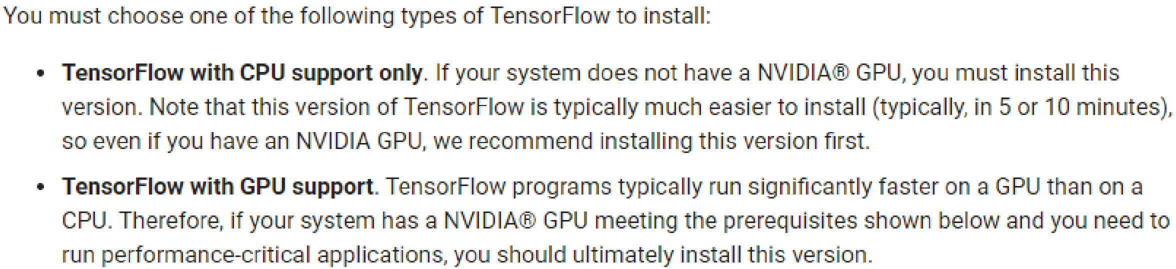
TENSORFLOW* ON MODERN INTEL® ARCHITECTURES

Jing Huang and Vivek Rane
Artificial Intelligence Product Group
Intel

TENSORFLOW* ON CPU HAS BEEN VERY SLOW



You must choose one of the following types of TensorFlow to install:

- **TensorFlow with CPU support only**. If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install (typically, in 5 or 10 minutes), so even if you have an NVIDIA GPU, we recommend installing this version first.
- **TensorFlow with GPU support**. TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting the prerequisites shown below and you need to run performance-critical applications, you should ultimately install this version.

https://www.tensorflow.org/install/install_linux

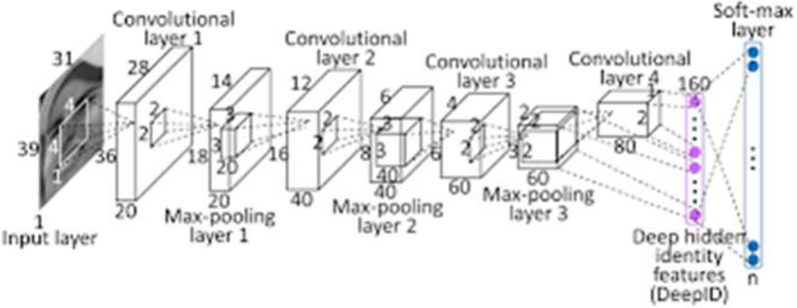
UNTIL TODAY.

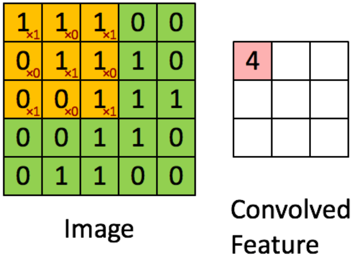
Up to 72x Speedup in Training and 86x Speedup in Inference!
Up-streamed and Ready to Use!

AGENDA

- **Deep Learning & TensorFlow**
- Optimizing TensorFlow on Intel® Architecture
- Summary & Call to Action
- Tutorial on Cray (cori) systems

DEEP LEARNING: CONVOLUTIONAL NEURAL NETWORK





$1_{\times1}$	$1_{\times0}$	$1_{\times1}$	0	0
$0_{\times0}$	$1_{\times1}$	$1_{\times0}$	1	0
$0_{\times1}$	$0_{\times0}$	$1_{\times1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

