人工智能丨低数值精度深度学习推理与训练

原创：IDZ英特尔开发人员专区9 月 14 日

如今，多数商用深度学习应用在训练与推理工作负载中使用 32 位浮点精度（ƒp32)。各领域的研究人员展示了能够以较低的数值精度执行深度学习训练与推理，使用 16 位乘法器进行训练，8 位或更低的乘法器进行推理，准确性的损失降至最低，甚至为零（训练过程中通常需要较高的精度 - 16 位和 8 位，以准确表示反向传播阶段的梯度）。使用这些低数值精度（训练时将 16 位乘法器累加至 32 位或更高，推理时将 8 位乘法器累加至 32 位）有可能成为明年的标准，尤其对于卷积神经网络（CNN)。

较低的精度会带来两大优势。首先，许多操作受到内存带宽的限制，降低精度将支持更好地使用高速缓存，减少带宽瓶颈。因此，可以更快速地在内存层级中传输数据，以最大限度地利用计算资源。其次，由于这些乘法器需要更少的芯片面积和功率，硬件可能以较低的精度支持更高的每秒操作数（OPS)。

在本文中，我们回顾了低精度训练与推理的历史，介绍了英特尔® 如何支持以较低的数值精度执行深度学习推理与训练。我们专门介绍了当前英特尔® 至强® 可扩展处理器中已有的指令，以及未来某些处理器将提供的指令。我们介绍了如何量化模型权重和激活，还介绍了面向深度神经网络的英特尔® 数学核心函数库（英特尔® MKL-DNN）提供的低

数值精度函数。最后，我们介绍了深度学习框架如何利用这些低数值精度函数，以及如何降低不同数值精度之间的转换开销。每个章节独立于其他章节，读者可以直接翻阅自己感兴趣的部分。此文详细阐述了用于深度学习训练的英特尔至强可扩展处理器，包括商业用例。

深度学习低精度发展简史

研究人员展示了使用 16 位乘法器的深度学习训练，它的推理采用了 8 位乘法器，或将更少的数值精度累加为高精度，各种模型上的准确性损失降至最低，甚至为零。Vanhoucke 等人（2011 年）将激活与权重量化为 8 位，并且面向 CPU 上的语音识别任务将偏差和第一层输入设置为全精度 (fp32)。Hwang 等人（2014 年）使用 -1、0 和 1 的量化权重在正向传播中训练了一个简单的网络，并使用 MNIST* 和 TIMIT* 数据集在反向传播中更新高精度权重，性能损失可以忽略不计。Courbariaux 等人（2015 年）使用 MNIST、CIFAR-10* 和 SVHN* 数据集借助低数值精度乘法器和高精度累加器进行训练，并更新了高精度权重。他们建议将动态定点（拥有一个张量或高维度阵列共享指数）和 Gupta 等人（2015 年）的随机舍入结合在一起，作为今后的研究方向。这成为 Koster 等人（2017 年）应用于英特尔® Nervana™ 神经网络处理器中的 Flexpoint 数值格式的核心部分。Kim 和 Smaragdis（2016 年）使用二进制权重进行训练，并更新了全精度权重，为 MNIST 数据集提供了极具竞争力的性能。Miyashita 等人（2016 年）在以 2 为底的对数表达式中对权重和激活进行了编码（由于权重/激活呈非均匀分布）。他们使用 5 位权重和 4 位激活对 CIFAR-10 进行了训练，最

大程度上减少了性能下降。Rastegari 等人（2016 年）使用二进制权重对 AlexNet 进行了训练（除了第一层和最后一层），并在全精度权重上进行了更新，精度损失为 2.9%，位列第一。根据实验结果，他们建议避免在拥有小通道或过滤器尺寸（如 1x1 内核）的完全连接层和卷积层中进行二值化。来自英特尔研究院的 Mellempudi 等人（2017 年）在卷积层中使用 4 位权重和 8 位激活对 ResNet-101 进行了训练，同时以全精度进行更新，精度损失为 2%，位列第一。Micikevicius 等人（2017 年）使用 16 位浮点（fp16）乘法器和全精度累加器进行训练，并且更新了全精度权重，AlexNet*、VGG-D*、GoogLeNet*、ResNet-50*、Faster R-CNN*、Multibox SSD*、DeepSpeech2*、Sequence-to-Sequence*、bigLSTM* 和 DCGAN*（某些模型要求梯度缩放，以匹配全精度结果）的精度损失可以忽略不计，甚至为零。百度研究人员（2017 年）成功使用 8 位固定精度，包含一个符号位，4 个整数部分位和 3 个小数部分位。Sze 等人（2017 年）使用了各种量化技术（参阅他们文章中的表 3），将低精度损失降至最低或零（除了第一层和最后一层保持全精度）。在 ICLR 2018 上匿名提交的一篇论文详细介绍了如何使用 16 位整数乘法器和 32 位累加器在 ResNet-50、GoogLeNet、VGG-16 和 AlexNet 上生成一流的低精度。

英特尔®至强®可扩展处理器

上的低数值精度

英特尔至强可扩展处理器目前包括英特尔® 高级矢量扩展指令集 512（英特尔® AVX-512），后者具有 512 位宽融合乘加（FMA）内核指

令。这些指令支持低数值精度乘法和高精度累加。将两个 8 位值相乘，并将结果累加至 32 位需要 3 条指令，其中一个 8 位矢量必须是无符号 int8 (u8)格式，另一个必须是带符号 int8 (s8)格式，在带符号 int32 (s32) 格式中进行累加。这使输入量增加了 4 倍，代价是指令增加 3 倍或计算量增加 33.33%，内存要求降至 1/4。内存的减少和频率的提高加快了低数值精度运算的速度，详细信息如图 1 所示。

