对机器说“欢迎”

面向智能物联网应用的低功耗机器学习技术

作者

Pieter van der Wolf
首席研发工程师,
Synopsys Inc.

Dmitry Zakharov
高级软件工程师,
Synopsys Inc.

Introduction

近年来,作为用于构建高级功能设备的技术,机器学习已在多个领域得到应用。例如:

- 具有语音控制功能的智能扬声器,可对大量的语音命令进行语音识别。
- 可穿戴的活动跟踪器,可基于陀螺仪、加速计和磁力计等传感器的输入数据来识别人类活动,如走着、站着和坐着。
- 带有摄像头的智能门铃,可通过面部检测而触发报警系统,向主人的智能手机发送警报,并附有图像或视频。
- 自动驾驶汽车,可采用先进的计算机视觉技术来检测车辆和行人等。

这些设备均使用机器学习技术,经训练后可从一个或多个传感器(如麦克风、陀螺仪和摄像头等)捕获到的数据中识别出某些复杂模式(如语音命令、人类活动、面部和行人等)。当识别出此类模式后,这些设备还能采取适当的行动。例如,当识别出语音命令“播放音乐”时,智能扬声器将能够启动歌曲播放程序。

机器学习的主题就是不需要显式编程而算法具有自我学习能力。如图1所示,机器学习分为训练和推理两部分。

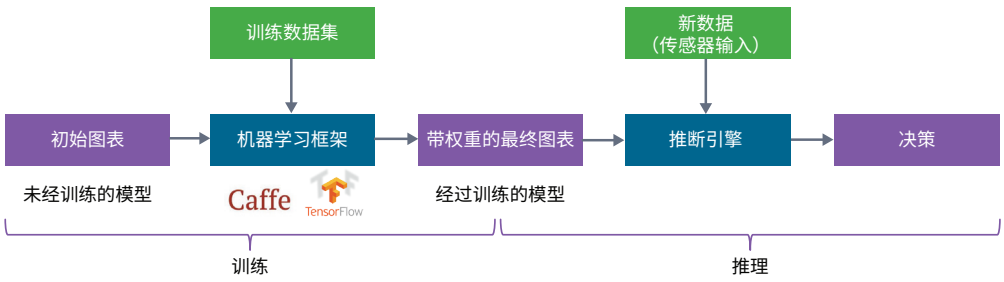

图1:机器学习中的训练与推理

训练始于未经训练的模型,例如具有选定图结构的多层神经网络。在这些神经网络中,每一层都使用系数或权重集将输入数据转换为输出数据。我们使用Caffe或TensorFlow等机器学习框架,通过大型训练数据集来训练这种模型,从而得到经过训练的模型,例如,权重已经过调准、可对输入数据进行分类的神经网络,能够对上述活动跟踪器中不同类型的人类活动进行分类。

推理使用经过训练的模型来处理传感器捕获到的输入数据，以便运用通过训练而获得的识别能力来推断出复杂模式。例如，它可以检查输入数据是否与已经训练好的神经网络所能识别出的某个类别相匹配，例如活动跟踪器设备中的“走着”或“坐着”。因此，推理是利用经过训练的模型来识别新数据的过程。推理常在现场进行。本书将重点讲述推理而不是训练。具体而言，本书将重点讲述如何在可编程处理器上高效实现机器学习推理。

机器学习推理的处理要求可能因应用不同而存在很大差异。影响处理要求的一些关键因素包括：

- 输入数据速率：指传感器捕获数据样本的速率。例如，这些样本可以是来自摄像头的像素，也可以是来自麦克风的脉冲编码调制(PCM)样本。输入数据速率的范围可以是每秒几十个样本，例如通过少量传感器来识别人类活动，也可以是每秒数十亿个样本，例如通过高清摄像头以高帧频捕获图像的高级计算机视觉应用。
- 训练后模型的复杂度：指在推理过程中对一组样本（如输入图像）执行的操作数量。例如，在神经网络中，复杂性取决于图中的层数、每层的（多维）输入和输出映射的大小、以及计算输出映射时使用的权重数。低复杂度的神经网络一般不到十层，而高复杂度的神经网络则可以多达几十层 [2]。