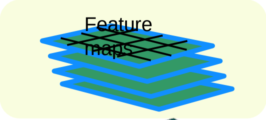
<u>Convolution Parameters:</u>

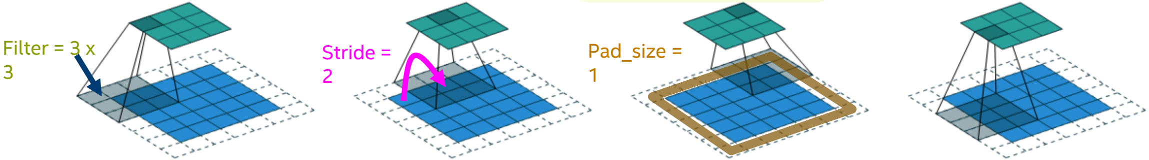
Number of outputs/feature-maps: < 4 >

Filter size: < 3 x 3 >

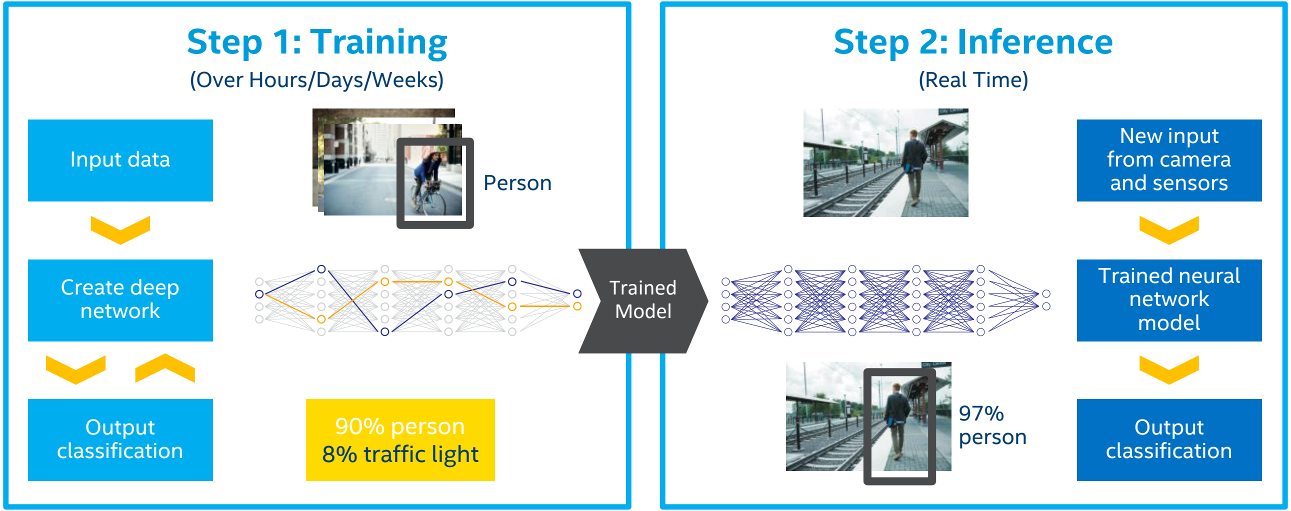
Stride: < 2 >

Pad_size (for corner case): <1>





DEEP LEARNING: TRAIN ONCE USE MANY TIMES



Step 1: Training

(Over Hours/Days/Weeks)

Input data

Create deep network

Output classification

Person

90% person
8% traffic light

Trained Model

Step 2: Inference

(Real Time)

New input from camera and sensors

Trained neural network model

Output classification

97% person

DEEP LEARNING: WHY NOW?

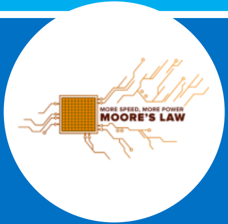
Bigger Data

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

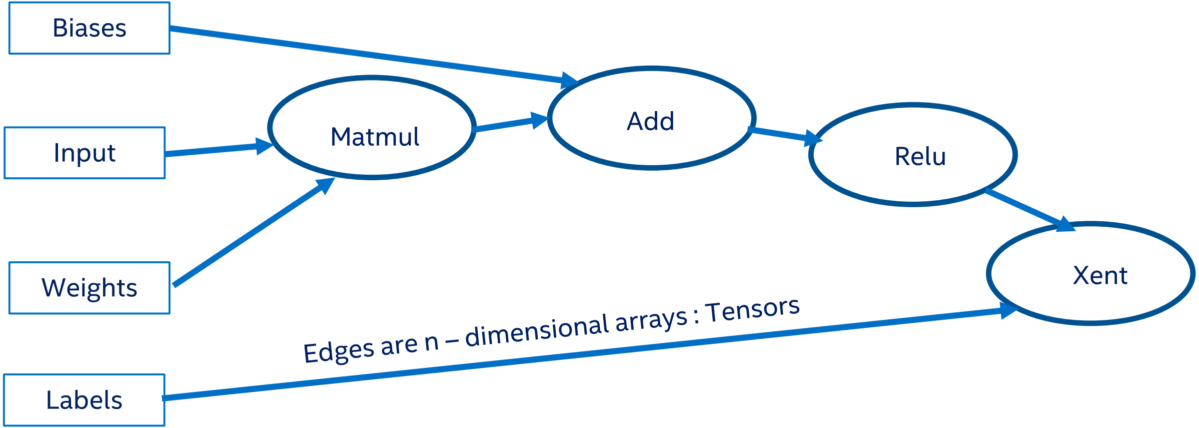
Advances in algorithm innovation, including neural networks, leading to better accuracy in training models

TENSORFLOW

- 2nd generation open source machine learning framework from Google*
- Widely used across Google in many key apps – search, Gmail, photos, translate, etc.
- General computing mathematical framework used on:
  - Deep neural network
  - Other machine learning frameworks
  - HPC applications
- Core system provides set of key computational kernel, extendable user ops
- Core in C++, front end wrapper is in python specifies/drives computation
- Multi-node support using proprietary GRPC protocols

TENSORFLOW: COMPUTATION IS A DATAFLOW GRAPH WITH TENSORS

- Google's open source machine learning framework
- https://github.com/tensorflow/tensorflow



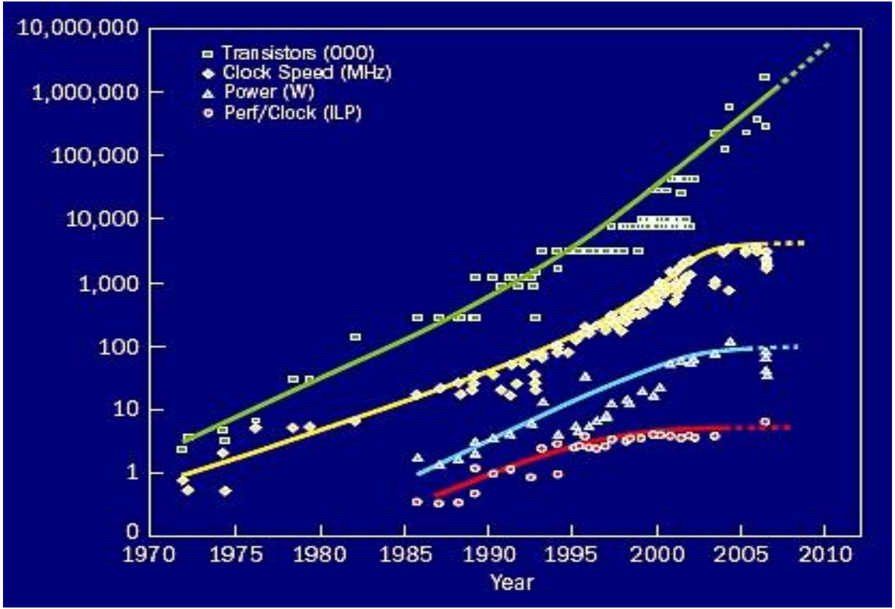
Example from Jeff Dean's presentation

AGENDA

- Deep Learning & TensorFlow
- **Optimizing TensorFlow on Intel® Architecture**
  - Why Optimize
  - Optimization Challenges & Techniques Used
  - Performance Results
- Summary & Call to Action

OPTIMIZATION MATTERS ON MODERN ARCHITECTURES WITH HIGH CORE COUNTS AND WIDE SIMD VECTORS

MOORE'S LAW GOES ON!



