High Performance Deep Learning on Intel® Architecture

Ivan Kuzmin

Engineering Manager for AI Performance Libraries

December 19, 2016

Fast Evolution of Technology

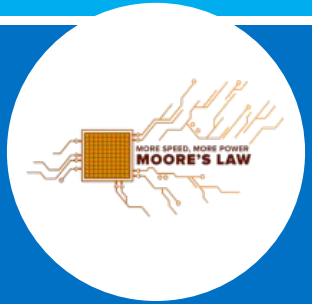
Bigger Data

Image: 1000 KB / picture

Audio: 5000 KB / song

Video: 5,000,000 KB / movie

Better Hardware



Transistor density doubles every 18 months

Cost / GB in 1995: $1000.00

Cost / GB in 2015: $0.03

Smarter Algorithms

Advances in neural networks leading to better accuracy in training models

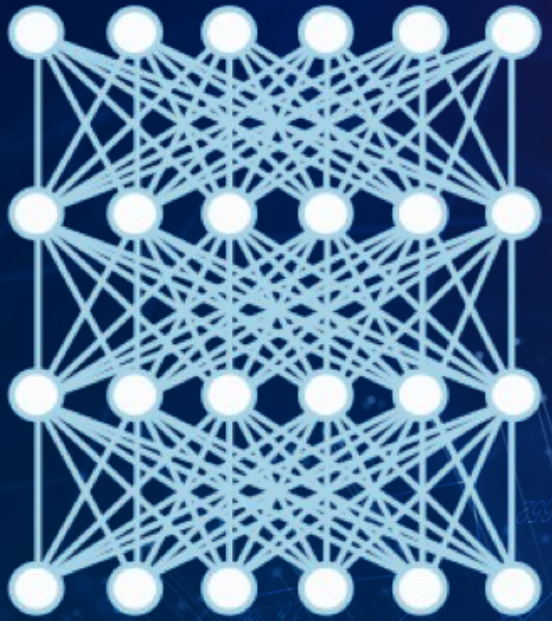
Classical Machine Learning

$(f_1, f_2, \ldots, f_K)$

CLASSIFIER
SVM
Random Forest
Naïve Bayes
Decision Trees
Logistic Regression
Ensemble methods

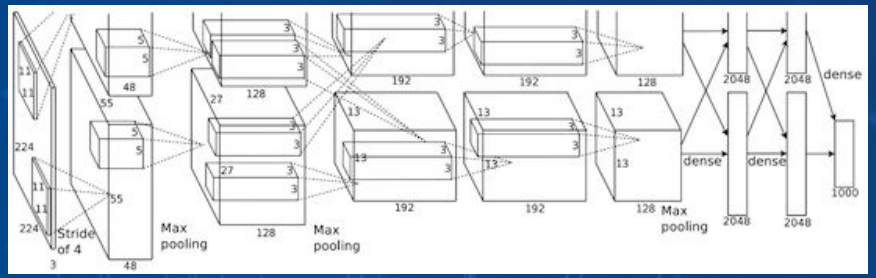
Arjun

Deep learning



- A method of extracting features at multiple levels of abstraction
- Features are discovered from data
- Performance improves with more data
- High degree of representational power

