UNDERSTANDING THE NEW VECTOR NEURAL NETWORK INSTRUCTIONS

Evarist Fomenko
Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)
05/24/2018
OUTLINE

• Reduced precision inference
• VNNI instructions
• 8-bit convolution implementation
OUTLINE

• Reduced precision inference
• VNNI instructions
• 8-bit convolution implementation
**REDUCED PRECISION INFERENCE**

Data types for CNNs:
- Training: fp32, fp16, bfloat16, int16, ...
- Inference: fp32, fp16, int8, ...

**Int8 vs fp32:**
- Better performance (instruction throughput)
- Lower memory consumption (higher bandwidth, better cache usage)
- Acceptable accuracy loss
PERFORMANCE INT8 VS FP32

For Intel® processors with AVX512_BW support

- Standalone convolutions: ~1.5x (see next slide)
- Topology level

<table>
<thead>
<tr>
<th>Topology</th>
<th>Batch size</th>
<th>FP32 img/s</th>
<th>INT8 img/s</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>448</td>
<td>368</td>
<td>517</td>
<td>1.40x</td>
</tr>
<tr>
<td>SSD/VGG16</td>
<td>448</td>
<td>29</td>
<td>46</td>
<td>1.59x</td>
</tr>
</tbody>
</table>

Measured with Intel® Optimization for Caffe* (source: https://github.com/intel/caffe/wiki/Introduction-of-Accuracy-Calibration-Tool-for-8-Bit-Inference)

With new VNNI instructions the performance is expected to be higher

*Other names and brands may be claimed as the property of others.
**INT8 INFEERENCE TECHNIQUES**

Int8 has lower dynamic range compare to fp32

Quantization:
- \( v \leftarrow \alpha \cdot i_8 + \beta \) (biased)
- \( v \leftarrow \alpha \cdot i_8 \) (symmetric)

Typically there is a single scale \( \alpha \sim \frac{\max(|v|)}{128} \) per tensor \( \{v\} \).
## INT8 CONVOLUTION

### Assumptions:
- Symmetric quantization
- Scales are precomputed
- Single scale per tensor

### Computations:
1. \( \gamma \leftarrow \frac{\text{src_scale} \cdot \text{w_scale}}{\text{dst_scale}} \)
2. \( \text{dst}_{i8} \leftarrow \gamma \cdot \text{conv}_{i32}(\text{src}_{i8}, \text{weights}_{i8}) \)
OUTLINE

• Reduced precision inference
• VNNI instructions
• 8-bit convolution implementation
THE GOAL

\[ \text{conv}_{i32}(\text{src}_{i8}, \text{weights}_{i8}) \]

Key component: int8-multiplication with int32 accumulation
**U8/S8 TO S32 MAC**

for $i = 0 .. 15$

for $j = 0 .. 3$

$$\text{dst}_{32}[i] += \text{src}_{1u8}[4i + j] \times \text{src}_{2s8}[4i + j]$$

<table>
<thead>
<tr>
<th>AVX-512 BW</th>
<th>VNNI</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpmaddubsw $zmm_{s16}$, $src_{1u8}$, $src_{2s8}$</td>
<td>vpdpbusd $dst_{s32}$, $src_{1u8}$, $src_{2s8}$</td>
</tr>
<tr>
<td>vpmaddwd $zmm_{s16}$, $zmm_{s16}$, $[16 \times 1_{16}]$</td>
<td>$zmm_{s16}$, $zmm_{s16}$, $zmm_{s16}$</td>
</tr>
<tr>
<td>vpaddd $dst_{s32}$, $dst_{s32}$, $zmm_{s16}$</td>
<td>$dst_{s32}$, $dst_{s32}$, $zmm_{s16}$</td>
</tr>
</tbody>
</table>

- 64 multiply-adds in 3 instructions
- **potential intermediate saturation after** vpmaddubsw
- saturate<s16>(u8*s8 + u8*s8)
- **1.33x** throughput of F32 equivalent (4x vfmadd231ps)

- 64 multiply-adds in 1 instruction
- **no** potential intermediate saturation
- src2 (s8) operand can be broadcasted from memory
- **4x** throughput of F32 equivalent (4x vfmadd231ps)
OUTLINE

• Reduced precision inference
• VNNI instructions
• 8-bit convolution implementation
THE GOAL

\[ \text{conv}_{s32}(\text{src}_{u8}, \text{weights}_{s8}) \]