Increasing clock speeds -> more cores + wider SIMD (Hierarchical parallelism)

COMBINED AMDAHL'S LAW FOR VECTOR MULTICORES*

$$\boldsymbol{Speedup} = \left(\frac{1}{Serial_{frac} + \frac{1 - Serial_{frac}}{\boldsymbol{NumCores}}}\right) * \left(\frac{1}{Scalar_{frac} + \frac{1 - Scalar_{frac}}{\boldsymbol{VectorLength}}}\right)$$

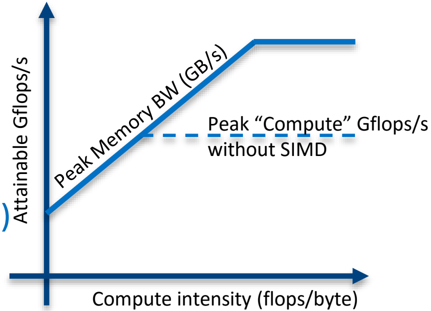
Goal: *Reduce* ***Serial Fraction*** and *Reduce* ***Scalar Fraction*** of Code

Compute Bound Performance

Most kernels of ML codes are compute bound i.e. raw FLOPS matter

Roofline Model

Gflops/s = min (Peak Gflops/s, Stream BW * flops/byte)



OPTIMIZING TENSORFLOW AND DEEP LEARNING WORKLOADS

PERFORMANCE OPTIMIZATION ON MODERN PLATFORMS

Hierarchical Parallelism

Coarse-Grained / multi-node

Domain decomposition

Fine-Grained Parallelism / within node

Sub-domain: 1) Multi-level domain decomposition (ex. across layers)
2) Data decomposition (layer parallelism)

Scaling

- Improve load balancing
- Reduce synchronization events, all-to-all comms

Utilize all the cores

- OpenMP, MPI, TBB…
- Reduce synchronization events, serial code
- Improve load balancing

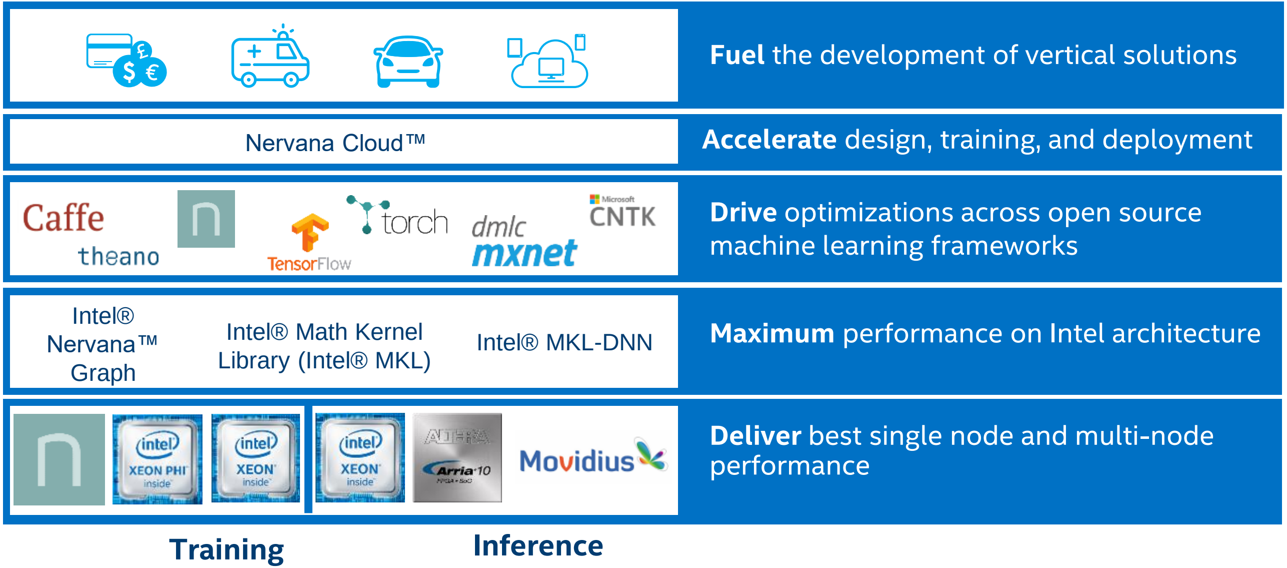
Vectorize/SIMD

- Unit strided access per SIMD lane
- High vector efficiency
- Data alignment

Efficient memory/cache use

- Blocking
- Data reuse
- Prefetching
- Memory allocation

INTEL STRATEGY: OPTIMIZED DEEP LEARNING ENVIRONMENT



Fuel the development of vertical solutions

Nervana Cloud™

Accelerate design, training, and deployment

Caffe theano TensorFlow torch dmlc mxnet Microsoft CNTK

Drive optimizations across open source machine learning frameworks

Intel® Nervana™ Graph

Intel® Math Kernel Library (Intel® MKL)

Intel® MKL-DNN

Maximum performance on Intel architecture

intel XEON PHI inside

intel XEON inside

intel XEON inside

ALTERA Arria 10 FPGA + SoC

Movidius

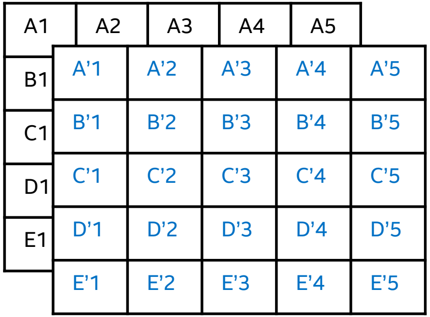
Deliver best single node and multi-node performance