End-to-End Deep Learning

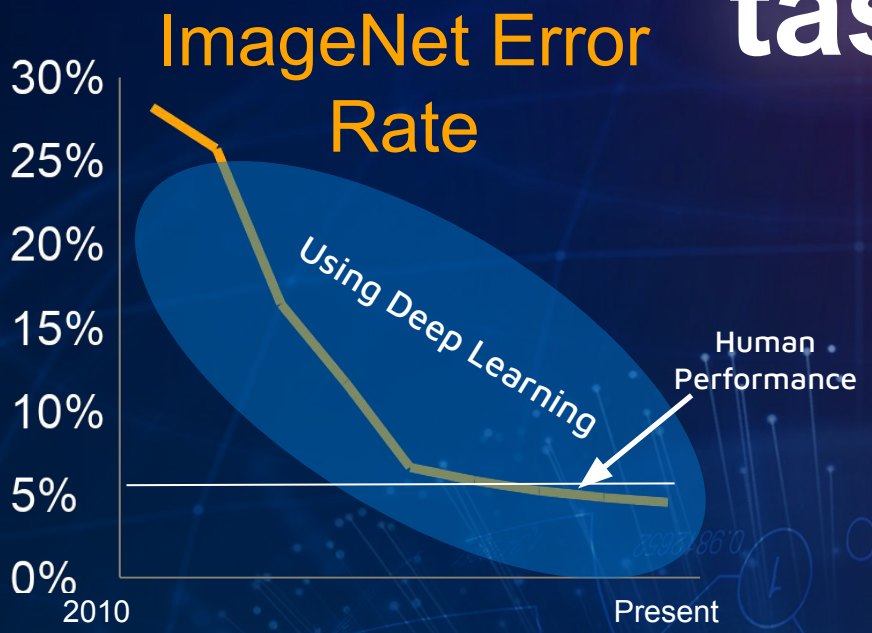


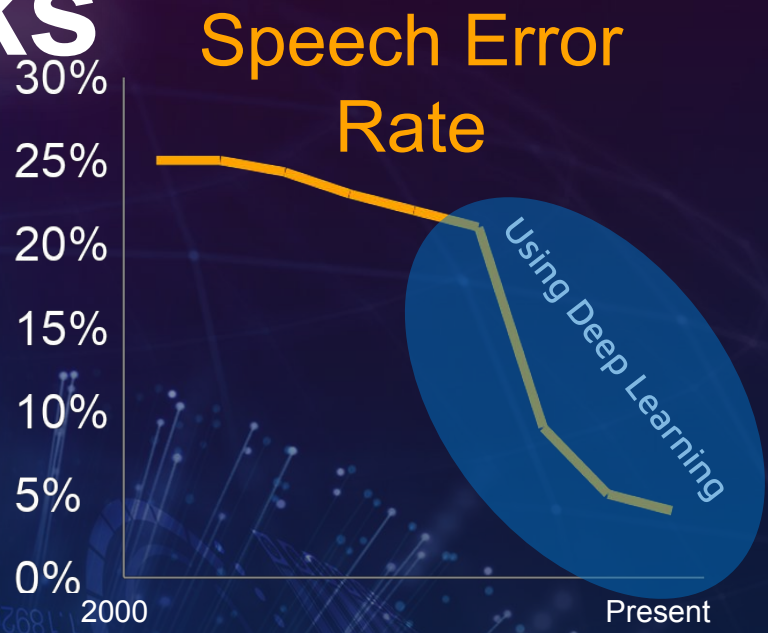
~60 million parameters

Arjun

But old practices apply:
Data Cleaning, Exploration, Data annotation, hyperparameters, etc.

Automating previously “human” tasks





Deep Learning Challenges

(1) Large compute requirements for training

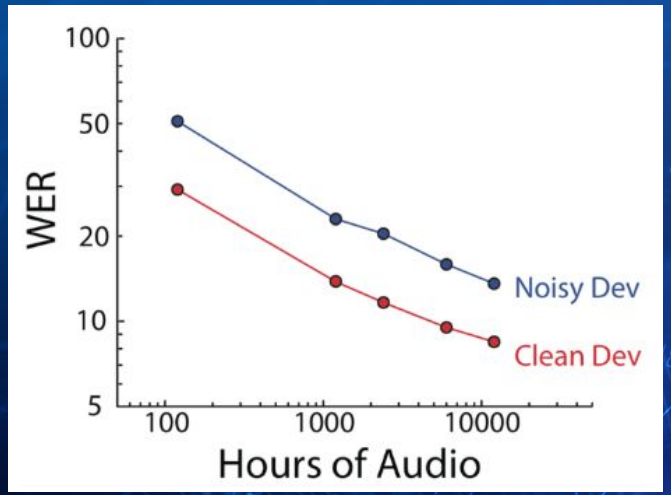
Compute/data
O(exaflops) / O(Terabytes)

Training one model is ~ 10 exaflops

Baidu Research

Deep Learning Challenges

(2) Performance scales with data



Baidu DS2

Google's YouTube-8M

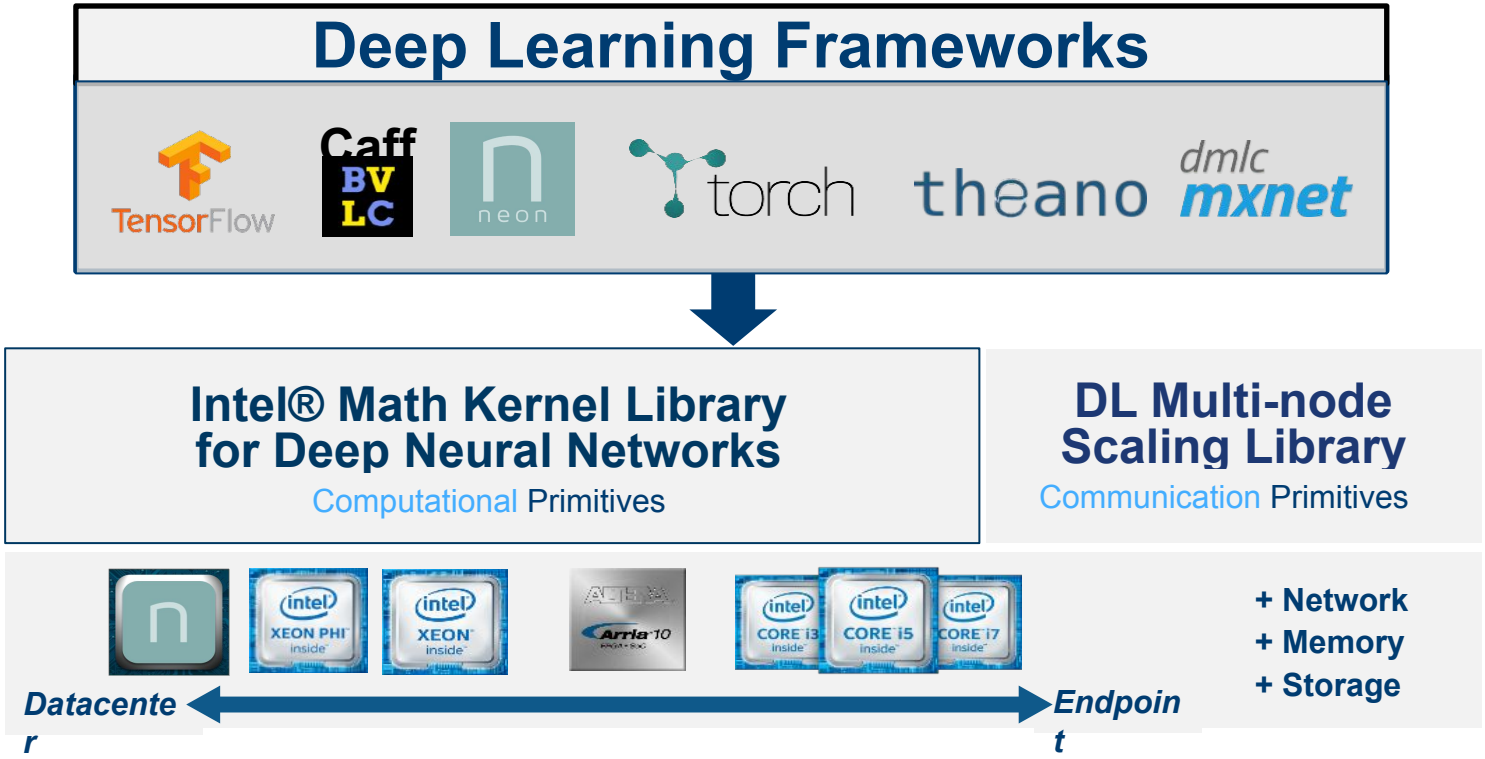
8 Million	0.5 Million	1.9 Billion	4800	1.8
Video URLs	Hours of Video	Frame Features	Classes	Avg. Labels / Video