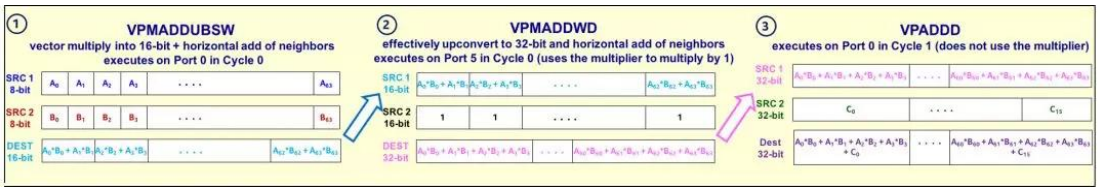

图 1.英特尔至强可扩展处理器内核可使用 3 条指令执行 8 位乘法和 32 位累加：VPMADDUBSW υ 8 × s8 → s16 multiple 、VPMADDWD broadcast1 s16→s32 和 VPADDD s32→s32 将结果添加至累加器。这使 fp32 上的输入量增加了 4 倍，代价是指令增加 3 倍或计算量增加 33.33%，内存要求降至 1/4。内存的减少和频率的提高加快了低数值精度运算的速度。图片由来自以色列的 Hirsh 提供。

英特尔 AVX-512 指令还支持 16 位乘法。将两个 16 位值相乘，并将结果累加至 32 位需要两条指令（两个周期），其中 16 位矢量为带符号 int16 (s16) 格式，累加为带符号 int32 (s32) 格式。这使输入量增加了两倍，代价是指令增加两倍，未产生额外的计算。但是，它减少了内存要求和带宽瓶颈，这两者可能会提升整体性能。请参阅图 2 了解详情。

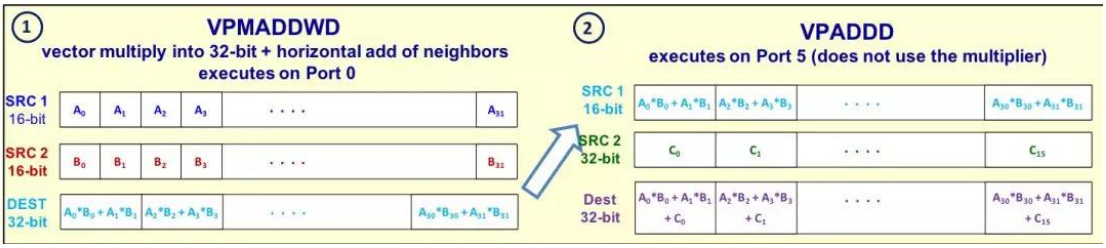

图 2.英特尔至强可扩展处理器内核能够使用两条指令执行 16 位乘法和 32 位累加：VPMADDWD s16×s16→s32 multiple 和 VPADDD s32→s32 将结果添加至累加器。这使 fp32 上的输入量增加了两倍，代价是指令增加两倍，或者不需要更多的计算，内存要求降至 ½。图片由来自以色列的 Hirsh 提供。

英特尔开发了一款全新英特尔 AVX-512 指令集 - AVX512_VNNI（矢量神经网络指令），以提升 DL 性能。Ice Lake 等未来的微架构（见表 1-1）将使用 AVX512_VNNI 指令。AVX512_VNNI 包括 1) 一个 FMA 指令，用于 8 位乘法和 32 位累加 υ8×s8→s32，如图 3 所示，2) 一个 FMA 指令，用于 16 位乘法与 32 位累加 s16×s16→s32，如图 4 所示。ƒp32 OPS 的理论峰值计算量是 int8 OPS 的 4 倍， int16 OPS 的两倍。由于存在内存带宽瓶颈，实际的提升可能会更低。

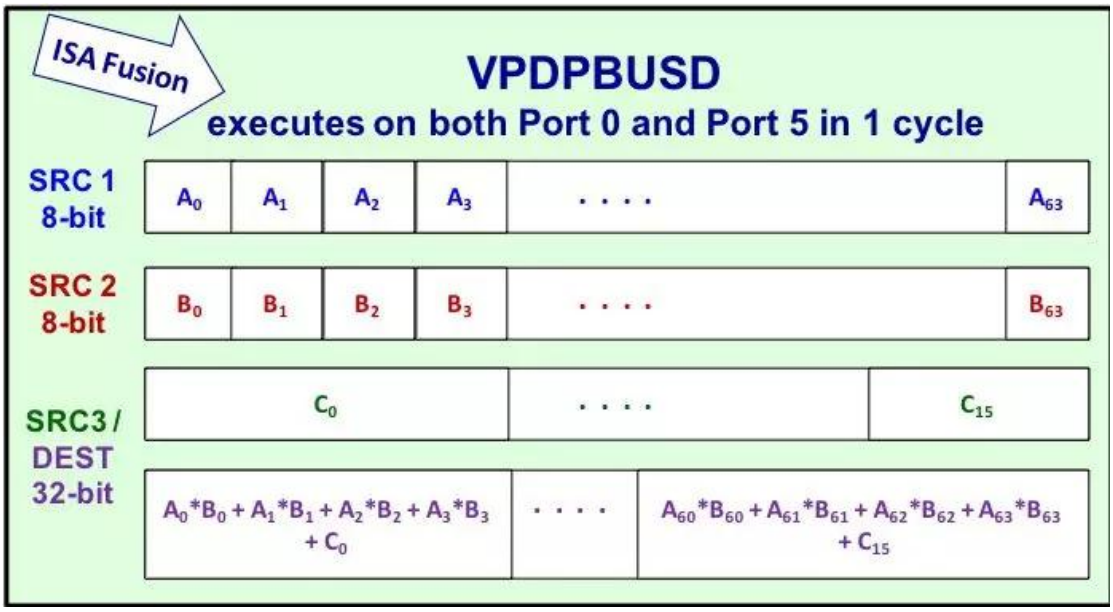

图 3.AVX512_VNNI 支持使用 1 条指令执行 8 位乘法与 32 位累加。图 1 中的 VPMADDUBSW、VPMADDWD、VPADDD 指令被融合成 VPDPBUSD 指令 υ8×s8→s32。这使 ƒp32 的输入量增加了 4 倍，（理论峰值）计算增加了 4 倍，内存要求降至 ¼。图片由来自以色列的 Hirsh 提供。