表1举例显示了几个机器学习应用的输入数据速率和模型复杂度。

机器学习应用	输入数据速率	训练后模型的复杂度
人类活动识别	10s Hz（少量传感器）	低至中
语音控制	10s kHz (如16 kHz)	低至中
人脸检测	100s kHz（低分辨率和帧频）	低至中
高级计算机视觉	100s MHz（高分辨率和帧频）	高

表1：例如机器学习应用的输入数据速率和模型复杂度

从表1中可以看出，输入数据速率和模型复杂度之间可能存在显著差异。因此，机器学习推理的计算要求也可能相差很大。本书将重点讲述具有中低计算要求的机器学习推理。具体而言，我们将着重讲述可用于构建智能物联网边缘设备的机器学习推理。请注意，对于高端机器学习，Synopsys提供功能强大的DesignWare ARC EV处理器[3]。

在下一节，我们将进一步详细说明如何高效实现低/中端机器学习推理。我们将以DSP增强型DesignWare®ARC®EM9D处理器为例来讨论能够高效执行机器学习推理中常用计算的可编程处理器的特性和功能。我们还将讨论可用于快速实施推理引擎以支持各种机器学习应用的扩充软件功能库，并通过基准测试来证明在ARC EM9D处理器上使用这个软件库对机器学习推理效率的提升。

低/中端机器学习推理的要求

采用机器学习推理的物联网边缘设备通常执行不同类型任务的处理流程，如图2所示。

图2：机器学习推理应用中不同类型任务的处理流程

在对经过训练的模型执行真正的神经网络处理之前，这些设备通常会对传感器输入数据执行一些预处理和特征提取操作。例如，具有语音控制能力的智能扬声器可能首先会执行声学回声消除和多麦克风波束形成等操作，以便对语音信号进行预处理。然后，它可能会通过FFT来提取将在已经被训练能够识别语音命令词汇的神经网络处理中使用的频谱特征。

神经网络处理

对于神经网络中的每一层，输入数据都必须转换为输出数据。卷积是常用的转换方式，是指通过一组训练好的权重将输入数据汇总在一起，或者更确切的说，关联在一起。这种转换方式常在卷积神经网络(CNN)中用于图像或视频识别。

图3示出了执行点积运算的二维卷积。该卷积使用二维加权核作为权重，使用输入数据中具有相同宽度和高度的选定二维区域作为加权核。点积运算在输出映射中生成了一个值(M23)。该示例未在输入数据的边界应用任何填充，因此输出值是并列的(2, 3)。为了计算完整的输出映射，我们在输入映射上“移动”加权核，并对选定的二维区域执行点积运算，从而为每个点积生成输出值。例如，通过向右移动一步以及对使用输入样本A24-A26、A34-A36和A44-A46对该区域执行点积运算，我们可能会算得M24。

