INTRODUCTION TO ARTIFICIAL INTELLIGENCE USING INTEL® HARDWARE PLATFORM

Dr. Fabio Baruffa
Sr. Technical Consulting Engineer, Intel IAGS
NAVIGATING THE AI PERFORMANCE PACKAGE

INTRODUCTION TO AI
Overview of Deep Learning Software

INTEL® XEON® SCALABLE PROCESSORS
First and second generation: Skylake / Cascade Lake

INTEL® DEEP LEARNING BOOST
Intel® AVX-512 Vector Neural Network Instructions (VNNI)
Introduction to AI
Overview of Deep Learning Software

- What are AI, Machine Learning, and Deep Learning?
- Deep Learning Software breakdown
- Popular AI Neural Networks and their uses
- Intel’s AI software tools
Artificial Intelligence is the ability of machines to learn from experience, without explicit programming, in order to perform cognitive functions associated with the human mind.

No one size fits all approach to AI
Choose the right AI approach for your challenge
Supervised learning example

Deep learning (image recognition)

A 'Supervised Learning' Example

Choose the right AI approach for your challenge
DEEP LEARNING GLOSSARY

LIBRARY
Hardware-optimized mathematical and other primitive functions that are commonly used in machine & deep learning algorithms, topologies & frameworks

FRAMEWORK
Open-source software environments that facilitate deep learning model development & deployment through built-in components and the ability to customize code

TOPOLOGY
Wide variety of algorithms modeled loosely after the human brain that use neural networks to recognize complex patterns in data that are otherwise difficult to reverse engineer

Translating common deep learning terminology
Deep Learning Usages & Key Topologies

Image Recognition
- Resnet-50
- Inception V3
- MobileNet
- SqueezeNet

Object Detection
- R-FCN
- Faster-RCNN
- Yolo V2
- SSD-VGG16, SSD-MobileNet

Image Segmentation
- Mask R-CNN

Language Translation
- GNMT

Text to Speech
- Wavenet

Recommendation System
- Wide & Deep, NCF

There are many deep learning usages and topologies for each
Deep learning in practice

Optimization Notice

Time-to-solution is more significant than time-to-train
SPEED UP DEVELOPMENT

using open AI software

TOOLKITS
App developers

LIBRARIES
Data scientists

KERNALS
Library developers

ANALYTICS
ZOO
Open source platform for building E2E Analytics & AI applications on Apache Spark* with distributed TensorFlow*, Keras*, BigDL

Analytics Zoo

Deep learning inference deployment on CPU/GPU/FPGA/VPU for Caffe*, TensorFlow*, MXNet*, ONNX*, Kaldi*

OpenVINO

Open source, scalable, and extensible distributed deep learning platform built on Kubernetes (BETA)

Nauba

Intel-optimized Frameworks

Intel® Distribution for Python*
Intel distribution optimized for machine learning

Intel® Data Analytics Acceleration Library (Intel® DAAL)
High performance machine learning & data analytics library

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)
Open source DNN functions for CPU / integrated graphics

nGraph

Open source compiler for deep learning model computations optimized for multiple devices (CPU, GPU, NNP) from multiple frameworks (TF, MXNet, ONNX)

Visit: www.intel.ai/technology

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
NAVIGATING THE AI PERFORMANCE PACKAGE

INTRODUCTION TO AI
Overview of Deep Learning Software

INTEL® XEON® SCALABLE PROCESSORS
First and second generation: Skylake / Cascade Lake

• Deploy AI Everywhere on Intel® Architecture
• Intel® AVX-512 (Advanced Vector Instructions)
DEPLOY AI ANYWHERE

with unprecedented hardware choice

Visit: www.intel.ai/technology

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Single instruction multiple data (SIMD) allows to execute the same operation on multiple data elements using larger registers.

- **Scalar mode**
  - one instruction produces one result
  - E.g. vaddss, (vaddsd)

- **Vector (SIMD) mode**
  - one instruction can produce multiple results
  - E.g. vaddps, (vaddpd)

```
for (i=0; i<n; i++) z[i] = x[i] + v[i];
```

- **SSE (128 Bits reg.):**
  - -> 4 floats
- **AVX (256 Bits reg.):**
  - -> 8 floats
- **AVX512 (512 Bits reg.):**
  - -> 16 floats
EVOLUTION OF SIMD FOR INTEL® PROCESSORS

Intel® Xeon®/Xeon® Scalable Processors
Willamette 2000
Prescott 2004
Merom 2006
Penryn 2007
Nehalem 2008
Sandy Bridge 2011
Haswell 2013
Knights Landing 2015
Skylake server 2015
Cascade Lake server 2019

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
## AVX-512 Compress and Expand

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**VCOMPRESSPD</td>
<td>PS</td>
</tr>
<tr>
<td>**VEXPANDPD</td>
<td>PS</td>
</tr>
</tbody>
</table>