Training

Inference

EXAMPLE CHALLENGE 1: DATA LAYOUT HAS BIG IMPACT ON PERFORMANCE

- Data layouts impact performance
  - Sequential access to avoid gather/scatter
  - Have iterations in inner most loop to ensure high vector utilization
  - Maximize data reuse; e.g. weights in a convolution layer
- Converting to/from optimized layout is some times less expensive than operating on unoptimized layout



A1	A2	A3	A4	A5
B1				
C1				
D1				
E1				

A'1	A'2	A'3	A'4	A'5
B'1	B'2	B'3	B'4	B'5
C'1	C'2	C'3	C'4	C'5
D'1	D'2	D'3	D'4	D'5
E'1	E'2	E'3	E'4	E'5

A1	A2	...	B1	..	A'1	A'2	..

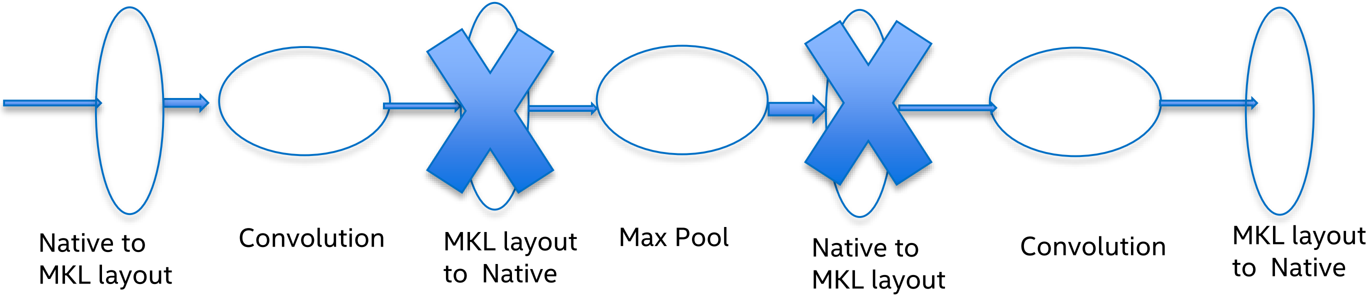
Better optimized for some operations

vs

A1	A'1	A2	A'2	..	B1	B'1	..

EXAMPLE CHALLENGE 2: MINIMIZE CONVERSIONS OVERHEAD

- End-to-end optimization can reduce conversions
- Staying in optimized layout as long as possible becomes one of the tuning goals
- Minimize the number of back and forth conversions
  - Use of graph optimization techniques



OPTIMIZING TENSORFLOW & OTHER DL FRAMEWORKS FOR INTEL® ARCHITECTURE

- Leverage high performant compute libraries and tools
  - e.g. Intel® Math Kernel Library, Intel® Python, Intel® Compiler etc.
- Data format/shape:
  - Right format/shape for max performance: blocking, 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:
  - e.g. batch size for more parallelism
  - Learning rate and optimizer to ensure accuracy/convergence

INITIAL PERFORMANCE GAINS ON MODERN XEON (2 SOCKETS BROADWELL – 22 CORES)

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**
ConvNet-GoogleNet v1	Images / sec	128	16.87	49.9	112.3	439.7	**6.7x**	**8.8x**
ConvNet-VGG	Images / sec	64	8.2	30.7	47.1	151.1	**5.7x**	**4.9x**

- Baseline using TensorFlow 1.0 release with standard compiler knobs
- Optimized performance using TensorFlow with Intel optimizations and built with
  - bazel build --config=mkl --copt=”-DEIGEN_USE_VML”

INITIAL PERFORMANCE GAINS ON MODERN XEON PHI (KNIGHTS LANDING – 68 CORES)

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**
ConvNet-GoogleNet v1	Images / sec	128	5.43	10.9	106	576.6	**19.5x**	**53x**
ConvNet-VGG	Images / sec	64	1.59	24.6	69.4	251	**43.6x**	**10.2x**

- Baseline using TensorFlow 1.0 release with standard compiler knobs
- Optimized performance using TensorFlow with Intel optimizations and built with
  - bazel build --config=mkl --copt=”-DEIGEN_USE_VML”

ADDITIONAL PERFORMANCE GAINS FROM PARAMETERS TUNING

- Data format: CPU prefers NCHW data format
- Intra_op, inter_op and OMP_NUM_THREADS: set for best core utilization
- Batch size: higher batch size provides for better parallelism
  - Too high a batch size can increase working set and impact cache/memory perf