Spacenet

Imagenet

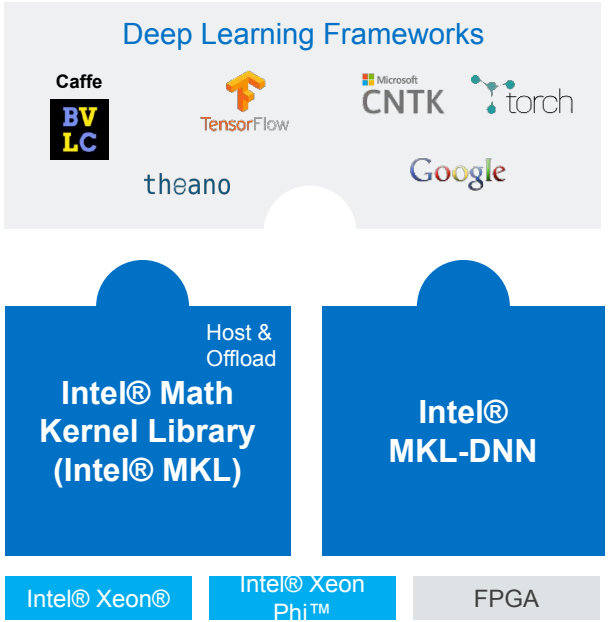
Scaling is I/O Bound

Intel Provides the Compute Foundation for DL



INTEL® MKL-DNN

Deep learning with Intel® MKL-DNN



Variety of popular Deep Learning frameworks

Intel® MKL-DNN

- Faster optimization of frameworks by linking Intel® MKL-DNN directly, C/C++ API
- New algorithms ahead of Intel® MKL productization
- Collaboration across Intel and community to reuse new algorithms, accept contributions
- Vehicle for collaboration with researchers on algorithmic innovation on Intel® CPU
- Open Source Apache 2.0 License

Intel® MKL

- SW building block to extract maximum performance on Intel® CPU
- Provide common interface to all Intel® CPUs.

Intel® MKL-DNN and Intel® MKL – path to bring performance to DL frameworks on Intel® CPU

*Other names and brands may be claimed as property of others.

Deep learning with Intel® MKL-DNN

Intel® MKL-DNN
- Tech preview, https://01.org/mkl-dnn
- Demonstrates interfaces and the library structure for accepting external contributions
- Single precision
- Plan to enable more primitives to support additional topologies
- Plan more optimizations for latest generation of Intel® CPU

Deep Learning primitives in Intel® MKL-DNN	For support of topologies	Optimized for
Direct (3-D) convolution Fully connected Max/Avg/Min Pooling Rectifier Linear Unit (reLu) Local Response Normalization (LRN) 3-D convolution+ reLu	AlexNet, VGG, GoogleNet, ResNet	Intel(R) Xeon(R) processor E5-xxxx v3 (codename Haswell) Intel(R) Xeon(R) processor E5-xxxx v4 (codename Broadwell)