**double/single-precision/doubleword/quadword**

```
vcompresspd YMMWORD PTR [rsi+rax*8]{k1}, ymm1
```
compress Loop Pattern
Auto-vectorization

https://godbolt.org/z/x7gNfb

```c
int compress(double *a, double *__restrict b, int na) {
    int nb = 0;
    for (int ia=0; ia < na; ia++)
    {
        if (a[ia] > 0.)
            b[nb++] = a[ia];
    }

    return nb;
}
```
COMPRESS LOOP PATTERN

AUTO-VECTORIZATION

https://godbolt.org/z/x7gNfb

```c
int compress(double *a, double *__restrict b, int na)
{
    int nb = 0;
    for (int ia=0; ia < na; ia++)
    {
        if (a[ia] > 0.)
            b[nb++] = a[ia];
    }
    return nb;
}
```

Targeting Intel® AVX2
-xcore-avx2 -qopt-report-file=stderr -qopt-report-phase=vec

LOOP BEGIN

remark #15344: loop was not vectorized: vector dependence prevents vectorization.
remark #15346: vector dependence: assumed FLOW dependence between b[nb] (7:4) and a[ia] (7:4)

LOOP END

Targeting Intel® AVX-512
-xcore-avx512 -qopt-report-file=stderr -qopt-report-phase=vec

LOOP BEGIN

remark #15300: LOOP WAS VECTORIZED

LOOP END
COMPRESS LOOP PATTERN
AUTO-VECTORIZATION

https://godbolt.org/z/x7gNfb

```c
int compress(double *a, double *__restrict b, int na) {
    int nb = 0;
    for (int ia=0; ia <na; ia++) {
        if (a[ia] > 0.)
            b[nb++] = a[ia];
    }
    return nb;
}
```

Key Take Aways
Compress/Expand loop pattern doesn’t vectorize on architectures like Intel® AVX2 and the previous ones and does with Intel® AVX512
## Compress Loop Performance

<table>
<thead>
<tr>
<th>Compiler Options</th>
<th>Speedup (in C)</th>
<th>Speedup (in FORTRAN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Loops (~O2 -xCORE-AVX2)</td>
<td>1.0x</td>
<td>1.0x</td>
</tr>
<tr>
<td>(-O2 -xCORE-AVX512)</td>
<td>12.8x</td>
<td>12.2x</td>
</tr>
</tbody>
</table>

Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. The results above were obtained on an Intel® Xeon® Platinum 8168 system, frequency 2.7 GHz, running Red Hat® Enterprise Linux® Server 7.2 and using the Intel® Fortran Compiler version 18.0 update 1.

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.
NAVIGATING THE AI PERFORMANCE PACKAGE

INTRODUCTION TO AI
Overview of Deep Learning Software

INTEL® XEON® SCALABLE PROCESSORS
First and second generation: Skylake / Cascade Lake

INTEL® DEEP LEARNING BOOST
Intel® AVX-512 Vector Neural Network Instructions (VNNI)
  • Boost Deep Learning Inference with VNNI

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
FAST EVOLUTION OF AI CAPABILITY ON INTEL® XEON® PLATFORM

Grantley

Haswell (HSX)
E5 V3

Broadwell (BDX)
E5 V4

Intel® AVX2
(256 bit)

Purley

Skylake (SKX)
SP (Scalable Processor)
E.g. Gold 8180
Gold 5117
Silver 4110

Intel® AVX512
(512 bit)
FP32, INT8, ...

Cascade Lake (CLX)
SP (Scalable Processor)
E.g. Gold 8280
Gold 5218

Intel® AVX512 VNNI
(512 bit)
FP32, VNNI INT8, ...

Intel® Deep Learning Boost

2015
2016
2017
2019
- **Intel® Deep Learning Boost** is a new set of AVX-512 instructions designed to deliver significant, more efficient Deep Learning (Inference) acceleration on second generation **Intel® Xeon® Scalable processor** (codename “Cascade Lake”)
Matrix Multiplies are the foundation of many DL applications
- **Multiply** a \( \text{row} \times \text{column} \) values, **accumulate** into a **single value**

Traditional HPC and many AI training workloads use floating point
- Massive dynamic range of values (FP32 goes up to \( \sim 2^{128} \))

Why INT8 for Inference?
- More power efficient per operation due to smaller multiplies
- Reduces pressure on cache and memory subsystem
- Precision and dynamic range sufficient for many models

What's different about INT8?
- Much smaller dynamic range than FP32: 256 values
- Requires **accumulation into INT32** to avoid overflow (FP handles this “for free” w/ large dynamic range)
FAST EVOLUTION OF AI CAPABILITY ON XEON PLATFORM
**PRECISION FOR DEEP LEARNING**