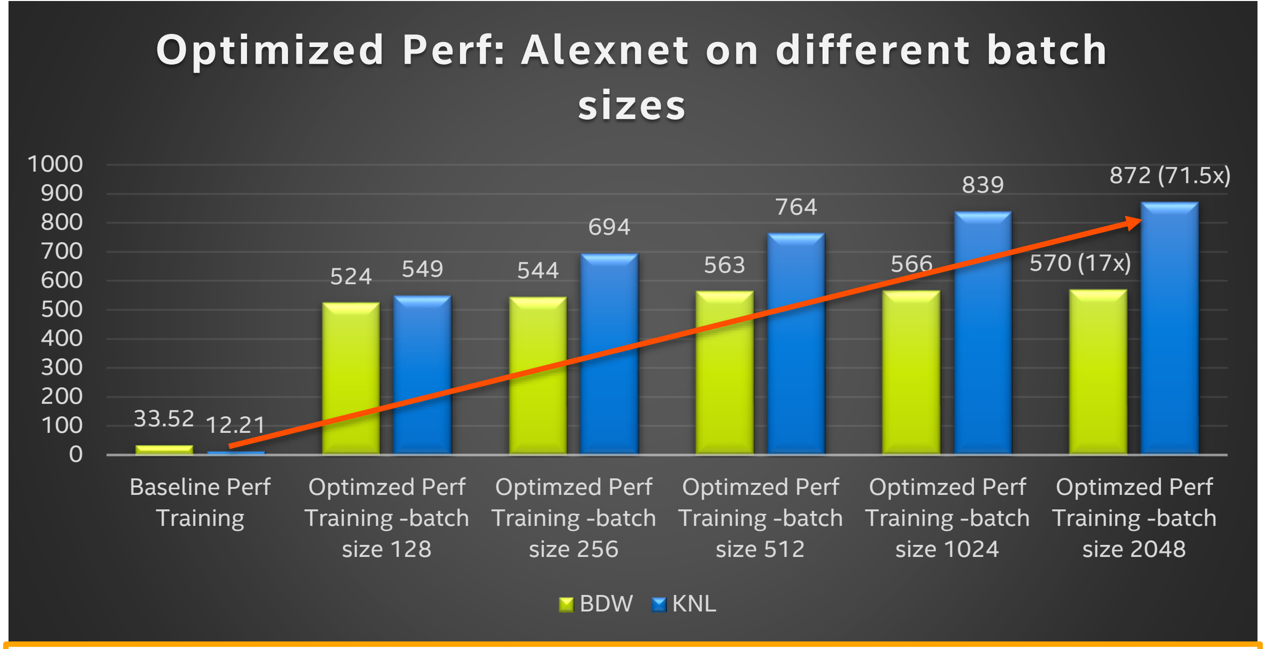
Best Setting for Xeon (Broadwell – 2 Socket – 44 Cores)

Benchmark	Data Format	Inter_op	Intra_op	KMP_BLOCKTIME	Batch size
ConvNet- AlexnetNet	NCHW	1	44	30	2048
ConvNet-Googlenet V1	NCHW	2	44	1	256
ConvNet-VGG	NCHW	1	44	1	128

Best Setting for Xeon Phi (Knights Landing – 68 Cores)

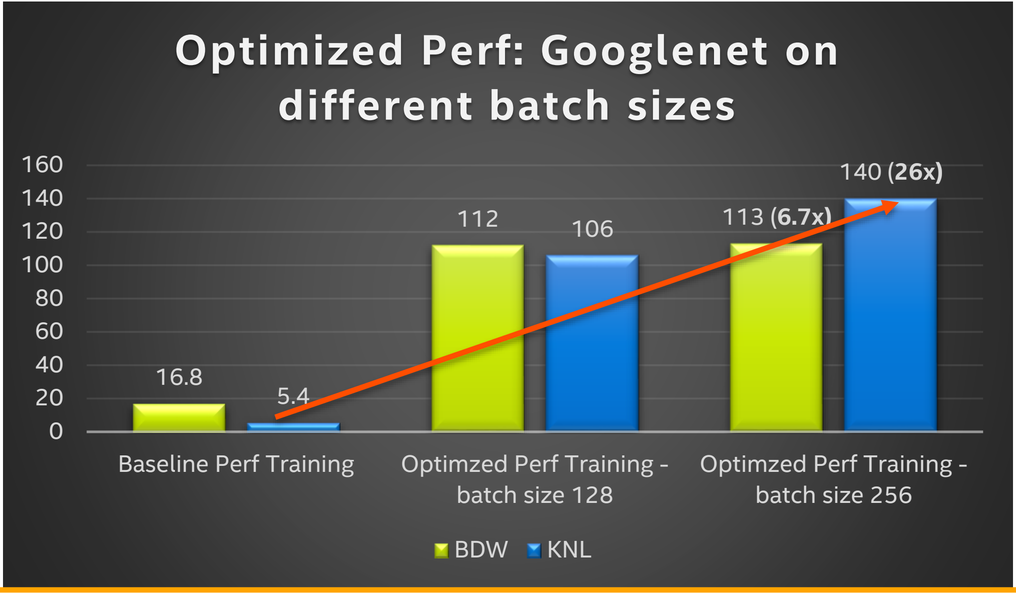
Benchmark	Data Format	Inter_op	Intra_op	KMP_BLOCKTIME	OMP_NUM_THREADS	Batch size
ConvNet- AlexnetNet	NCHW	1	136	30	136	2048
ConvNet-Googlenet V1	NCHW	2 training 1 inference	68	Infinite	68	256
ConvNet-VGG	NCHW	1	136	1	136	128

PERFORMANCE GAINS – CONVNET-ALEXNET TRAINING (IMAGES PER SECOND)