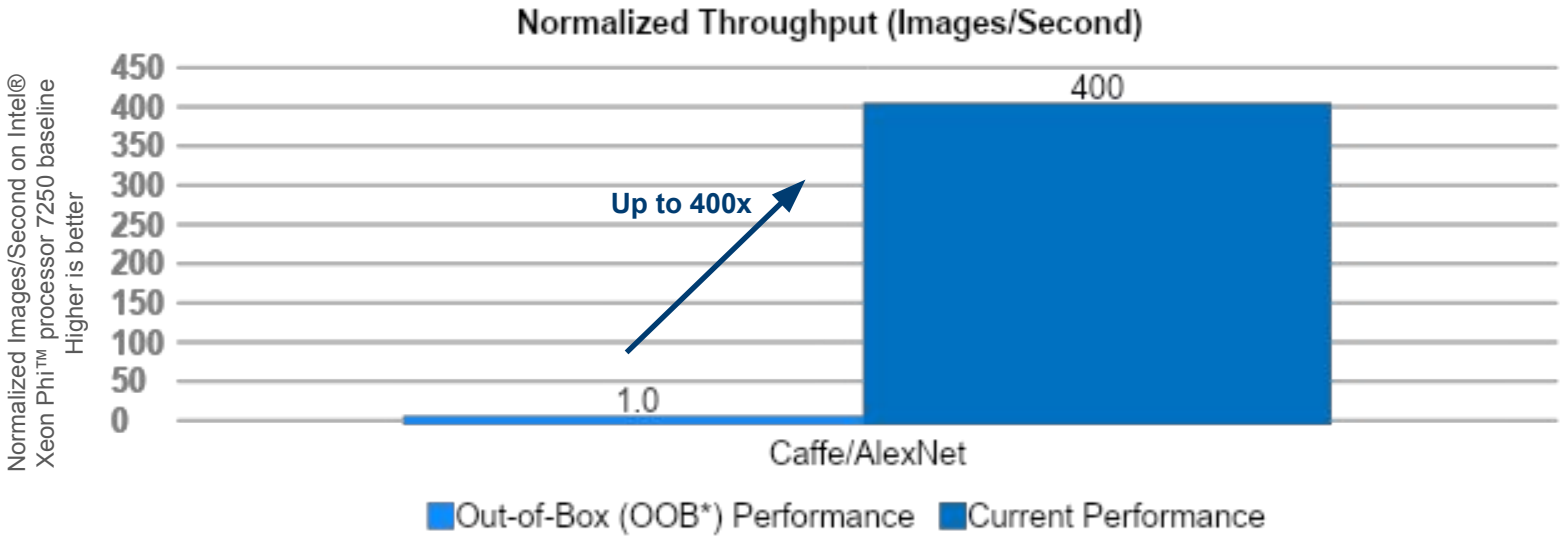
Deep learning with Intel® MKL-DNN

Intel® MKL-DNN Programming Model

- **Primitive** – any operation (convolution, data format re-order, memory)
  - **Operation/memory descriptor** - convolution parameters, memory dimensions
  - **Descriptor** - complete description of a primitive
  - **Primitive** – a specific instance of a primitive relying on descriptor
- **Engine** – execution device (e.g., CPU)
- **Stream** – execution context

```
/* Initialize CPU engine */
auto cpu_engine = mkldnn::engine(mkldnn::engine::cpu, 0);
/* Create a vector of primitives */
std::vector<mkldnn::primitive> net;
/* Allocate input data and create a tensor structure that describes it */
std::vector<float> src(2 * 3 * 227 * 227);
mkldnn::tensor::dims conv_src_dims = {2, 3, 227, 227};
/* Create memory descriptors, one for data and another for convolution input */
auto user_src_md = mkldnn::memory::desc({conv_src_dims},
mkldnn::memory::precision::f32, mkldnn::memory::format::nchw);
auto conv_src_md = mkldnn::memory::desc({conv_src_dims},
mkldnn::memory::precision::f32, mkldnn::memory::format::any);
/* Create convolution descriptor */
auto conv_desc = mkldnn::convolution::desc(
mkldnn::prop_kind::forward, mkldnn::convolution::direct,
conv_src_md, conv_weights_md, conv_bias_md, conv_dst_md,
{1, 1}, {0, 0}, mkldnn::padding_kind::zero);

/* Create a convolution primitive descriptor */
auto conv_pd = mkldnn::convolution::primitive_desc(conv_desc, cpu_engine);

/* Create a memory descriptor and primitive */
auto user_src_memory_descriptor
= mkldnn::memory::primitive_desc(user_src_md, engine);
auto user_src_memory = mkldnn::memory(user_src_memory_descriptor, src);

/* Create a convolution primitive and add it to the net */
auto conv = mkldnn::convolution(conv_pd, conv_input, conv_weights_memory,
conv_user_bias_memory, conv_dst_memory);
net.push_back(conv);

/* Create a stream, submit all primitives and wait for completion */
mkldnn::stream().submit(net).wait();
```

Intel® Xeon Phi ™ processor 7250 up to 400x performance increase with Intel Optimized Frameworks compared to baseline out of box performance

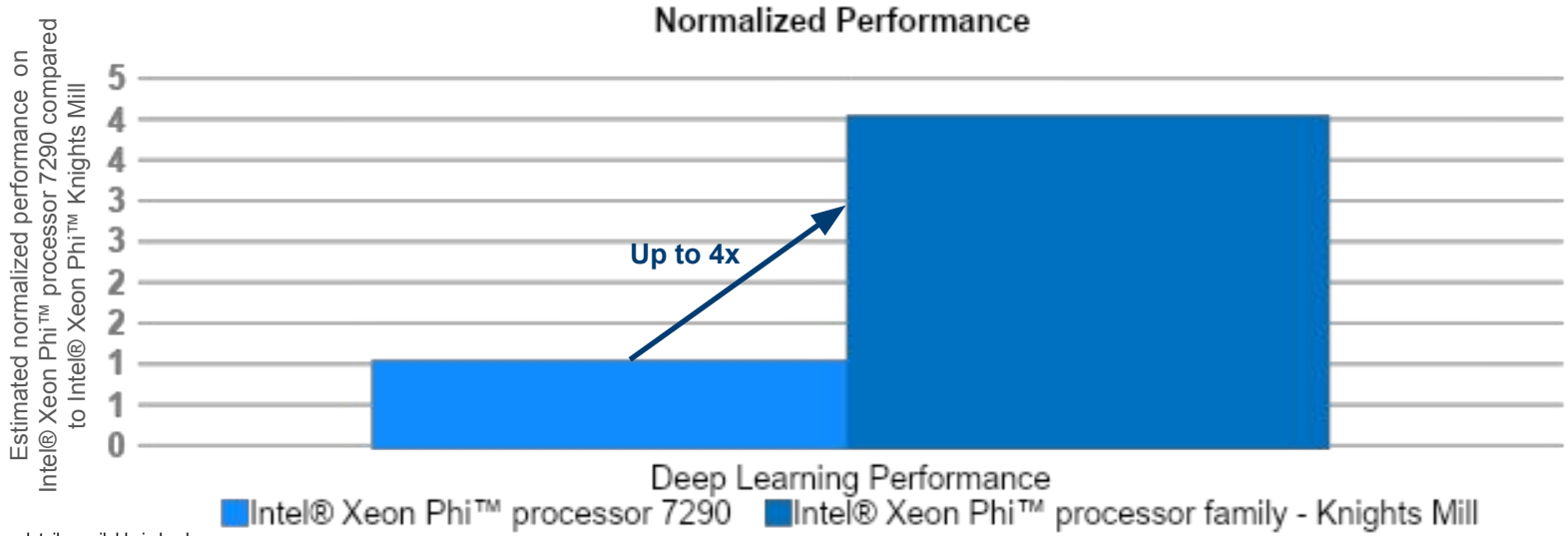


Configuration details available in backup

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 as of November 2016

Intel® Xeon Phi ™ processor Knights Mill up to 4x estimated performance improvement over Intel® Xeon Phi™ processor 7290