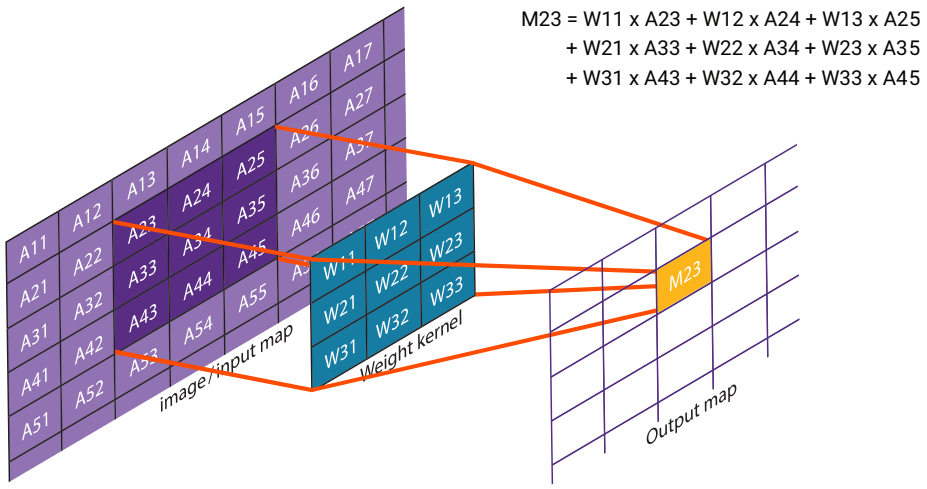

图3：对输入数据应用加权核、以计算输出映射值的2D卷积

输入和输出映射常是三维的。也就是说，它们具有宽度、高度和深度，具有不同深度的平面通常被称为通道。对于深度> 1的输入映射，您可通过对来自多条输入通道的输入数据执行点积运算来计算输出值。对于深度> 1的输出映射，您必须对每条输出通道逐一执行卷积运算，并且对不同的输出通道使用不同的加权核。深度卷积(Depthwise convolution)是具有相同数量输入通道和输出通道的一类特殊的卷积层，它使用与输出通道具有相同深度值的输入通道来计算每条输出通道的值。

全连接层是另一类卷积层，它使用与输入映射中的样本数相同的权重数对每个输出值执行点积运算。

上述卷积层中的主要运算都是对输入样本和权重执行的点积运算。因此，处理器必须能够高效执行此类点积运算，从而要求处理器必须具有高效计算能力，例如，使用乘法累加(MAC)指令以及高效访问输入数据、加权核和输出数据等。

CNN是前馈神经网络。卷积层在处理输入映射时，不会保留影响下一个输入映射处理工作的任何状态。递归神经网络(RNN)则是另一类不同的神经网络，在处理输入序列时会保留状态。因此，RNN还具有跨时间识别模式的能力。RNN常在文本和语音识别环境中使用。

您可使用种类繁多的RNN单元来构建网络。在基本形式中，RNN单元计算的输出如下：

$$h_t = f(x_t \cdot W_x + h_{t-1} \cdot W_h + b)$$

其中 x_t是输入序列中的帧 t ，h_t 是 x 的输出，w_x和 w_h是权重集，b是偏差，f ()是输出激活函数。因此，一个输出的计算一方面需要使用新的输入数据对一组权重执行点积运算，另一方面需要使用先前的输出数据对另一组权重执行另一个点积运算。鉴于此，对RNN而言，点积运算也是必须高效执行的关键运算。长短期记忆(LSTM)单元是另一个众所周知的RNN单元。LSTM单元的结构比我们刚才讨论的基本RNN单元更复杂，但点积同样是主导运算。

神经网络中使用激活函数来执行一些非线性映射，以转换数据。例如线性整流函数(ReLU)，sigmoid函数和双曲正切函数(TanH)。激活函数可对单个数据值进行运算并产生单个结果。因此，对激活层而言，输出映射的大小等于输入映射的大小。

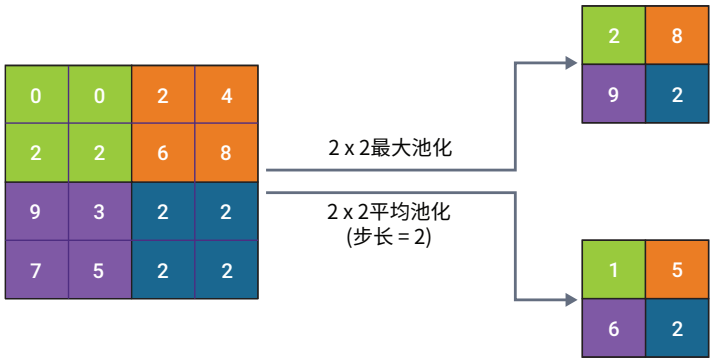

Figure 4: Example pooling operations: max pooling and average pooling

神经网络也可能带有池化层，用于针对输入数据的（小型）区域来计算单一输出值，从而将输入映射转变成更小的输出映射。图4显示了两个示例：最大池化和平均池化。实际上，池化层可对具有宽度和高度维度的数据执行下采样。输出映射的深度与输入映射的深度相同。

