
MareNostrum4/BSC* Configuration Details

***MareNostrum4/Barcelona Supercomputing Center**: https://www.bsc.es/

Compute Nodes: 2 sockets Intel® Xeon® Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of 48 cores per node, 2 Threads per core, L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K, 96 GB of DDR4, Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, 10 Gbit Ethernet, 200 GB local SSD, Red Hat* Enterprise Linux* 6.7.

Intel optimized Caffe*: Intel® version of Caffe; http://github.com/intel/caffe/, revision 8012927bf2bf70231cbc7ff55de0b1bc11de4a69.
Intel® MKL version: mklml_lnx_2018.0.20170425; Intel® MLSL version: l_mlsl_2017.1.016

Model: Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50) and modified for wide-RedNet-50. Batch size as stated in the performance chart

Time-To-Train: measured using "train" command. Data copied to memory on all nodes in the cluster before training. No input image data transferred over the fabric while training;

Performance measured with:
export OMP_NUM_THREADS=44 (the remaining 4 cores are used for driving communication), export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2

OMP_NUM_THREADS=44 KMP_AFFINITY="proclist=[0-87],granularity=thread,explicit" KMP_HW_SUBSET=1t MLSL_NUM_SERVERS=4 mpiexec.hydra -PSM2 -l -n $SLURM_JOB_NUM_NODES -ppn 1 -f hosts2 -genv OMP_NUM_THREADS 44 -env KMP_AFFINITY "proclist=[0-87],granularity=thread,explicit" -env KMP_HW_SUBSET 1t -genv I_MPI_FABRICS tmi -genv I_MPI_HYDRA_BRANCH_COUNT $SLURM_JOB_NUM_NODES -genv I_MPI_HYDRA_PMI_CONNECT alltoall sh -c 'cat /ilsvrc12_train_lmdb_striped_64/data.mdb > /dev/null ; cat /ilsvrc12_val_lmdb_striped_64/data.mdb > /dev/null ; ulimit -u 8192 ; ulimit -a ; numactl -H ; /caffe/build/tools/caffe train --solver=/caffe/models/intel_optimized_models/multinode/resnet_50_256_nodes_8k_batch/solver_poly_quick_large.prototxt -engine "MKL2017"

SURFsara blog: https://blog.surf.nl/en/imagenet-1k-training-on-intel-xeon-phi-in-less-than-40-minutes/ ; Researchers: Valeriu Codreanu, Ph.D. (PI).; Damian Podareanu, MSc; SURFsara* & Vikram Saletore, Ph.D. (co-PI): Intel Corp.

*SURFsara B.V. is the Dutch national high-performance computing and e-Science support center. Amsterdam Science Park, Amsterdam, The Netherlands.


intel
APPENDIX

Download and Install Frameworks

To download Intel® optimized frameworks, as well as install documentation, you can go to https://software.intel.com/en-us/ai-academy/frameworks.

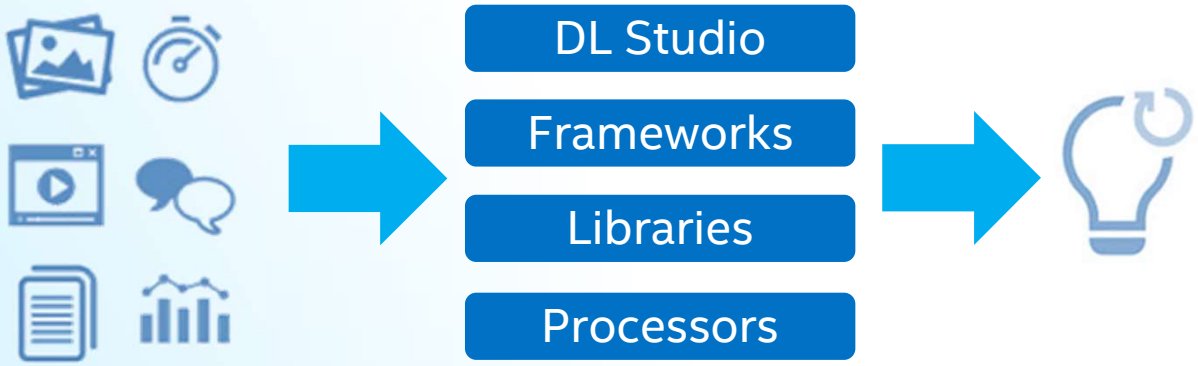
This webpage contains links to GitHub* pages to download the optimized frameworks.

Install Intel® Distribution for Python*

1. Install Anaconda www.continuum.io/downloads
2. Choose Intel Packages: conda config --add channels intel
3. Create the environment: conda create –n intelpython3 intelpython3_full python=3
4. Activate the environment: source activate intelpython3
5. Run jupyter notebook*: jupyter notebook –no browser (only use no browser if running remotely or using BASH on windows)
6. Access the notebook: http://localhost:8888

Intel® Deep Learning Platform

Enterprise-ready AI deployed in the cloud or on premise. A turnkey, full stack & user-friendly system that enables businesses to develop and deploy high-accuracy AI solutions in record time.



Optimizing DL Frameworks for Intel® Architecture

Leverage high performant compute libraries and tools such as Intel® MKL, Intel® optimized Python*, and Intel® Compiler.

- Data format/shape: right format/shape for max performance (blocking and gather/scatter).
- Data layout: minimize cost of data layout conversions.
- Parallelism: use all cores, eliminate serial sections, load imbalance.
- Memory allocation: unique characteristics and ability to reuse buffers.
- Data layer optimizations: parallelization, vectorization, IO.
- Optimize hyper parameters, such as batch size for more parallelism.