- Precision is a measure of the detail used for numerical values primarily measured in bits

**FP32 TRAINING** → **BF16 BIT TRAINING**
- Better cache usage
- Avoids bandwidth bottlenecks
- Maximizes compute resources
- Lesser silicon area & power

**NEED** - FP32 WEIGHTS → INT 8 WEIGHTS FOR INFERENCE

**BENEFIT** - REDUCE MODEL SIZE AND ENERGY CONSUMPTION

**CHALLENGE** - POSSIBLE DEGRADATION IN PREDICTIVE PERFORMANCE

**SOLUTION** - QUANTIZATION WITH MINIMAL LOSS OF INFORMATION

**DEPLOYMENT** - OPENVINO TOOLKIT OR DIRECT FRAMEWORK (TF)
**INSTRUCTIONS FUSION**

**VPMADDUBSW**
Multiply and Add Packed Signed and Unsigned Bytes

<table>
<thead>
<tr>
<th>SRC 1 8-bit</th>
<th>A₀</th>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
<th>...</th>
<th>Aₙ₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC 2 8-bit</td>
<td>B₀</td>
<td>B₁</td>
<td>B₂</td>
<td>B₃</td>
<td>...</td>
<td>Bₙ₀</td>
</tr>
<tr>
<td>DEST 16-bit</td>
<td>A₀⁺B₀ + A₁⁺B₁ + A₂⁺B₂, ... + Aₙ₀⁺Bₙ₀</td>
<td>A₀⁺Bₙ₀ + A₁⁺Bₙ₁ + A₂⁺Bₙ₂</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VPMADDWD**
effectively upconvert to 32-bit and horizontal add of neighbors

| SRC 1 16-bit | A₀⁺B₀ + A₁⁺B₁, A₂⁺B₂ + A₃⁺B₃, ... | A₀⁺Bₙ₀ + A₁⁺Bₙ₁ + A₂⁺Bₙ₂ + A₃⁺Bₙ₃ |
| SRC 2 16-bit | 1 | 1 | ... | 1 |
| DEST 32-bit | A₀⁺B₀ + A₁⁺B₁ + A₂⁺B₂ + A₃⁺B₃, ... | A₀⁺Bₙ₀ + A₁⁺Bₙ₁ + A₂⁺Bₙ₂ + A₃⁺Bₙ₃ |

**VPADD**
Add Packed Double-Precision Floating-Point Values

| SRC 1 32-bit | A₀⁺B₀ + A₁⁺B₁ + A₂⁺B₂ + A₃⁺B₃, ... | A₀⁺Bₙ₀ + A₁⁺Bₙ₁ + A₂⁺Bₙ₂ + A₃⁺Bₙ₃ |
| SRC 2 32-bit | C₀ | ... | Cₙ₃ |
| DEST 32-bit | A₀⁺B₀ + A₁⁺B₁ + A₂⁺B₂ + A₃⁺B₃ + C₀ | A₀⁺Bₙ₀ + A₁⁺Bₙ₁ + A₂⁺Bₙ₂ + A₃⁺Bₙ₃ + Cₙ₃ |

**VPDPBUSD**
Multiply and Add Unsigned and Signed Bytes

<table>
<thead>
<tr>
<th>SRC 1 8-bit</th>
<th>A₀</th>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
<th>...</th>
<th>Aₙ₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC 2 8-bit</td>
<td>B₀</td>
<td>B₁</td>
<td>B₂</td>
<td>B₃</td>
<td>...</td>
<td>Bₙ₀</td>
</tr>
<tr>
<td>C₀</td>
<td>...</td>
<td>Cₙ₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VPMADDUBSW + VPMADDWD + VPADD**
fused into **VPDPBUSD** (3x peak OPs)
Generational Performance Projections on Intel® Scalable Processor for Deep Learning Inference for Popular CNNs

<table>
<thead>
<tr>
<th>Platform</th>
<th>Software</th>
<th>ResNet-50</th>
<th>Inception v3</th>
<th>SSD-VGG16</th>
<th>ResNet-50</th>
<th>Inception v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>F</td>
<td>1.5x¹</td>
<td>1.6x³</td>
<td>1.3x¹</td>
<td>1.3x¹</td>
<td>1.3x¹</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>P</td>
<td>1.9x¹</td>
<td>1.8x³</td>
<td>2.0x¹</td>
<td>1.8x³</td>
<td>2.0x¹</td>
</tr>
</tbody>
</table>

¹/24/2018) Results have been estimated 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. 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. 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

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.