实施要求

为了在实施机器学习推理引擎时获得足够准确的结果，您必须使用适当的数据类型。在训练阶段，数据和权重通常表示为32位浮点数据。在嵌入式系统中部署推理模型时，常见的做法是使用量化数据和权重的整数或定点表示法[4]。在训练阶段，应考虑量化误差的潜在影响，以避免对模型性能产生显著影响。输入映射、输出映射和加权核的元素，通常使用16位或更小的数据类型来表示。例如，在语音应用中，数据样本通常表示为16位数据类型。在图像或视频应用中，12位或更小的数据类型一般足以表示输入样本。每层神经网络的精度要求可能有所不同。在执行点积运算时，您应特别注意用于对部分结果求和的累加器的数据类型。这些累加器应足够宽，以便当您对（量化的）权重和输入样本执行点积运算时避免（中间）结果溢出。

由于输入数据速率以及所用神经网络的复杂度均有限，因此，低/中端机器学习推理的存储要求通常并不高。输入和输出映射的大小经常是有限的，即几万字节或更小，并且加权核的数量和大小也相对较小。针对输入映射、输出映射和加权核尽量使用最小的数据类型，有助于降低存储需求。

低/中端机器学习推理需要以下类型的处理功能：

- 各种类型的预处理和特征提取：常通过DSP密集型计算来实现。
- 神经网络处理：针对多维数据将点积运算作为主导计算并应用常规访问模式。当对二维数据使用标量激活函数和池化操作时，则需要满足额外要求。
- 决策：发生在神经网络处理完成之后，更加以控制为导向。

您可使用异构多处理器架构来实现不同类型的处理，借助不同类型的处理器来满足不同的处理要求。但是，对于低端/中端机器学习推理而言，总体计算要求经常并不高，可通过以合理频率运行的单个处理器来完成，前提是它具有适当的功能。使用单个处理器可避免与多处理器架构相关的区域和通信开销，还可因为通过一条工具链来支持整个应用来简化软件开发工作。但前提是处理器必须能以出色的性能来执行不同类型的处理任务，即DSP、神经网络处理和控制处理。在下一节，我们将进一步讨论可编程处理器凭借哪些功能来达到此类执行效率。

低成本是很多物联网边缘设备的关键要求。因此，通过添加机器学习推理来提高物联网边缘设备的智能性必须具有成本效益。成本主要来自硅面积，尤其是对于大容量产品而言，因此，处理器在实施机器学习推理时必须尽可能地减少逻辑区域并利用小型存储器，这一点非常重要。小型代码是限制指令存储区域大小的关键。

很多物联网边缘设备都是电池供电的，并且电力预算很紧张。因此，需要以uW/MHz为功耗度量并具有出色执行效率的高能效处理器，以确保处理器能以低频运行。低功耗对于需要始终在线的物联网边缘设备而言尤为重要，例如：

- 智能扬声器和智能手机等需要“始终聆听”以执行语音命令的设备。
- 需要“始终观察”以执行例如 面部检测或手势识别等任务的带摄像头的设备。
- 需要“始终感应”的健康和健身监测设备。

这些设备通常都借助智能技术来降低功耗。例如，“始终聆听”的设备可对麦克风信号进行采样并使用简单的语音检测技术来检查是否有人在说话。它仅在检测到确实存在语音活动时才会启动计算密度更高的机器学习推理，以识别语音命令。处理器必须限制这些不同状态的功耗，即语音检测和语音命令识别。因此，它必须具有多项电源管理功能，包括有效的睡眠模式和省电模式。