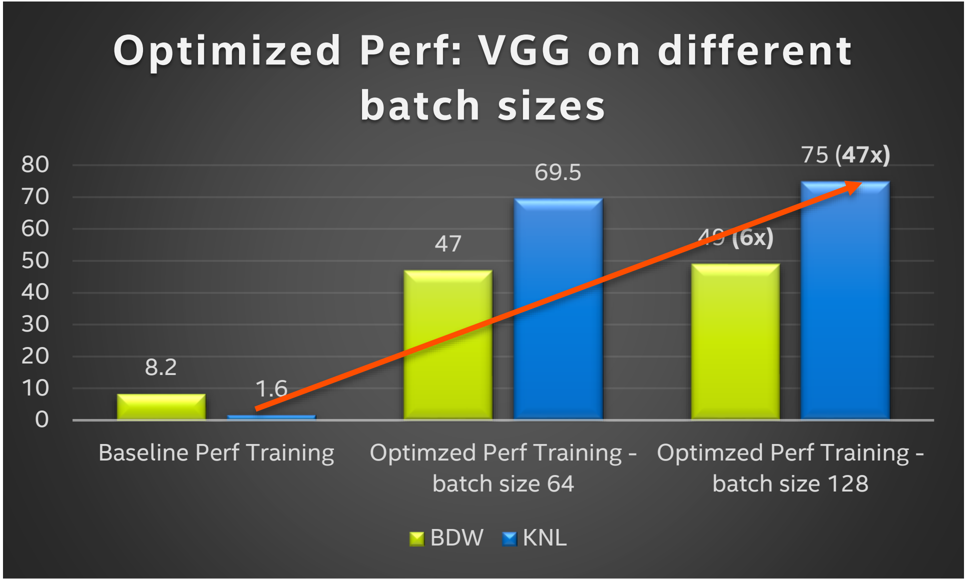
72x Speedup From New Optimizations – available through Google's TensorFlow Git

PERFORMANCE GAINS – CONVNET-GOOGLENET V1 TRAINING (IMAGES PER SECOND)



26x Speedup From New Optimizations – available through Google's TensorFlow Git

PERFORMANCE GAINS – CONVNET-VGG TRAINING (IMAGES PER SECOND)



47x Speedup From New Optimizations – available through Google's TensorFlow Git

HOW DO I GET ORDER OF MAGNITUDE CPU SPEEDUP FOR MY TOPOLOGY?

- Optimized TensorFlow on Intel architectures available from the public git.
  - git clone https://github.com/tensorflow/tensorflow.git
- Configure for best performance on CPU:
  - Run “./configure” from the TensorFlow source directory
  - Select option for MKL (CPU) optimization
  - Automatically downloads latest MKL-ML
- Building for best performance on CPU
  - Use following command to create a pip package that can be used to install the optimized TensorFlow wheel
  - bazel build --config=mkl --copt=“-DEIGEN_USE_VML” --s --c opt //tensorflow/tools/pip_package:build_pip_package
- Install the optimized TensorFlow wheel
  - bazel-bin/tensorflow/tools/pip_package/build_pip_package ~/path_to_save_wheel
  - pip install --upgrade --user ~/path_to_save_wheel/wheel_name.whl

HOW DO I GET ORDER OF MAGNITUDE CPU SPEEDUP FOR MY TOPOLOGY? (2)

- Maximum performance requires using all the available cores efficiently
- Users and data scientists should experiment with environment variable settings
  - Best setting depend on topology and platform (e.g., number of cores)
  - Example of ConvNet-Alexnet environment settings on Knights Landing
    - KMP_BLOCKTIME = 30
    - KMP_SETTINGS = 1
    - KMP_AFFINITY= granularity=fine,verbose,compact,1,0
    - OMP_NUM_THREADS= 136 (Xeon Phi has 68 physical cores)
- Knobs in the Python topology can also impact performance:
  - Data format: using NCHW format to avoid additional internal format conversions to get maximum performance
  - Matmul layer: the second input matrix should be transposed for better performance
  - Intra_op /inter_op: experiment with intra_op/inter_op for each topology/ platform
    - Example of ConvNet-Alenet settings on Xeon Phi
      - inter_op = 2
      - intra_op = 136

SUMMARY

- TensorFlow is widely used DL and AI framework
  - It has been slow on CPU until now
- Significant performance gains from optimization on modern Intel® Xeon® and Xeon Phi™ processors
- Traditional optimization techniques: vectorization, parallelization, cache blocking, etc.
- Unique performance challenges: data layout, hyper parameters, inter/intra layer parallelization, etc.

CALL TO ACTION

Latest TensorFlow with Intel optimizations directly from TensorFlow GIT repository

+

Use the right configuration, building and best parameter settings

=

Orders of magnitude higher CPU performance for inference and training

TUTORIAL ON CRAY (CORI) KNL SYSTEMS

Instructions also at:
/global/cscratch1/sd/vrane/README

1. Get TensorFlow from Google's repository

```
git clone https://github.com/tensorflow/tensorflow.git
```

2. Get convnet Alexnet

```
wget https://raw.githubusercontent.com/soumith/convnet-benchmarks/master/tensorflow/benchmark_alexnet.py
```

3. Load Java (bazel 0.5.4 needs Java 1.8+)

```
module load java
```

4. Load Python/Pip

```
module load python
```

5. Load gcc (need 5.4+)

```
module load gcc
```

6. Setup Bazel

```
wget https://github.com/bazelbuild/bazel/releases/download/0.4.5/bazel-0.4.5-installer-linux-x86_64.sh
chmod +x bazel-0.4.5-installer-linux-x86_64.sh
./bazel-0.4.5-installer-linux-x86_64.sh --user
export PATH=~/bin/:$PATH
```

7. Configure tensorflow

```
cd tensorflow
./configure
MKL -> yes
everything else -> default
```

8. Build!

```
bazel build --config=mkl --copt="-DEIGEN_USE_VML" -s -c opt //tensorflow/tools/pip_package:build_pip_package
```

9. Make a wheel

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package /<full-path>/wheel/
```

10. Install the wheel

(If you don't want to build, there is a pre-built one using the instructions above available at: /global/cscratch1/sd/vrane/tensorflow-1.1.0-cp27-cp27mu-linux_x86_64.whl)

```
pip install /<full-path>/wheel/tensorflow-1.1.0rc2-cp27-cp27mu-linux_x86_64.whl --upgrade --target=/<full-path>/install/
```

11. Set PYTHONPATH to installed location

```
export PYTHONPATH=/<full-path>/install/
```

12. Find a node to run the benchmark