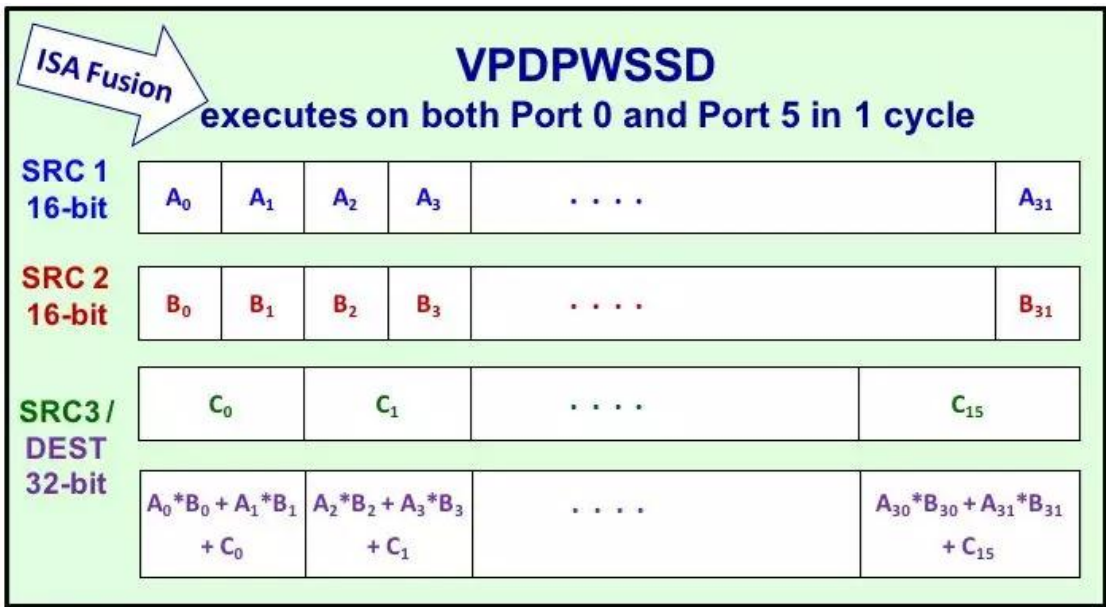

图 4.AVX512_VNNI 支持使用一条指令执行 16 位乘法与 32 位累加。

图 2 中的 VPMADDWD、VPADDD 指令被融合至 VPDPWSSD 指令 s16×s16→s32。这使 ƒp32 上的输入量增加了两倍，（理论峰值）计算增加了两倍，内存要求降至 ½。图片由来自以色列的 Hirsh 提供。

溢出的未定义行为是一个潜在的问题，可能在使用 VPMADDUBSW 指令 υ8×s8→s16（见图 1）时出现。当 υ8 和 s8 的值均接近最大值时，该问题产生。2 可以通过将输入精度降低 1 位来解决该问题。在使用 AVX512_VNNI VPDPBUSD FMA 指令 υ8×s8→s32 时，实际上不存在这种问题。

溢出更有可能发生在 AVX512_VNNI VPDPWSSD FMA 指令 s16×s16→s32 中。同样可以通过将激活和权重的精度降低 1 或 2 位来解决该问题。防止溢出的另一个技巧是使用 ƒp32 上的第二个累加器，然后转换到 ƒp32，在达到设定的 s32 累加数量后使用该累加器。初步结果显示使用这些技巧并不会影响统计性能。

目前正开发面向 AVX512_VNNI 指令的编译器支持。GCC 8 开发代码和 LLVM/Clang 6.0 编译器均支持 AVX512_VNNI 指令。X86 编码器解码器 (XED) 和英特尔软件开发仿真器 (SDE) 2017 年 10 月的更新添加了面向 AVX512_VNNI 指令的支持。

英特尔® MKL-DNN 库

低数值精度基元

英特尔 MKL-DNN 库包含常见的深度学习函数和基元，它们被广泛应用于各种模型，如内积、卷积、修正线性单元 (ReLU) 和批处理标准化 (BN)，以及控制张量或高维度阵列布局所需的函数。英特尔 MKL-DNN 专为包含英特尔 AVX-512、英特尔® AVX-2 和英特尔® SIMD 流指令扩展 4.2（英特尔® SSE4.2）指令集的英特尔处理器而优化。这些函数使用 ƒp32 运行训练与推理工作负载。最近推出了新功能，以支持卷积、ReLU、融合卷积与 ReLU 和池化层中 8 位精度的推理工作负载。长短期记忆 (LSTM) 函数、其他融合运算和 8 位 Winograd 卷积被视作未来的发展趋势。在指令可用时，英特尔 MKL-DNN 对使用 AVX512_VNNI 指令的 16 位乘法的支持被视为未来的发展趋势。

目前，英特尔 MKL-DNN 不包括由 8 位精度（仅 fp32）实施的局部响应归一化 (LRN)、完全连接 (FC)、softmax 或批处理标准化 (BN) 层，原因如下。现代模型不使用 LRN，旧模型经过修改后，可以使用批处理标准化。现代 CNN 模型通常没有太多 FC 层，尽管添加 FC 层支持被视为未来的发展趋势。softmax 函数目前要求全精度，这是因为 8 位精度无法保持它的准确性。如框架章节支持低数值精度部分所述，不需要 BN 推理层，因为它能通过缩放权重值或修改偏差被上一层吸收。