低功耗机器学习推理的处理器功能要求

选择适当的处理器是确保低/中端机器学习推理高效运行的关键。在本节，我们将介绍32位DSP增强型ARC EM9D处理器的多个关键功能，并举例说明如何使用这些功能来高效实施神经网络处理。

如前所述，对输入样本和权重而言，点积运算是主导计算。乘法累加(MAC)运算是执行点积运算的关键原语，可用于对输入样本和权重的乘积逐步求和。MAC运算的矢量化是提高神经网络处理效率的重要途径。图5显示了ARC EM9D处理器的两类矢量MAC指令。

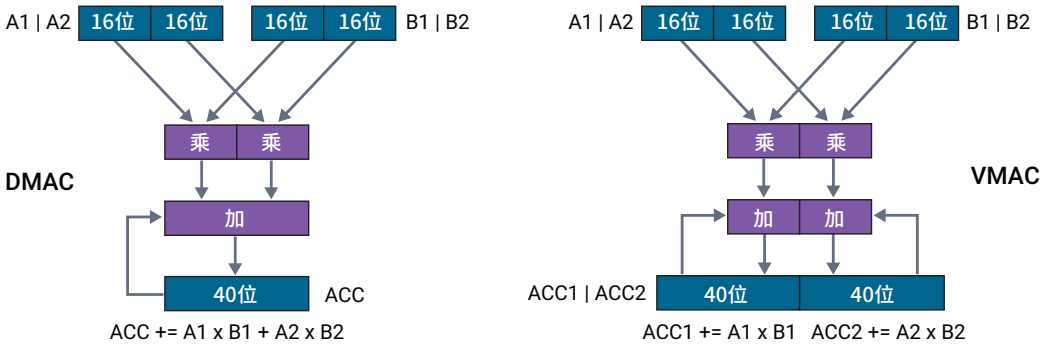

图5：ARC EM9D处理器的两类矢量MAC指令

这两个矢量MAC指令均作用于2x16位矢量操作数。左侧的DMAC指令是可用于实施点积的双MAC，A1和A2是来自输入映射的两个相邻样本，B1和B2是来自加权核的两个相邻权重。ARC EM9D处理器支持32位累加器，可以启用额外的8个警戒位来避免溢出。DMAC运算可以有效地用于具有偶数宽度的加权核，与标量实施相比，可将MAC指令数量减少一半。但是，该指令对于具有奇数宽度的加权核效果较差。这种情况下，如图5右侧所示，VMAC（矢量-MAC）指令可用于并行执行两个点积运算，将中间结果累积到两个累加器中。如果加权核在步长为1的情况下“移入”输入映射，A1和A2则是来自输入映射的两个相邻样本，并且B1和B2的值是同时适用于A1和A2的相同权重。

点积运算的高效执行不仅需要适当的矢量MAC指令，而且还需要足够的存储带宽来为这些MAC指令提供操作数，并且需要采取适当方式来避免与地址更新及数据大小转换相关的开销。出于这些目的，ARC EM9D处理器提供具有高级地址生成特性的XY存储器。基本来说，XY架构允许处理器同时并行访问多达三个逻辑存储器，如图6所示。处理器可通过常规加载或存储指令来访问这些存储器，或者允许功能单元通过地址生成单元(AGU)访问这些存储器。您可通过地址指针或修饰符来设置AGU，让地址指针指向某个存储器中的数据，当通过AGU访问数据时，则使用修饰符以特殊方式来更新该地址指针。设置完成后，您即可通过在指令中使用AGU来直接从存储器访问操作数和存储结果到存储器，无需对这些操作数和结果执行显式加载或存储指令。通常情况下，AGU都是在都是在某个软件循环体之前设置，然后在循环周期内重复使用。