- \text{unsigned int8}
- \text{signed int8}
CONVOLUTION AND LAYOUT

Naïve layout: nhwc / hwio

\[
\begin{align*}
src_{u8} & \colon [mb][ih][iw][ic] \\
weis_{s8} & \colon [kh][kw][ic][oc] \\
dst_{s32} & \colon [mb][oh][ow] \ [oc]
\end{align*}
\]

Optimized layout: nhwc / hwIo4i

\[
\begin{align*}
src_{u8} & \colon [mb][ih][iw][ic/4] \quad [4ic] \\
weis_{s8} & \colon [kh][kw][ic/4][oc/16][16oc][4ic] \\
dst_{s32} & \colon [mb][oh][ow] \quad [oc/16][16oc]
\end{align*}
\]
**Optimized Convolution**

Optimized layout: nhwc / hwIo4i

- **src** \( u8 \) \( [mb][ih][iw][ic/4] \) [4ic] \( [4ic] \)
- **wei** \( s8 \) \( [kh][kw][ic/4][oc/16][16oc][4ic] \)

---

- **dst** \( s32 \) \( [mb][oh][ow] \) [oc/16][16oc] [4ic]

Innermost loop:

- \( 16 \times [4ic] \) \( \leftarrow \) **vpbroadcastd** \( s[4ic] \)
- \( dst_{532} \) \( \leftarrow \) **vpdpbusd** \( 16 \times [4ic], [16oc][4ic] \)

Parallelization: \( mb, oh \)
IMPLEMENTATION IN INTEL MKL-DNN

Optimizations that are not shown:
• Even more blocking (weights format is OIhw4i16o4i)
• More parallelism (mb, oc/16, oh)
• Reuse sources and weights
• And more...

Check https://github.com/intel/mkl-dnn:
src/cpu/jit_avx512_core_u8s8s32x_conv*.cpp
LIMITATIONS

Intel MKL-DNN supports:

\[
\text{\(dst_{dt} \leftarrow \gamma \cdot \text{conv}_{s32}(src_{u8}, weights_{s8}), dt \in \{f_{32}, s_{32}, s_{8}, u_{8}\}\)}
\]

Source data must be non-negative

- Convolution after ReLU
  - all but first in AlexNet, ResNet-50, Inception-v3
- (Optional) implementation w/ compensation: \(src_{s8} \rightarrow src'_{u8} - 128\)

Number of channels must be a multiple of 16

- Quite common
  - all but first in AlexNet, ResNet-50, Inception-v3
- (Optional) implementation w/ padding
Convolutions Performance in ResNet-50 Batch 50

F32 and INT8 convolutions amb. Intel MKL-DNN benchdnn
Intel® Xeon® Platinum 8180, 28 cores @2.5 GHz

Benchmark results 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. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Configuration: Intel® Xeon® Platinum 8180 CPU @2.50 GHz (1 socket, 28 cores), HT disabled, Turbo enabled, 96 GB DDR4-2666 RAM, Red Hat Enterprise Linux Server release 7.2 (3.10.0-327.62.4.el7.x86_64). Intel MKL-DNN version v0.14. Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=28. Performance measured with benchdnn --conv --mode=PERF --cfg=f32 ${prb} --cfg=u8s8u8s32 ${prb}.

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.

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 or estimated as of April 2018.
BACKUP
NEW VNNI INSTRUCTIONS (CONT’D)

VPDPBUSD(S)

vpdpbusd zmm0, zmm1, zmm2/m512/m32bcst
s_{32} \leftarrow s_{32} + \sum_{0}^{3} u_{8} \cdot s_{8}

vpdpbusds zmm0, zmm1, zmm2/m512/m32bcst
s_{32} \leftarrow \text{saturate}<s_{32}>(s_{32} + \sum_{0}^{3} u_{8} \cdot s_{8})

VPDPWSSD(S)

vpdpwssd zmm0, zmm1, zmm2/m512/m32bcst
s_{32} \leftarrow s_{32} + \sum_{0}^{1} s_{16} \cdot s_{16}

vpdpwssds zmm0, zmm1, zmm2/m512/m32bcst
s_{32} \leftarrow \text{saturate}<s_{32}>(s_{32} + \sum_{0}^{1} s_{16} \cdot s_{16})