英特尔 MKL-DNN 实施 8 位卷积运算时，激活（或输入）值为 υ8 格式，权重为 s8 格式，偏差为 s32 格式（偏差可以保存在 ƒp32，它们仅

占总体计算的一小部分）。图 5 显示了将 8 位乘法器累加至 s32 的推理运算过程。

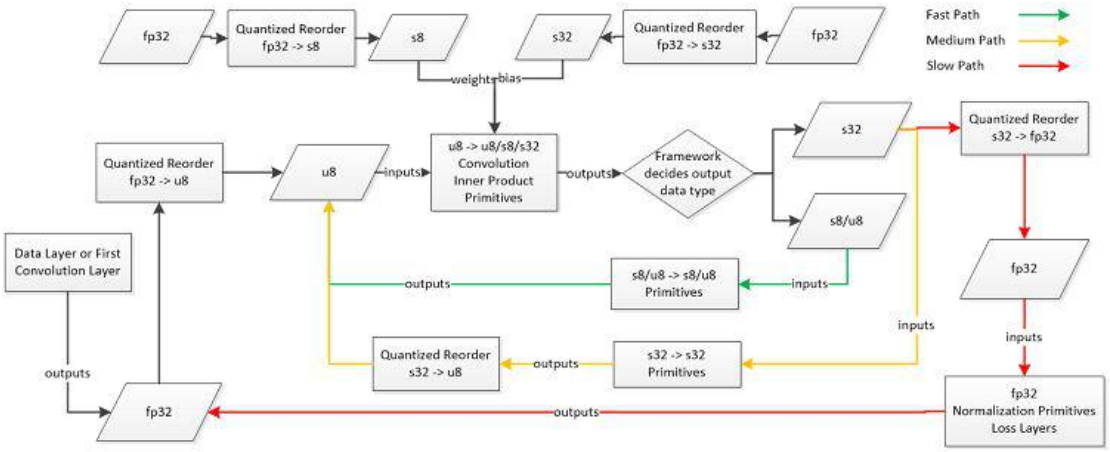

图 5.数据层或第一个卷积层的激活被量化为 υ8 ,作为下一个卷积层的输入。权重被量化为 s8，偏差被格式化为 s32，并添加至 s32 卷积累加。根据下一层的参数，框架选择 s8、υ8 或 s32 作为卷积输出的格式，图片由 Jiong Gong 提供。

对包含非负值的激活和权重执行 8 位量化

英特尔 MKL-DNN 现在假设激活是非负值，这是 ReLU 激活函数之后的情况。我们将在后文讨论如何使用负值量化激活。英特尔 MKL-DNN 按照以下方法量化特定张量或张量内每个通道的值（由框架开发人员做出选择）。

R{a,w}= max (abs(T{a,w}))，此处，T{a,w} 是与权重 w、激活或模型输入 a 相对应的张量。

Qa = 255/Ra 是负值激活的量化因子，Qw = 127/Rw 是权重的量化因子。量化激活、权重和偏差为：

$$\boldsymbol{a}_{u8} = \|Q_a \boldsymbol{a}_{f32}\| \in [0,255]$$

$$\boldsymbol{W}_{s8} = \|Q_w \boldsymbol{W}_{f32}\| \in [-127,127]$$

$$\boldsymbol{b}_{s32} = \|Q_a Q_w \boldsymbol{b}_{f32}\| \in [-2^{31}, 2^{31}-1]$$

此处，函数 $\|\cdot\|$ 被四舍五入为最接近的整数。请注意，尽管 s8 格式支持 -128，但是权重值使用的最小量化 s8 为 -127。

使用 8 位乘法器和 32 位累加器的仿射变换得出

$$\boldsymbol{x}_{s32} = \boldsymbol{W}_{s8}\boldsymbol{a}_{u8} + \boldsymbol{b}_{s32} \approx Q_a Q_w\left(\boldsymbol{W}_{f32}\boldsymbol{a}_{f32} + \boldsymbol{b}_{f32}\right) = Q_a Q_w \boldsymbol{x}_{f32}$$

出现近似值是因为等式忽略了取整运算，以及

$$\boldsymbol{x}_{f32} = \boldsymbol{W}_{f32}\boldsymbol{a}_{f32} + \boldsymbol{b}_{f32} \approx \frac{1}{Q_a Q_w}\boldsymbol{x}_{s32} = D\boldsymbol{x}_{s32}$$

是 f32 格式的仿射变换，D = 1/Qa Qw 是反量化因子。

在量化为 υ8 和 s8 格式的过程中，零值映射为特定值，无需四舍五入。鉴于零是最常见的一个值，最好通过准确的映射来减少量化误差和提升统计的准确性。

以上量化因子可以在英特尔至强可扩展处理器中采取 fp32 格式。但是，某些架构不支持除法（如 FPGA）和使用位移。对于这些架构，标量被四舍五入为最接近的二次方，通过位移来进行缩放。统计准确性的损失降至最低（通常 <1%）。

高效的 8 位乘法

图 6 展示了如何有效地执行 A × W 的 8 位乘法。英特尔 MKL-DNN 使用面向激活张量的 N H W C 数据布局，此处，N 为批量大小，H 为高度，W 为宽度，C 为通道数量。还使用了面向权重张量的 (O/16)

K (C/4)T16o4c 的数据布局，此处，O 为内核或输出通道数量，C 为输入通道数量，K 为高度，T 为宽度。张量 A 的前 32 位（4 个 int8 值，显示为灰色）被广播了 16 次，以填充 512 位寄存器。量化权重后，英特尔 MKL-DNN 对张量 W 的数据布局进行修改。张量 W 数据布局被重新布置为 W′，每组 16 列，每列拥有 32 位（4 个 int8 值），在内存中被持续读取，首先，第 1 列的前 4 个值占用寄存器（红色）的前 32 位，后 4x1 占用寄存器（橙色）的后 32 位，然后继续下去（绿色）。以相同的模式重新布置第一个数据块下的第二、第三和第四个数据块（黄色）。然后处理下一组数据块（蓝色）。在实践中，通常在重新布置内存布局前对张量 W 进行转置，以访问 1x4 连续内存值，而不是在重新布置数据布局时访问 4x1 散射值。修改该数据布局通常是一次性完成的，然后存储起来，以供所有推理迭代重复利用。