*Other names and brands may be claimed as the property of others.
AVX512_VNNI is a new set of AVX-512 instructions to boost Deep Learning performance

- VNNI includes FMA instructions for:
  - 8-bit multiplies with 32-bit accumulates (u8 x s8 ⇒ s32)
  - 16-bit multiplies with 32-bit accumulates (s16 x s16 ⇒ s32)

- Theoretical peak compute gains are:
  - 4x int8 OPS over fp32 OPS and ¼ memory requirements
  - 2x int16 OPS over fp32 OPS and ½ memory requirements

- Ice Lake and future microarchitectures will have AVX512_VNNI
ENABLING INTEL® DL BOOST ON CASCADE LAKE

THEORETICAL IMPROVEMENTS: FP32 VS. INT8 & DL BOOST

UP TO 4X BOOST IN MAC/CYCLE

UP TO 4X IMPROVED PERFORMANCE / WATT

DECREASED MEMORY BANDWIDTH

IMPROVED CACHE PERFORMANCE

UP NEXT: MICROBENCHMARKING WITH INTEL® MKL-DNN'S

Workloads
Topologies
Frameworks
Intel® MKL-DNN Libraries
Intel® Processors

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. 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 www.intel.com/benchmarks.
INTEL DATA-CENTRIC HARDWARE: HIGH PERFORMANCE, FLEXIBLE OPTIONS

- **General Purpose CPU**
  - XEON
- **Programmable Data Parallel Accelerator**
  - Intel® Processor Graphics & Future Products
- **FPGA**
  - Intel® Stratix 10
- **Domain Optimized Accelerator**
  - Intel Neural Network Processor

**GENERAL PURPOSE**
Provide optimal performance over the widest variety of workloads

**HARDWARE**

**WORKLOAD OPTIMIZED**
Deliver highest performance per $/Watt/U/Rack for critical applications

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

Diverse set of data-centric hardware

No common programming language or APIs

Inconsistent tool support across platforms

Each platform requires unique software investment

SVMS
**INTEL’S ONEAPI CORE CONCEPT**

**oneAPI** is a project to deliver a unified programming model to simplify development across diverse architectures.

- Common developer experience across Scalar, Vector, Matrix and Spatial (SVMS) architecture
- Unified and simplified language and libraries for expressing parallelism
- Uncompromised native high-level language performance
- Support for CPU, GPU, AI and FPGA
- Based on industry standards and open specifications

---

Optimization Notice

Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.
ONEAPI FOR CROSS-ARCHITECTURE PERFORMANCE

Optimized Applications

Optimized Middleware & Frameworks

oneAPI Product

Direct Programming

API-Based Programming

Porting Tool

Data Parallel C++

Libraries

Analysis & Debug Tools

oneAPI Product

CPU

SCALAR

GPU

VECTOR

AI

MATRIX

FPGA

SPATIAL

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Legal Disclaimer & Optimization Notice

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 [www.intel.com/benchmarks](http://www.intel.com/benchmarks).

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.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, the Intel logo, Pentium, Xeon, Core, VTune, OpenVINO, Cilk, are trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries.

**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
**Quantization**

**Post Training Quantization**
Train normally, capture FP32 weights; convert to low precision before running inference and calibrate to improve accuracy.

**Quantization Aware Training**
Simulate the effect of quantization in the forward and backward passes using FAKE quantization.

---

**Reduce Precision & Keep Model Accuracy**

**FP32 Weights Training** → **8 Bit Inference**
Collect statistics for the activation to find quantization factor. Calibrate using subset data to improve accuracy, Convert FP32 weights to INT8.

**FP32 Weights Training** → **8 Bit Inference**
Captured FP32 weights are quantized to int8 at each iteration after the weight updates. Convert FP32 weights to INT8.
DATA PARALLEL C++
STANDARDS-BASED, CROSS-ARCHITECTURE LANGUAGE

Language to deliver uncompromised parallel programming productivity and performance across CPUs and accelerators

- Allows code reuse across hardware targets, while permitting custom tuning for a specific accelerator

Based on C++

- Delivers C++ productivity benefits, using common and familiar C and C++ constructs
- Incorporates SYCL* from the Khronos* Group to support data parallelism and heterogeneous programming

Language enhancements being driven through community project

- Extensions to simplify data parallel programming
- Open and cooperative development for continued evolution

Builds upon Intel’s years of experience in architecture and compilers
POWERFUL LIBRARIES FOR DATA-CENTRIC FUNCTIONS

Key domain-specific functions to accelerate compute intensive workloads

Custom-coded for uncompromised performance on SVMS (Scalar, Vector, Matrix, Spatial) architectures
ADVANCED ANALYSIS & DEBUG TOOLS

Productive performance analysis across SVMS architectures

Intel® VTune™ Profiler
Profiler to analyze CPU and accelerator performance of compute, threading, memory, storage, and more

Intel® Advisor
Design assistant to provide advice on offload, threading, and vectorization

Debugger
Application debugger for fast code debug on CPUs and accelerators
Facilitate addressing multiple hardware choices through a modern language like DPC++