Configuration details available in backup

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

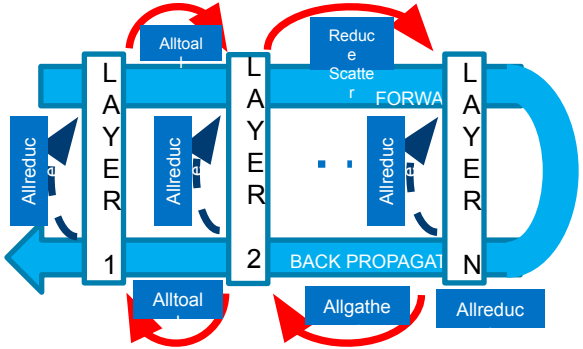
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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 baseline Intel® Xeon Phi™ processor 7290 as of November 2016

INTEL® Machine Learning Scaling Library

Intel® Machine Learning Scaling Library (MLSL)

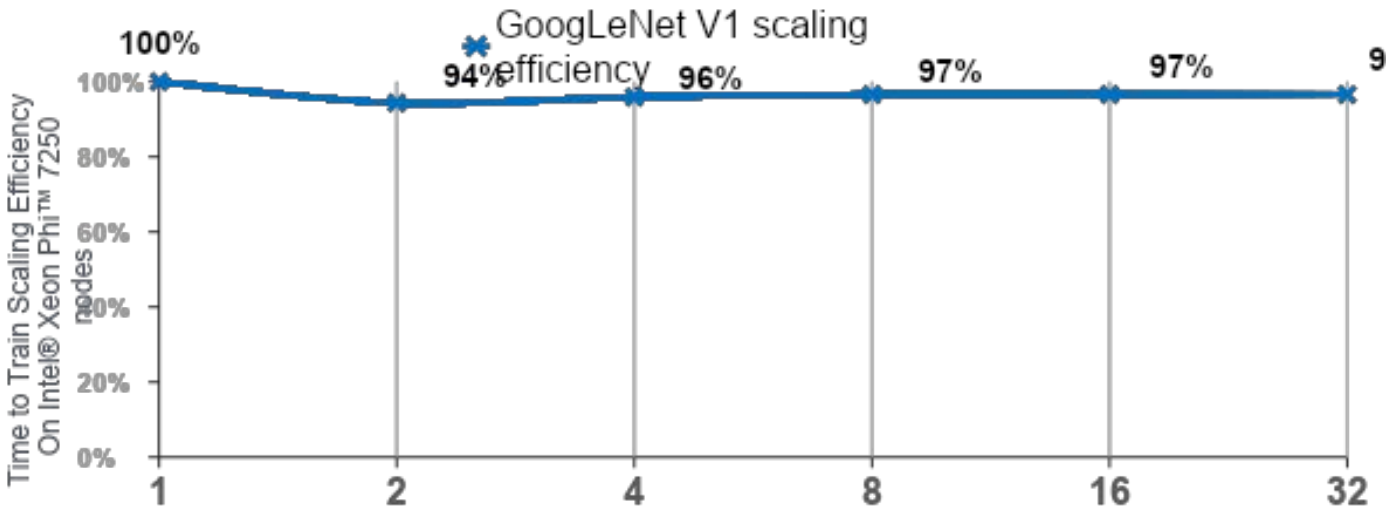
Deep learning abstraction of message-passing implementations.

- Built on top of MPI, allows other communication libraries to be used
- Optimized to drive scalability of communication patterns
- Works across various interconnects: Intel® Omni-Path Architecture, InfiniBand, and Ethernet
- Common API to support Deep Learning frameworks (Caffe, Theano, Torch etc.)



Scaling Deep Learning to 32 nodes and beyond.

Intel® Xeon Phi™ Processor 7250 GoogleNet V1 Time-To-Train Scaling Efficiency up to 97% on 32 nodes



Data pre-partitioned across all nodes in the cluster before training. There is no data transferred over the fabric while training.