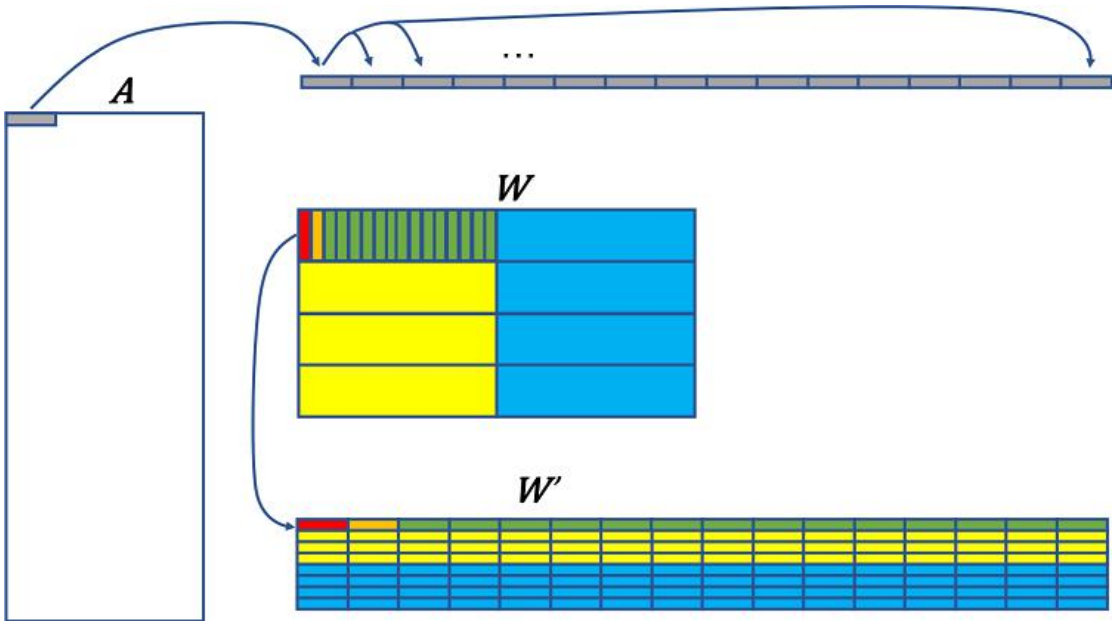

图 6.有效使用 int8 乘法来计算 A×W 的积需要转换张量 W 的数据布局，以读取连续位。A 的 32 位组被广播了 16 次，以填充 512 位寄存器，然后乘以来自张量 W′ 的 512 位组。

在寄存器中，A 的前 4 个 int8 值（复制 16 次）乘以 W′ 的 64 个 int8 值（512 位），然后累加。A 的后 4 个值被广播至另一个寄存器，共广播了 16 次，然后乘以 W′ 的后 64 个 int8 值。在 A 的第一行被读取以及结果被累加前，它将持续运行。输出（在 8 位 FMA 的所有 3 条指令之后）是前 16 个输出值（需要 s32 上的 512 位）。然后将第一行 A 乘以 W′ 后面的值，将获得后 16 个输出值。

英特尔至强可扩展处理器拥有多达 32 个寄存器。在包含两个 FMA 单元的处理器上执行 512 位寄存器端口方案时，3 端口 0 FMA 的延迟为 4 个周期，端口 5 FMA 的延迟为 6 个周期。int8 上用于深度学习工作负载的指令支持绕过端口 0 和 5，在这两个端口上的延迟为 5 个周期（详见第 15.17 节）。在实践中，W′ 的多个行被加载至多个寄存器，以隐藏这些延迟。

用于训练的 16 位函数

在指令可用时，英特尔 MKL-DNN 借助 AVX512_VNNI 指令为 16 位乘法提供支持将成为今后的研究方向。尽管如此，研究人员已经 展示了如何使用 16 位乘法和 32 位累加训练各种 CNN 模型，需要执行 AVX512_4VNNI 指令（也被称作 QVNNI，在部分英特尔® 至强® 融核™ 处理器中提供），尤其是 AVX512_4VNNI VP4DPWSSD 指令（类似于之前介绍的 AVX512_VNNI VPDPWSSD 指令）。

这些研究人员获得了 ResNet-50、GoogLeNet-v1、VGG-16 和 AlexNet 的 fp32 统计性能，使用了与 ƒp32 模型相同的迭代次数，未更改超参数。他们使用 s16 存储激活、权重和梯度，并保存 ƒp32 权重的主副本，以便在每次迭代后，对返回 s16 的量化权重进行更新。他们使用的量化因子是 2 次方，通过张量乘法管理量化/反量化因子。

在框架中启用较低数值精度

这款流行的框架支持用户定义他们的模型，无需亲自编写所有函数定义。对框架用户来说，各种函数的实施详情可能较为隐蔽。这些实施由框架开发人员完成。该章节介绍了如何在框架层面进行修改，以启用低数值精度

推理开始前，量化权重已经完成。高效量化激活需要预计算权重因子。对激活量化因子进行预计算，通常通过采样验证数据集来找到上述范围。该范围以外的测试数据集的值已达到饱和。对于负激活值，饱和前的范围被放宽到−128Ra′/127，以使用 s8 = −128 值，此处，Ra′ 是这些激活的最大绝对值。然后将这些标量写入文件。

对带负值的激活或输入执行 8 位量化

可以在框架层面量化带负值的激活或输入值，如下所示：Qa′ = 127/Ra′ 是带负值的激活的量化因子。s8 量化格式为 as8 = ||Qa′ aƒ32 || ∈ [-128, 127]，此处，函数 ||· || 被四舍五入为最接近的整数。但是，激活必须为 υ8 格式，以利用 AVX512_VNNI VPMADDUBSW 指令或 AVX512_VNNI VPDPBUSD 指令（英特尔至

强可扩展处理器上的低数值精度章节对这两条指令进行了说明）。因此，K = 128 将 as8 中的所有值转换为非负值：

$$\boldsymbol{a}_{u8} = \boldsymbol{a}_{s8} + K\mathbf{1} \in [0, 255]$$

此处，1 为所有 1 的矢量，偏差 bf32 被修改为

$$\boldsymbol{b}'_{f32} = \boldsymbol{b}_{f32} - \frac{K}{Q'_a}\boldsymbol{W}_{f32}\mathbf{1}$$

量化权重和修改的偏差的方法和以前一样：

$$\boldsymbol{a}_{u8} = \|Q_a \boldsymbol{a}_{f32}\| \in [0,255]$$

$$\boldsymbol{W}_{s8} = \|Q_w \boldsymbol{W}_{f32}\| \in [-127,127]$$

$$\boldsymbol{b}_{s32} = \|Q_a Q_w \boldsymbol{b}_{f32}\| \in [-2^{31}, 2^{31} - 1]$$