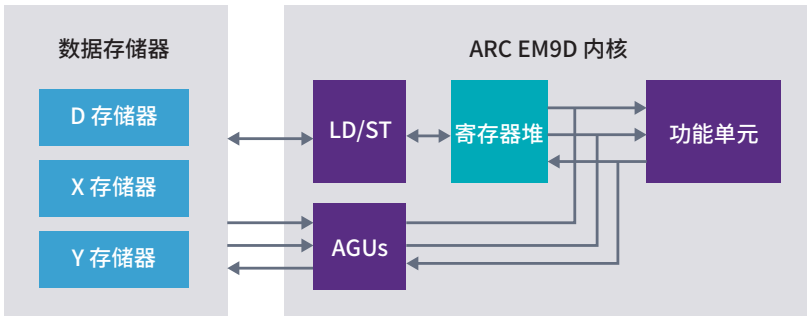

图6：具有XY存储器和地址生成单元(AGU)的ARC EM9D处理器

AGU支持与机器学习推理相关的以下功能：

- 每个地址指针具有多个修饰符，允许您针对地址指针更新规定并使用不同的方案。例如，为了支持2D访问模式，您可以通过一个修饰符来规定在输入映射中的行内执行小的水平步长，同时通过另一个修饰符来规定大步长，以便将指针移至输入映射中的下一行。
- 数据大小转换功能，例如，允许您即时扩展2x8位数据，将其用作2x16位矢量操作数。无需额外指令用于解压缩和符号扩展。
- 复制功能，允许您将数据值即时复制到矢量中。例如，如上所述，您可将单个权重值复制到2x16矢量中，以便在VMAC指令中使用。

总之，使用XY存储器和AGU可以实现非常高效的编码，因为无需指令便可加载和存储数据、执行指针数学运算、或者转换和重新排列数据。所有这些都是在处理器通过AGU访问数据时隐式执行的，每个处理器周期可访问三次存储器。在下一节，我们将通过代码示例来说明处理器的XY存储器和AGU是如何作用于机器学习推理的。

大多数的其他嵌入式处理器都必须发出显式加载和存储指令才能访问存储器。在单发射处理器中，这些指令的执行可能会消耗大部分的可用处理器周期，从而大大降低以MACs/Cycle为基准的吞吐量。VLIW等多发射处理器旨在将加载和存储操作和MAC计算操作并行，以提高吞吐量。但是，由于必须采用宽指令，因此，指令存储器中的代码尺寸和功耗都更高一些。

在典型的机器学习推理用例中，处理器需要一边处理先前捕获到的传感器数据，一边捕获新的传感器数据。ARC EM9D处理器内含小型集成式uDMA，可在内核处理当前数据时将下一个输入数据移入存储器。通过使用多存储体存储器和适当的存储器仲裁机制，使得处理器无阻塞并在后台传输这些数据。

ARC EM9D处理器提供广泛的DSP功能，可助力预处理和特征提取（如FFT）等其他功能高效发挥作用。这些功能包括：

- 零开销环路
- 支持饱和与舍入的定点算术
- 具有警戒位的宽累加器
- 2x16和4x8矢量支持
- 16+16复杂算术和butterfly算法支持
- 循环和位反转寻址

DSP和高效控制处理功能相结合，导致ARC EM9D非常适合在单个处理器上实施完整的低/中端机器学习推理应用。

用于机器学习推理的软件库

选好适当的处理器后，下一个问题便是如何针对选定的机器学习推理应用来高效实施软件。为此，我们将提供一个可复用的软件模块库，并展示如何在ARC EM9D处理器上高效实施这些模块。

embARC机器学习推理(MLI)库[5]是用于构建神经网络的一组C函数和相关数据结构。库函数，又名内核，在神经网络中执行与整个层相关的处理任务，以张量结构来表示多维输入/输出映射。因此，神经网络图可作为一系列的MLI函数调用来实施。表2列出了该处理器目前支持的MLI内核，共分六组。每个内核均与该库中的多个函数相对应，包括专门针对特定数据类型、加权核大小和步长进行优化处理的函数。此外，该库还提供辅助函数，用于执行数据类型转换、数据指向、以及不由内核功能直接执行的其他操作。

embARC MLI库的一个显著特征是对循环层提供显式支持。诸如麦克风或加速计等传感器提供的序列数据常在物联网中使用。如上所述，递归神经网络在处理输入序列时保持其原有状态，因此能够跨时间来识别模式。这一效力已经得到证实，导致该网络在语音识别等最先进的解决方案中得到广泛应用[6]。作为完整内核，embARC MLI库可直接支持LSTM和基本 RNN。该库尚不支持GRU单元或Projected LSTM等RNN变体，但您可以使用基本RNN、激活、元素操作和数据操作内核等现有内核来实施它们。