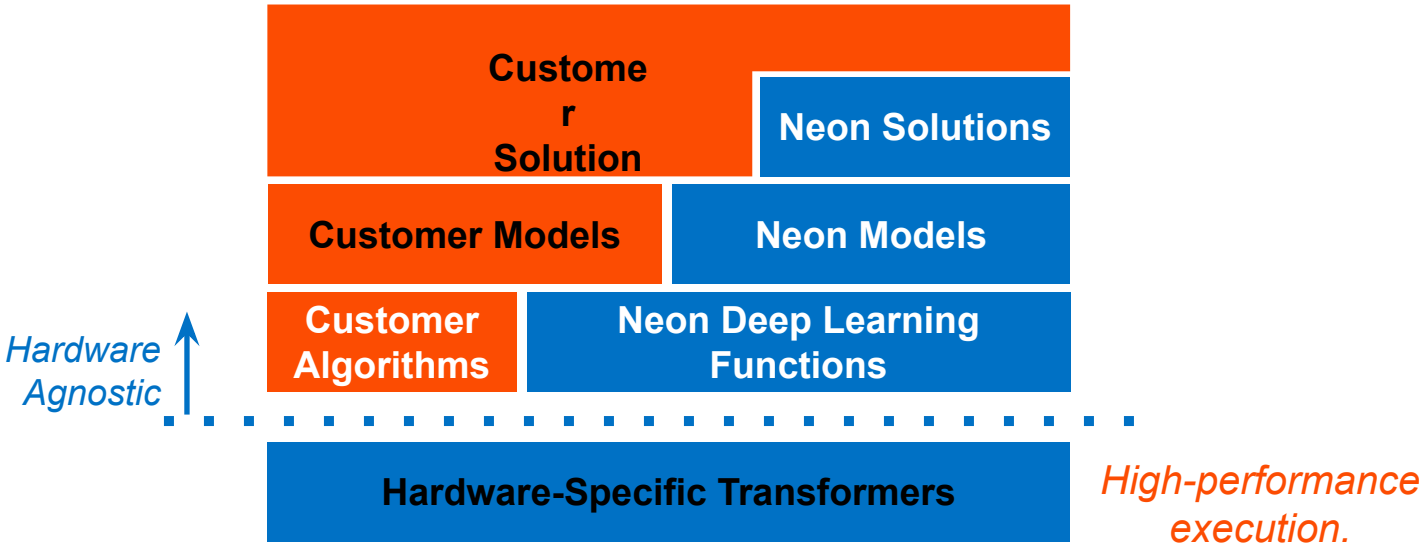
Configuration details available in backup

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

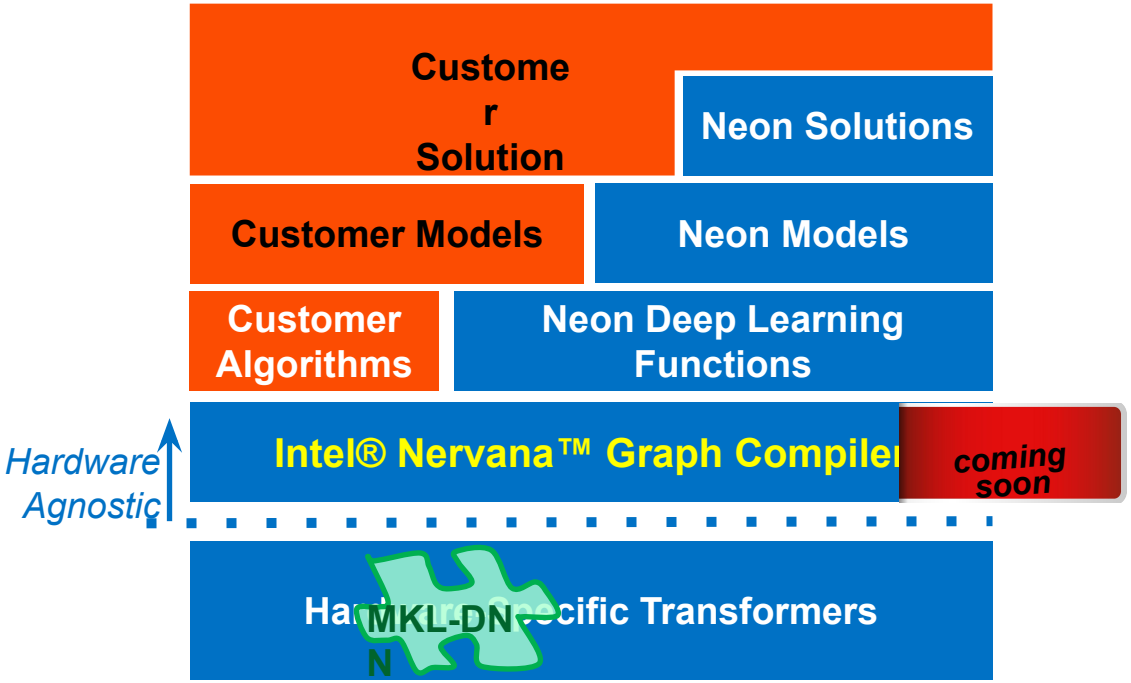
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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 as of November 2016