使用 8 位乘法器和 32 位累加器的仿射变换得出

$$\boldsymbol{x}_{s32} = \boldsymbol{W}_{s8}\boldsymbol{a}_{u8} + \boldsymbol{b}'_{s32} \approx Q_w \boldsymbol{W}_{f32}\left(Q_{a'}\boldsymbol{a}_{f32} + K\mathbf{1}\right) + Q_w Q_{a'}\left(\boldsymbol{b}_{f32} - \frac{K}{Q'_a}\boldsymbol{W}_{f32}\mathbf{1}\right) = Q_{a'}Q_w\left(\boldsymbol{W}_{f32}\boldsymbol{a}_{f32} + \boldsymbol{b}_{f32}\right) = Q_{a'}Q_w\boldsymbol{x}_{f32}$$

此处

$$\boldsymbol{x}_{f32} = \boldsymbol{W}_{f32}\boldsymbol{a}_{f32} + \boldsymbol{b}_{f32} \approx \frac{1}{Q_{a'}Q_w}\boldsymbol{x}_{s32} = D\boldsymbol{x}_{s32}$$

此处，D = 1/Qa′ Qw 是反量化因子。

当输入信号为 υ8 格式（如 RGB 图像），但是需要通过预处理步骤减去平均信号时，可以使用上述等式，此处，K 为平均值，aυ8 为输入信号（非预处理），Qa′ = 1。

研究人员通常将第一个卷积层保存为 fp32 格式，将其他卷积层保存为 int8 格式（相关示例详见深度学习低精度发展简史章节）。我们发现

这些量化技术支持使用所有 int8 格式的卷积层，不会明显降低统计准确性。

综上所述，为了使用带负值的激活，激活被量化为 s8 格式，然后使用 K=128 将其转换为 υ8 格式。唯一增加的步骤是修改偏差：

$$b'_{f32'} = b_{f32} - {}^{K}/_{Q'a}\, W_{f32}\, 1$$

对于卷积层，对乘积 Wf32 1 进行概化，使它等于所有 Wf32 值的总和，包含除了与 bf32 共享的尺寸之外的所有尺寸。详情请参见附录 A。

融合量化

融合量化通过如下方式将反量化与量化结合在一起，提升了性能，因此，无需转换为 fp32。l+1 层的激活为：

$$\boldsymbol{a}_{f32}^{(l+1)} = g\left(\boldsymbol{x}_{f32}^{(l)}\right) = g\left(D^{(l)}\boldsymbol{x}_{s32}^{(l)}\right)$$

此处 g(·)为非线性激活函数。假设是 ReLU 激活函数，激活可以表示为 υ8 格式

$$\boldsymbol{a}_{u8}^{(l+1)} = \left\| Q_a^{(l+1)}\boldsymbol{a}_{f32}^{(l+1)} \right\| = \left\| Q_a^{(l+1)} D^{(l)} \max\left(0, \boldsymbol{x}_{s32}^{(l)}\right) \right\|$$

此处，乘积 Qa(l+1) D(l) 支持以 υ8 格式计算下一层的量化激活，无需计算 fp32 表示法。

当 g(·) 为 ReLU 函数 (如以下等式) 并且 Q ≥ 0 时，以下属性成立：

$$Qg\left(D^{(l)}\boldsymbol{x}_{s32}^{(l)} + D^{(h)}\boldsymbol{x}_{s32}^{(h)}\right) = g\left(QD^{(l)}\boldsymbol{x}_{s32}^{(l)} + QD^{(h)}\boldsymbol{x}_{s32}^{(h)}\right)$$

该属性对包含跳跃连接的模型非常实用，如 ResNet，此处，跳跃连接分支可能依赖各种激活。例如，使用 ResNet-50 作者在

Caffe deploy.prototxt 中创造的术语（见图 7），res2b_branch2a 层的量化输入激活（在以下等式中简称为 2b2a）为

$$\begin{aligned} \boldsymbol{a}_{u8}^{(2b2a)} &= Q_a^{(2b2a)} g\left(D^{(2a1)}\boldsymbol{x}_{s32}^{(2a1)} + D^{(2a2c)}\boldsymbol{x}_{s32}^{(2a2c)}\right) \\ &= g\left(Q_a^{(2b2a)}D^{(2a1)}\boldsymbol{x}_{s32}^{(2a1)} + Q_a^{(2b2a)}D^{(2a2c)}\boldsymbol{x}_{s32}^{(2a2c)}\right) \end{aligned}$$

此处，a υ 8(2b2a) ∈ [0,127]（而非 [0,255]），原因是 Qa(2b2a) D(2a1) xs32(2a1) ∈ [−128, 127] 为 s8 格式，这是因为乘积在 ReLU 函数之前并且 Qa(2b2a) = 127/Ra(2b2a) 为激活因子。遵循该流程，在附录 B 中显示激活 a(2c2a)υ8 依赖 xs32(2a1)、xs32(2a2c) 和 xs32(2b2c)。同样，激活 aυ8(3ca) 依赖 xs32(2a1)、xs32(2a2c)、xs32(2b2c) 和 xs32(2c2c)。