```
salloc --reservation=CUG2C -N 1 -p regular -C knl,quad,flat -t 60 -A ntrain
```

13. Run the benchmark you downloaded in step 2 (convnet Alexnet)

```
python benchmark_alexnet.py
```

14. Now optimize the benchmark for KNL:

a. OMP_NUM_THREADS and inter/intra-op settings

```
        import os
        os.environ["OMP_NUM_THREADS"] = "136"
        os.environ["KMP_BLOCKTIME"] = "30"
        os.environ["KMP_SETTINGS"] = "1"
        os.environ["KMP_AFFINITY"]= "granularity=fine,verbose,compact,1,0"

tf.app.flags.DEFINE_integer('inter_op', 2, """Inter Op Parallelism Threads.""")

tf.app.flags.DEFINE_integer('intra_op', 136, """Intra Op Parallelism Threads.""")
```

b. Change batch size to 2048

c. Instead of relu_layer, use matmul (transposed) and relu. This formats the data in a manner that allows for faster processing in MKL.

d. Set the allocator to BFC, and supply the intra and inter-op parallelism flags to the session.

```
        config = tf.ConfigProto(inter_op_parallelism_threads=FLAGS.inter_op,intra_op_parallelism_threads=FLAGS.intra_op )
        config.gpu_options.allocator_type = 'BFC'
        sess = tf.Session(config=config)
```

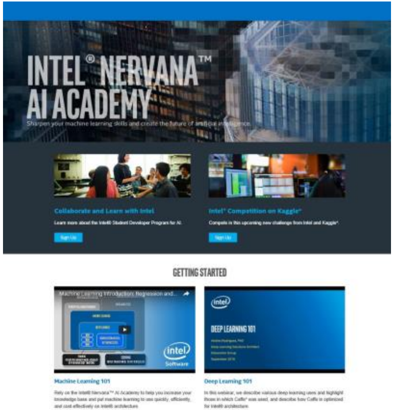
15. Rerun the benchmark to see the performance improvement.

[You can find an optimized version of the benchmark here (with the modifications from step 14): /global/cscratch1/sd/vrane/benchmark_alexnet_knl.py]