组	内核	组的说明
卷积	· 2D卷积 · 深度2D卷积	使用一组训练好的权重来汇总输入要素
池化	· 平均池化 · 最大池化	通过函数来汇总输入特征
递归与核心	· 基本 RNN · 长短期记忆(LSTM) · 全连接	递归和全连接内核
转换（激活）	· ReLU（包括Relu1、Relu6和Leaky ReLU） · Sigmoid和TanH · SoftMax	根据特定的非线性函数来转换每一个输入元素
元素	· 加法、减法、乘法 · 最大值和最小值	将多操作数函数元素作用于多个输入
数据操作	· 置换 · 串联 · 填充2D	按指定模式移动或扩展输入数据

表2：embARC MLI库中支持的内核

数据表示法是MLI定义的重要组成部分。尽管内核种类繁多，但直接参与计算的数据却具有共同属性。因此，深度学习框架通常采用统一的数据表示法。例如，TensorFlow使用张量，而Caffe则使用名为blob的类似对象。这些对象表示拥有适当大小的多维数组，并且可以包括附加数据，例如通往相关对象的链接和同步原语等。采用类似方式的mli_tensor是MLI中的通用数据容器。这个轻量级张量中包含描述数据的必要元素：指向数据缓冲区的指针、其容量、形状、等级和数据格式的特定值等。内核可将多个张量用作输入，以生成输出张量。例如，2D卷积内核带有三个输入张量：输入映射、权重和偏差。所有特定于内核的参数（例如卷积内核的步长和填充，或者置换内核全新的维度顺序等）都将被归入不同的配置结构组。这种数据表示法具有以下几个优点：

- 为访问函数提供简单明了的界面。
- 允许针对一个内核的多个版本使用相同的接口。
- 可与分层神经网络很好地匹配，因此易于使用并且自然具有良好的库可扩展性

embARC MLI库采用基于Q-notation的带符号的定点数据类型[7]。该库可同时针对输入/输出数据和权重支持具有8位和16位数据类型的张量结构。在构建应用时，您应选择能为神经网络中的每一层都提供足够精确结果的数据类型。除了同时针对数据和权重支持具有相同数据类型（16位或8位）的内核外，MLI库还提供具有16位数据和8位权重的内核，以便您在精准度与存储容量之间更加灵活地做出折中选择。

ARC EM9D处理器的AGU可支持复杂的存储器访问模式，无需在加载和存储指令上花费处理器周期。我们将以全连接层的代码为例来说明AGU的优势。每个输出值都是使用点积运算算得的。我们假设输入是16位值的矢量，权重是8位值的矢量。通常情况下，这意味着将权重操作数扩展为16位值，因此会在循环体内产生额外的处理周期。ARC EM9D处理器包含多个双MAC指令，可通过累加来执行两次乘法运算（见图5中的左图）。例如，在16x8双MAC指令中，一个操作数是2x16位矢量，另一个操作数是2x8位矢量，因此允许您直接使用8位数据。图7中显示了使用高级C代码生成的汇编代码，我们可以看到，这将会生成一个主体仅为一条指令、但每个循环却具有两个16x8 MAC吞吐量的零开销环路。

```
    …         ; AGU 0 is used for input data. AGU 1 is used for weights
    SR …      ; setup AGU 0 for loading next 2x16-bit vector
    SR …      ; setup AGU 1 for loading next 2x8-bit into lower bytes of 32-bit word
    …
    LP        lpend
    DMACHBL   0, %agu_u0, %agu_u1 ; DMAC + 2 loads + 2 pointer updates
lpend:          …
```