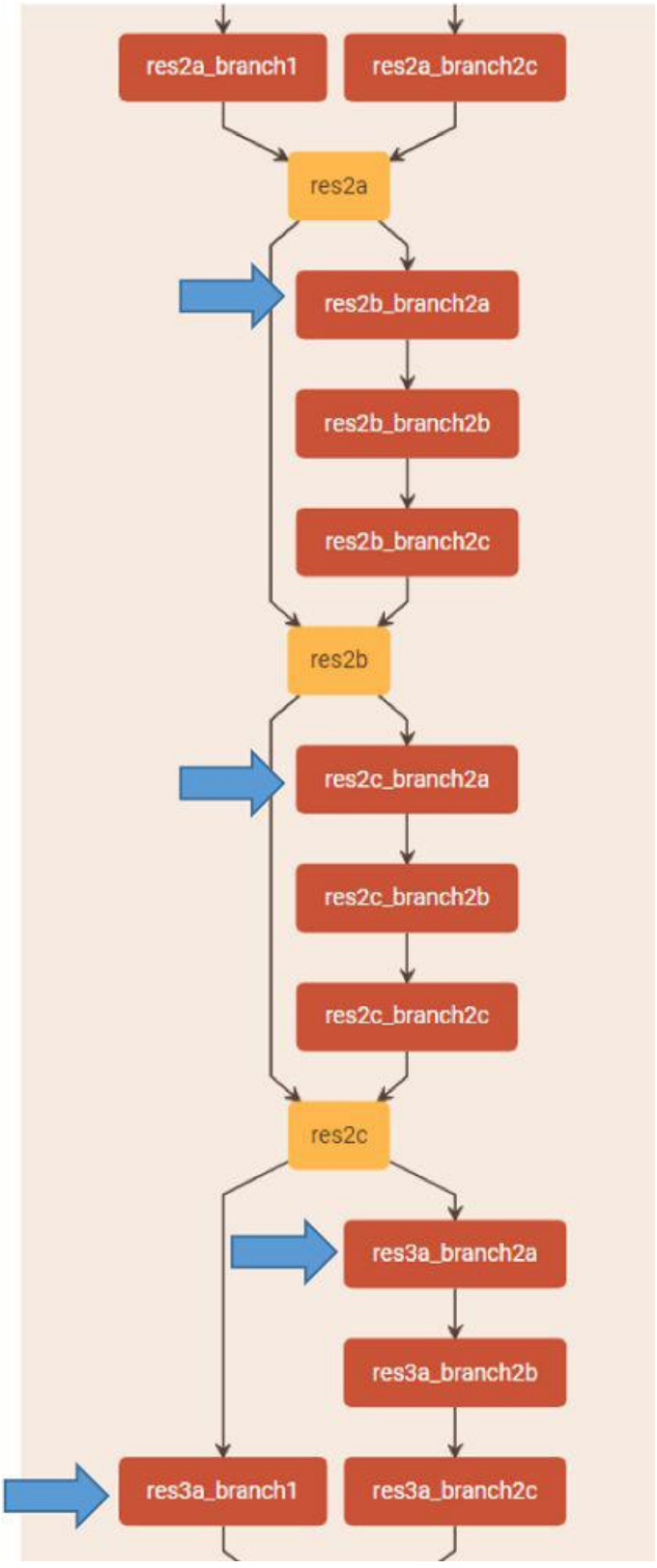

图 7.ResNet-50 中第二组残余块（和第三组中的第一个分支）结构图，使用 ResNet-50 作者在 Caffe deploy.prototxt 中创造的术语。蓝色

箭头标记的层依赖两个或更多激活。图片由 Barukh Ziv、Etay Meiri、Eden Segal 提供。

批处理标准化

不需要批处理标准化 (BN) 推理层，因为它能通过缩放权重值和修改偏差被前一层吸收。该技术仅适用于推理，并非低数值精度所独有。它可以实施于框架层面 ,而非英特尔 MKL-DNN。BN 通常应用于仿射变换 x = W a + b 之后、激活函数之前 （详情请参阅原始的 BN 论文）。BN 将 x 归一化为零均值和单位范数，然后分别将标准化矢量缩放与位移 γ 和 β，后两者也是训练期间学习的参数。在训练迭代期间，使用迷你批次统计数据对 x 进行标准化。使用整个训练数据集的统计数据或变量(如在训练过程中计算的统计数据的连续平均数）来预计算 x 的平均值 E 和方差 V，以进行推理。在推理过程中，BN 输出 y 为：

$$\boldsymbol{y} = BN(\boldsymbol{x}) = \gamma\frac{\boldsymbol{x} - E\boldsymbol{1}}{V} + \beta\boldsymbol{1} = \gamma\frac{\boldsymbol{W}\boldsymbol{a} + \boldsymbol{b} - E\boldsymbol{1}}{V} + \beta\boldsymbol{1} = \frac{\gamma}{V}\boldsymbol{W}\boldsymbol{a} + \frac{\gamma}{V}\boldsymbol{b} + \left(\beta - \frac{\gamma E}{V}\right)\boldsymbol{1} = \boldsymbol{W}'\boldsymbol{a} + \boldsymbol{b}'$$

此处，W′ = γ/V W，b′ = γ/V b + (β − γE /V)1。也就是说，在推理过程中，可以通过调整前一个卷积层或完全连接层的权重与偏差来替换 BN 层。

框架

英特尔在英特尔® Caffe* 分发包中提供了 8 位推理支持。英特尔 DL 推理引擎、Apache* MXNet* 和 TensorFlow* 优化预计将于 2018 年第二季度上市。上述所有 8 位优化目前仅限于 CNN 模型。RNN 模型和其他框架优化将在 2018 年晚些时候推出。

在英特尔 Caffe 分发包中，model.prototxt 文件被修改为包含预计算标量，如图 8 所示。目前，英特尔 Caffe 优化可以提供用作二次方或 fp32 正则值的量化因子，也可以使用每张量 1 个量化因子或每通道 1 个量化因子。使用内置于英特尔 Caffe 分发包的采样工具计算这些量化因子。

```
layer {
  name: "conv2"  type: "Convolution"
  …
  quantization_param {
    precision: DYNAMIC_FIXED_POINT
    bw_layer_in: 8    // input bit-width
    bw_layer_out: 8   // output bit-width
    bw_params: 8      // weights bit-width
    fl_layer_in: 0    // input fraction length
    fl_layer_out: -2  // output fraction length
    fl_params: 8      // weights fraction length
  }
}
```

图 8.量化因子被添加至 model.prototxt 文件。图片由 Haihao Shen 提供。

英特尔深度学习推理引擎是英特尔® 深度学习部署工具套件和英特尔® 计算机视觉 SDK 的一部分。它可以在 Linux* 和 Windows* 操作系统上运行，最初支持在 Caffe、MXNet* 和 TensorFlow* 框架上训练的模型。推理引擎通过为各种硬件后端提供统一 API，帮助部署 DL 解决方案，这些后端包括：搭载英特尔 AVX-2 和英特尔 AVX-512 的英特尔至强处理器、英特尔凌动® 处理器、英特尔® 核芯显卡和英特尔® Arria® 10（英特尔® A10）独立显卡，基于硬件提供各种数值精度。自 2018 年第二季度起，推理引擎将在英特尔至强可扩展处理器上支持 8 位推理。