NeON framework

Neon: DL Framework with Blazing Performance



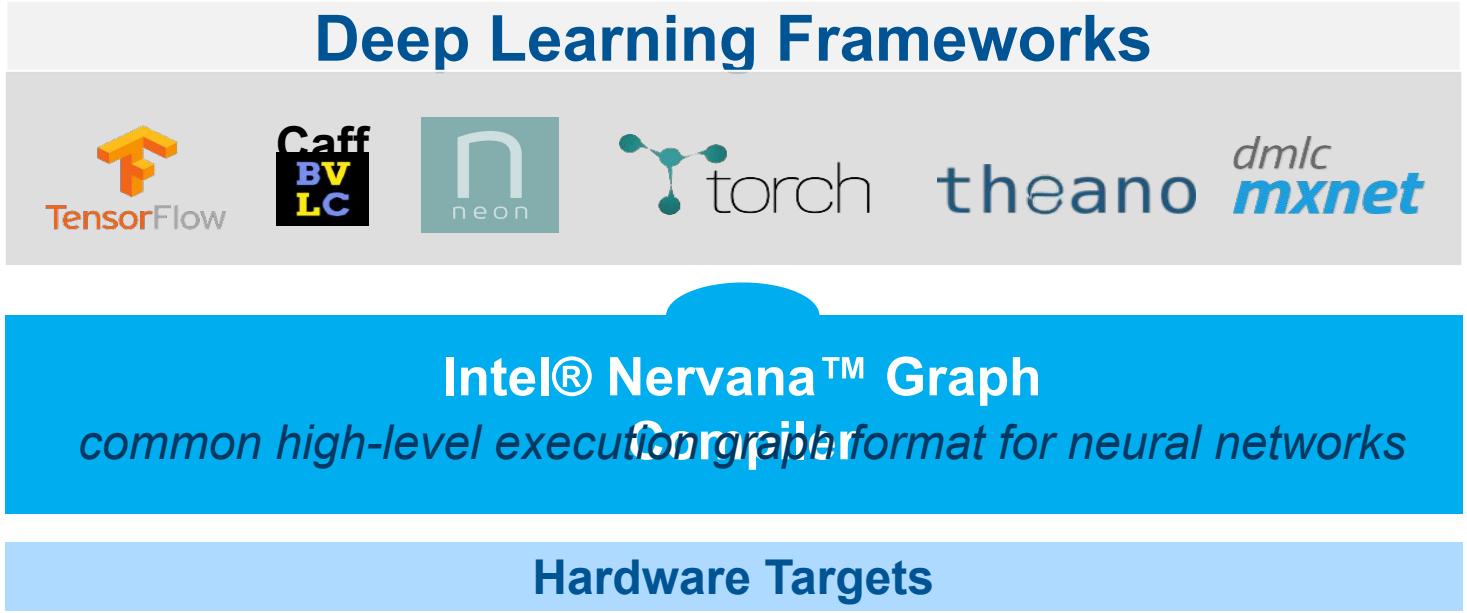
Intel® Nervana™ Graph Compiler



Intel® Nervana™ Graph Compiler:
High-level execution graph for neural networks to enable optimizations that are applicable across multiple HW targets.

- Efficient buffer allocation
- Training vs inference optimizations
- Efficient scaling across multiple nodes
- Efficient partitioning of subgraphs
- Compounding of ops

Intel® Nervana™ Graph Compiler as the performance building block…



…to accelerate all the latest DL innovations across the industry.

INTEL® DEEP Learning SDK

Intel® Deep Learning SDK

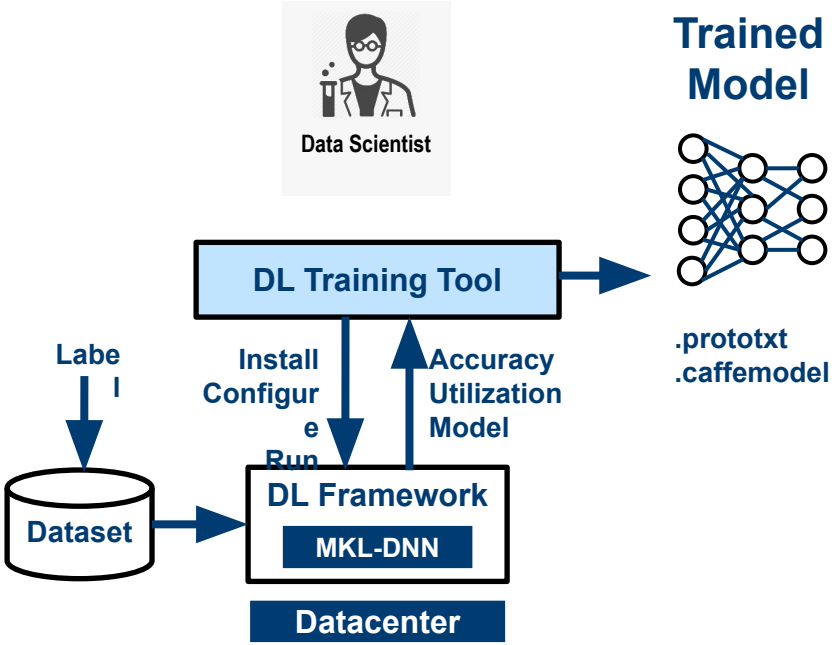
Accelerate Your Deep Learning Solution

A free set of tools for data scientists and software developers to develop, train, and deploy deep learning solutions

"Plug & Train/Deploy"	Maximum Performance	Increased Productivity
Simplify installation & preparation of deep learning models using popular deep learning frameworks on Intel hardware	Optimized performance for training and inference on Intel® Architecture	Faster Time-to-market for training and inference, Improve model accuracy, Reduce total cost of ownership