```
python benchmark_alexnet.py
```

MORE FROM INTEL...

Intel® Nervana™ AI Academy



INTEL® NERVANA™ AI ACADEMY

Sharpen your machine learning skills and create the future of artificial intelligence.

Collaborate and Learn with Intel

Learn more about the Intel® Student Developer Program for AI.

Sign Up

Intel® Competition on Kaggle*

Compete in this upcoming new challenge from Intel and Kaggle*.

Sign Up

GETTING STARTED

Machine Learning 101

Rely on the Intel® Nervana™ AI Academy to help you increase your knowledge base and put machine learning to use quickly, efficiently, and cost effectively on Intel® architecture.

Deep Learning 101

In this webinar, we describe various deep learning uses and highlight those in which Caffe* was used, and describe how Caffe is optimized for Intel® architecture.

Visit:
software.intel.com/ai/academy

Kaggle Competition: Intel® & MobileODT Cervical Cancer Screening

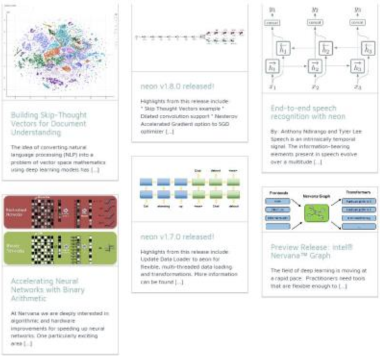
kaggle

MobileODT
Mobile Optical Detection Technologies

intel
Software

Register:
kaggle.com/c/intel-mobileodt-cervical-cancer-screening

Additional Deep Learning Resources



Building Skip-Thought Vectors for Document Understanding

The idea of converting natural language processing (NLP) into a problem of vector space mathematics using deep learning models has […]

Accelerating Neural Networks with Binary Arithmetic

At Nervana we are deeply interested in algorithmic and hardware improvements for speeding up neural networks. One particularly exciting area […]

neon v1.8.0 released!

Highlights from this release include: * Skip Thought Vectors example * Dilated convolution support * Nesterov Accelerated Gradient option to SGD optimizer […]

neon v1.7.0 released!

Highlights from this release include: Update Data Loader to aeon for flexible, multi-threaded data loading and transformations. More information can be found […]

End-to-end speech recognition with neon

By: Anthony Ndirango and Tyler Lee Speech is an intrinsically temporal signal. The information-bearing elements present in speech evolve over a multitude […]

Preview Release: Intel® Nervana™ Graph

The field of deep learning is moving at a rapid pace. Practitioners need tools that are flexible enough to […]

Visit:
nervanasys.com/learn

Legal Disclaimers

- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number
- Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect 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.
- Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.
- Relative performance is calculated by assigning a baseline value of 1.0 to one benchmark result, and then dividing the actual benchmark result for the baseline platform into each of the specific benchmark results of each of the other platforms, and assigning them a relative performance number that correlates with the performance improvements reported.
- SPEC, SPECint, SPECfp, SPECrate, SPECpower, SPECjbb, SPECompG, SPEC MPI, and SPECjEnterprise* are trademarks of the Standard Performance Evaluation Corporation. See http://www.spec.org for more information.
- TPC Benchmark, TPC-C, TPC-H, and TPC-E are trademarks of the Transaction Processing Council. See http://www.tpc.org for more information.
- No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Xeon® processor E7-8800/4800/2800 v2 product families or Intel® Itanium® 9500 series-based system (or follow-on generations of either.) Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient System Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient Memory Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel® processor and enabled technology(ies). built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Q&A:

TENSORFLOW* ON MODERN INTEL® ARCHITECTURES

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

CURRENT INTEL® XEON PLATFORMS

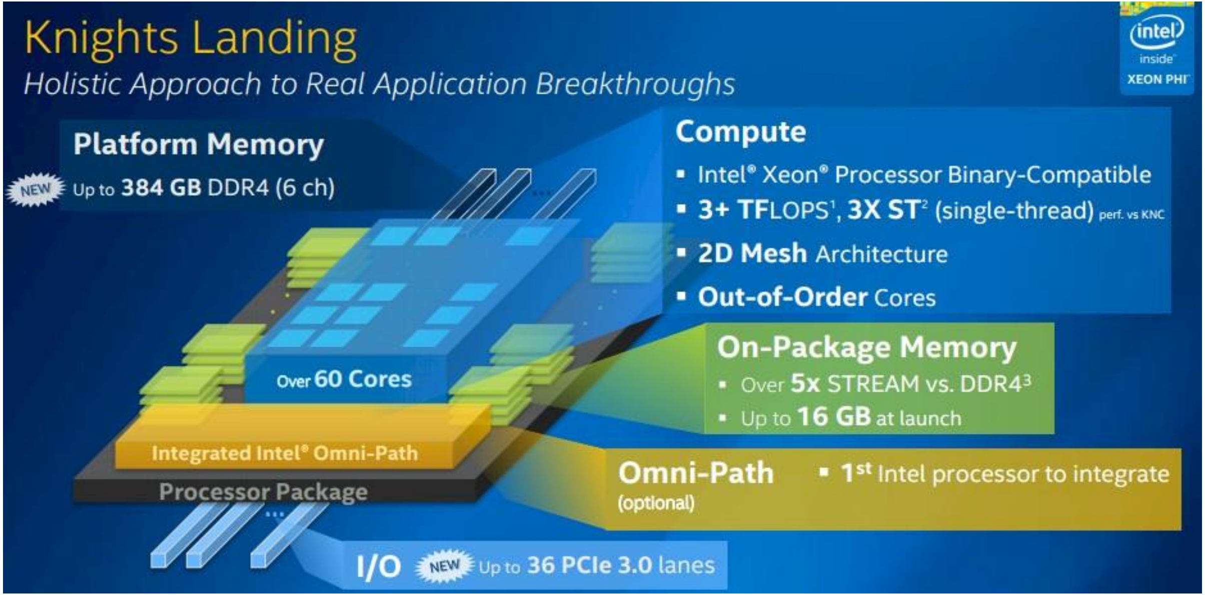
45nm Process Technology	32nm Process Technology		22nm Process Technology		14nm Process Technology
Nehalem	Westmere	Sandy Bridge	Ivy Bridge	Haswell	Broadwell
NEW Intel® Microarchitecture (Nehalem)	Intel Microarchitecture (Nehalem)	NEW Intel Microarchitecture (Sandy Bridge)	Intel Microarchitecture (Sandy Bridge)	NEW Intel Microarchitecture (Haswell)	Intel Microarchitecture (Haswell)
TOCK	TICK	TOCK	TICK	TOCK	TICK

Latest released – Broadwell (14nm process)

- Intel's foundation of HPC and ML performance
- Suited for full scope of workloads
- Industry leading performance/watt for serial & highly parallel workloads.
- Upto 22 cores / socket (Broadwell-EP) (w/ Hyper-Threading technology)

Software optimization helps maximize benefit and adoption of new features

2ND GENERATION INTEL® XEON PHI™ PLATFORM



Knights Landing

Holistic Approach to Real Application Breakthroughs

Platform Memory

NEW Up to **384 GB** DDR4 (6 ch)

Compute

- Intel® Xeon® Processor Binary-Compatible
- **3+ TFLOPS**[1], **3X ST**[2] (single-thread) perf. vs KNC
- **2D Mesh** Architecture
- **Out-of-Order** Cores

On-Package Memory

- Over **5x** STREAM vs. DDR4[3]
- Up to **16 GB** at launch

Omni-Path

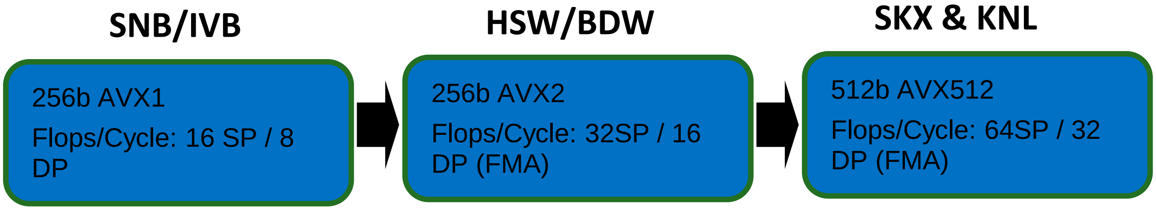
(optional)

- 1st Intel processor to integrate

I/O

NEW Up to **36 PCIe 3.0** lanes

INTEL® AVX TECHNOLOGY



SNB/IVB

256b AVX1

Flops/Cycle: 16 SP / 8 DP

HSW/BDW

256b AVX2

Flops/Cycle: 32SP / 16 DP (FMA)

SKX & KNL

512b AVX512

Flops/Cycle: 64SP / 32 DP (FMA)

AVX	AVX2
256-bit basic FP	Float16 (IVB 2012)
16 registers	256-bit FP FMA
NDS (and AVX128)	256-bit integer
Improved blend	PERMD
MASKMOV	Gather
Implicit unaligned	

AVX512
512-bit FP/Integer
32 registers
8 mask registers
Embedded rounding
Embedded broadcast
Scalar/SSE/AVX “promotions”
Native media additions
HPC additions
Transcendental support
Gather/Scatter

OPTIMIZATION NOTICE

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