TensorFlow 已支持 8 位推理和各种优化方法。它可以在训练或校准阶段动态计算范围或收集统计数据，然后分配量化因子。包含这些标量的 TensorFlow 计算图被写入文件。在推理过程中，量化与运行包含各自标量的计算图。TensorFlow 支持两种量化方法。一种方法类似于英特尔 MKL-DNN，它将最小与最大值设置为加性逆元。另一种方法将最小值与最大值设置为需要偏移加缩放的任意值（不受英特尔 MKL-DNN 的支持）。请参阅 Pete Warden 的 博文了解更多详情，但是请注意，该博文已过时，因为它不包含在 TensorFlow 中进行量化的所有方法。

另一款 TensorFlow 工具用于在较低的数值精度下进行重新训练或微调。微调可以提升统计性能。假设有一个在 ƒp32 上训练的模型，它的权重被量化后，使用量化权重对该模型进行微调，每次训练迭代后，对权重进行重新量化。

GemmLowP 是应用于 TensorFlow Lite* 的谷歌库。它使用 υ8 乘法，此处 ,ƒ32 = D × (υ8 − K) ,K 为映射至 ƒp32 = 0 的 υ8 值 ,D > 0 是反量化因子。

Apache MXNet 分支目前不支持 8 位。但是，一位 MXNet 主要贡献者所创建的分支支持 8 位推理。该分支总共包含两种量化值的方法：第一种方法将最小值映射为 0，最大值映射为 255（请注意，零不会映射为准确值）；另一种方法将绝对值的最大值映射为 −127 或 127（请注意，零映射为零 - 类似于英特尔 MKL-DNN）。上述方法的主要区

别在于该 MXNet 分支中的标量未经过预计算。在实际的推理步骤中对它们进行计算，这将减弱低数值精度的优势。在该分支中，通过将来自输入的标量乘以来自权重的标量，计算出用于激活的标量：激活标量 = 输入标量 * 权重标量 ,此处 ,输入 = 输入标量 * 量化输入 ;权重 = 权重标量 * 量化权重；激活 = 激活标量 * 量化激活；输入、权重、激活和标量均为 ƒp32 格式，量化输入与量化权重为 int8 格式，量化激活为 int32 格式（了解详情）。由于激活的最小值与最大值已被跟踪，只有遇到 fp32 层（如 softmax）时，才对值进行反量化。

TensorRT 量化为 s8 格式的方法类似于英特尔 MKL-DNN，前者通过最大限度地缩小量化分布与参考分布之间的 KL 散度 ,发现更小的范围。TPU 团队声称使用 int8 乘法的 TPU 被广泛应用于各种模型中，包括 LSTM 模型。软件堆栈将来自 TensorFlow 计算图的 API 调用转化为 TPU 指令。

Caffe2 文档指出“量化计算的未来发展方向不会一成不变”，但是目前未披露任何量化计划。

PyTorch 的分支提供了各种量化选项，但是并未讨论哪一种更好。

Microsoft 借助在英特尔® Stratix® 10 FPGA 上运行的自定义 8 位浮点格式 (ms-fp8) 推出了 Project Brainwave*。尚未披露该格式、量化技术或框架实施的详细信息。Project Brainwave* 不仅支持 CNTK*

和 TensorFlow，还计划将在常用框架中训练的模型转换为基于计算图的内部中间代码，以支持更多其他框架。

模型与计算图优化

模型优化可以进一步提升推理性能。例如，在 ResNet 中，步长操作可以被迁移至更早的层，无需修改最终结果或减少操作数量，如图 9 所示。该修改适用于 8 位和 32 位。

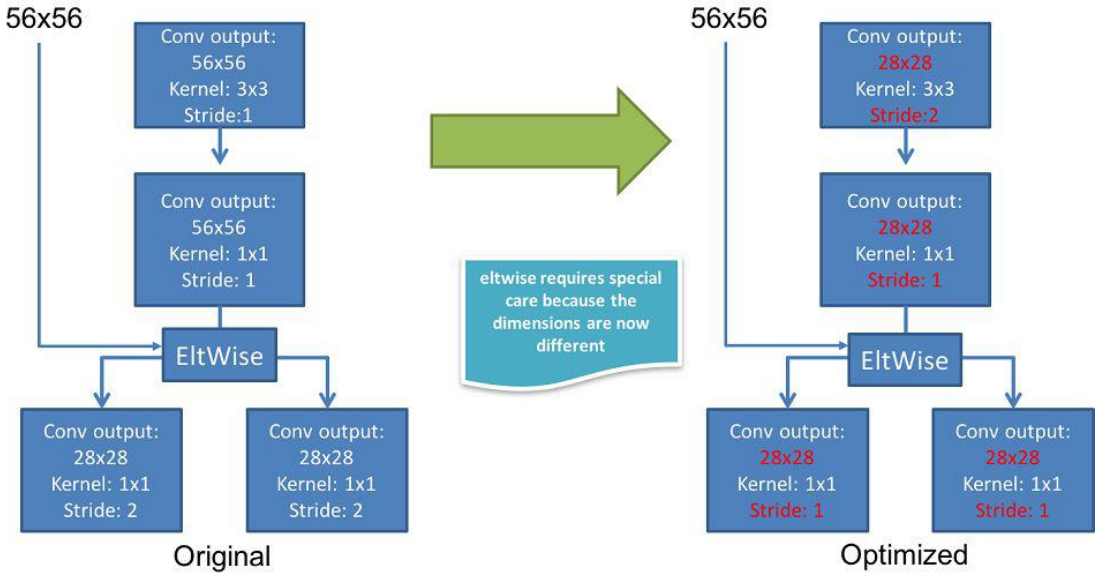

图 9.在推理过程中，左图层上的步幅 2 可以被迁移至更早的层，这将减少操作数量，并且不会修改结果。图片由 Eden Segal 和 Etay Meiri 提供。

结论

低数值精度推理与训练可以提升计算性能，将统计准确性的损失降至最低或零。英特尔在最新一代英特尔至强可扩展处理器上支持 8 位精度推理。英特尔还支持在未来的硬件与软件微架构上实施 8 位精度推理和

16 位精度训练，为编译器、英特尔 MKL-DNN 库和常用的深度学习框架提供支持。