Deep Learning Training Tool

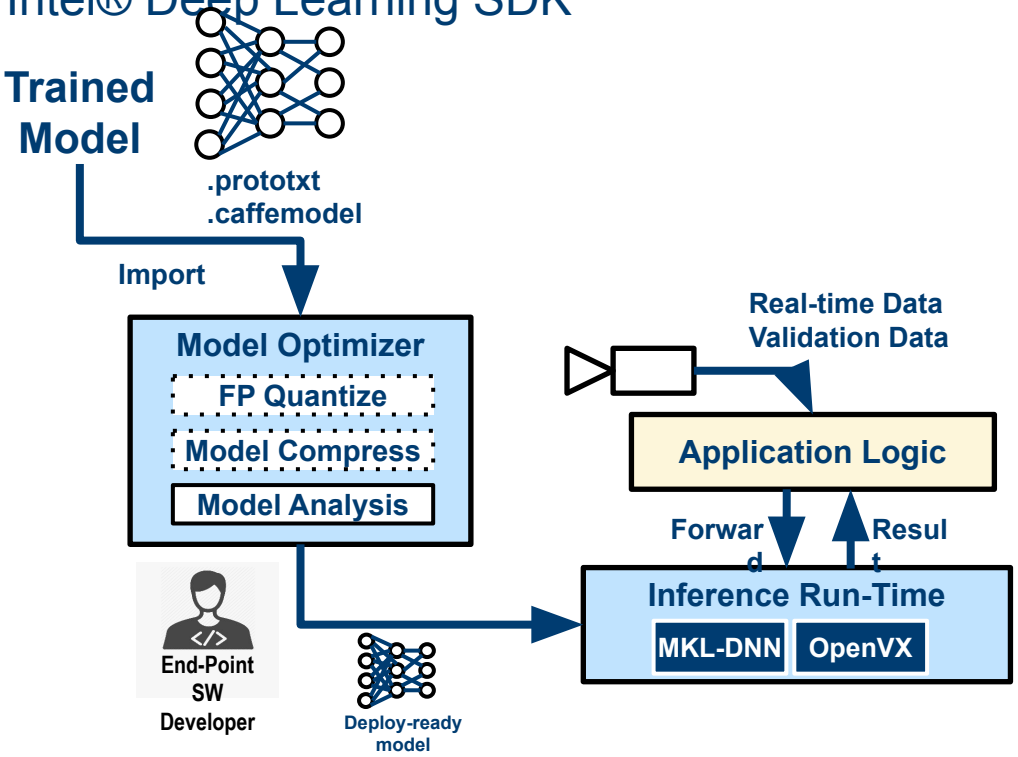
Intel® Deep Learning SDK



- Simplify installation of Intel optimized Deep Learning Frameworks
- Easy and Visual way to Set-up, Tune and Run Deep Learning Algorithms:
  - ✔ Create training dataset
  - ✔ Design model with automatically optimized hyper-parameters
  - ✔ Launch and monitor training of multiple candidate models
  - ✔ Visualize training performance and accuracy

Deep Learning Deployment Tool

Intel® Deep Learning SDK



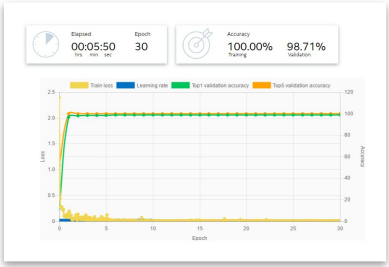
Unleash fast scoring performance on Intel products while abstracting the HW from developers

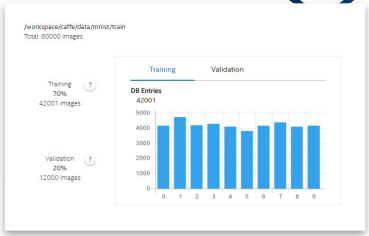
- Imports trained models from all popular DL framework regardless of training HW
- Compresses model for improved execution, storage & transmission (pruning, quantization)
- Generate Inference HW-Specific Code (C/C++, OpenVX, OpenCL, etc.)
- Enables seamless integration with full system / application software stack

Deep Learning Tools for End-to-End Workflow

Intel® Deep Learning SDK

Intel DL Training Tool





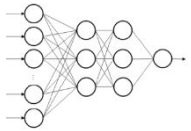
- **INSTALL / SELECT IA-Optimized Frameworks**
- **PREPARE / CREATE Dataset with Ground-truth**
- **DESIGN / TRAIN Model(s) with IA-Opt. Hyper-Parameters**
- **MONITOR Training Progress across Candidate Models**
- **EVALUATE Results and ITERATE**

MKL-DNN Optimized Machine Learning Frameworks

theano Caffe Microsoft Google TensorFlow torch

Xeon (local or cloud)

Intel DL Deployment Tool



```
configure_nn(fpga/,…)
allocate_buffer(…)
fpga_conv(input,output);
fpga_conv(…);
mkl_SoftMax(…);
mkl_SoftMax(…);
…
```

- **IMPORT Trained Model (trained on Intel or 3rd Party HW)**
- **COMPRESS Model for Inference on Target Intel HW**
- **GENERATE Inference HW-Specific Code (OpenVX, C/C++)**
- **INTEGRATE with System SW / Application Stack & TUNE**

Optimized libraries & run-times (MKL-DNN, OpenVX, OpenCL)
Data acquisition (sensors) and acceleration HW (FPGA, etc)

Target Inference Hardware Platform (physical or simulated)

Leading AI research

ALGORITHMIC EVOLUTION FOR FASTER TRAINING

LESS DATA AND SUPERVISION REQUIRED

DEEP NEURAL NETWORKS GETTING AUGMENTED: (NN + X) + MEMORY

LEARNING MODELS THAT ARE EASIER TO REASON

SPARSIFICATION AND PRUNING OF TRAIN MODELS

EFFICIENT DEPLOYMENT

THROUGHPUT, ACCURACY, AND MODEL SIZE TRADEOFFS

REDUCED PRECISION ARITHMETIC

ALL THE WAY TO SINGLE BIT

IMPROVED STRONG SCALING OF DEEP LEARNING TRAINING

LARGER BATCH SIZES AND HIGHER ORDER METHODS

Summary

- Intel provides highly optimized libraries to accelerate all DL frameworks
- Intel® Machine Learning Scaling Library (MLSL) allow to scale DL to 32 nodes and beyond
- Nervana graph compiler, next innovation for DL performance
- Intel® Deep Learning SDK make it easy for you to start exploring DeepLearning
- Intel is committed to provide algorithmic, SW and HW innovations to get best performance for DL on IA

Get more details at:
https://software.intel.com/en-us/ai/deep-learning

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

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


intel
experience
what's inside™

Configuration details

- 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

Configuration details

- 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

Configuration details

- 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® Caffe: Intel internal version of Caffe
- GoogLeNetV1: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf, batch size 1536