图7：使用MLI C代码为具有16位输入数据和8位权重的全连接层生成的汇编代码

其他内核也可通过双MAC指令取得类似的优化效果，如RNN内核和一些卷积内核。但是，这种方法在任何情况下都比较麻烦。我们以具有3x3加权核大小的通道-高度-宽度(CHW)数据布局为例，来演示一下深度卷积应用。这种布局对每条输入通道分别应用2D卷积。由于使用奇数加权核以及短MAC系列来计算输出值，因此，双MAC指令在这里无法得到优化利用。如果卷积步长参数等于1，则使用相邻输入数据元素来计算相邻输出值。这意味着我们可以使用两个累加器的矢量MAC指令来同时计算两个输出值（见图5中的右图），如图8中生成的汇编代码所示。

```
    …         ; AGU 0 and AGU 1 are used for input data (one data pointer with two modifiers)
              ; AGU 2 is used for weights
    SR …      ; setup AGU 0 for loading next two 16-bit input values
    SR …      ; setup AGU 1 for loading two 16-bit input values at next row of input data
    SR …      ; setup AGU 2 for loading next 8-bit value with sign extension & replication
    …
    LP            lpend
    VMAC2HNFR     0, %agu_u0, %agu_u2 ; VMAC + 2 loads + sign ext + repl + 2 pointer updates
    VMAC2HNFR     0, %agu_u0, %agu_u2 ; VMAC + 2 loads + sign ext + repl + 2 pointer updates
    VMAC2HNFR     0, %agu_u1, %agu_u2 ; VMAC + 2 loads + sign ext + repl + 2 pointer updates
lpend:            …
```

图8：使用MLI C代码为具有16位输入数据和8位权重的2D卷积而生成的汇编代码

该示例演示了AGU在处理复杂数据寻址模式时的灵活性，包括使用两个修饰符对输入数据及权重的符号扩展和复制进行2D访问。典型的卷积层计算方法 - 如Caffe推广的方法 - 是使用额外的重排图像块为矩阵列(im2col)转换。这种转换能够简化后续的卷积计算工作，因此在某些处理器上很有用，但却要以巨大的开销为代价。ARC EM9D处理器的高级AGU消除了这些转换需求，因此可实现高效嵌入。

embARC MLI库中包含适用于每个内核的通用函数，可与任何有效的参数值组合结合使用。从上面的示例可以看出，当我们针对步长、加权核大小或填充等任何特定的参数值来优化卷积内核时，都能显著提高其性能。对于内核池化组也是如此。因此，embARC MLI库还为卷积和池化组中的每个内核提供了一系列的专用函数和通用函数。此类专用函数已针对特定的参数值进行了调整，这一点从函数名称中可见一斑。例如，上面讨论的具有3x3加权核大小、步长为1、以内核大小来填充（相当于TensorFlow中的“SAME”填充方案）的深度卷积函数格式如下：

```
mli_krn_depthwise_conv2d_chw_fx8w16d_k3x3_str1_krnpad(...)
```

尽管存在诸多参数组合，并且embARC MLI库提供大量的专用函数，但并不是每个函数都必须手动实施。实际上，专用函数是通过与类型无关的泛型C++模板生成的包装函数(wrapper)。因此，专门化本身就是具有特定参数值的函数的实例化。embARC MLI库是使用ARC EM9D处理器的高级MetaWare编译器通过高级C/C++代码编写的，其内联函数仅在少数几个地方使用。通常情况下，遵循通用DSP编程建议并依赖于编译器驱动的常量传播，足以确保专用函数得到高效实施。

虽然上面的示例主要聚焦ARC EM9D处理器上的embARC MLI库实施，但该库也可在其他ARC处理器上使用。embARC MLI库的这种可移植性以高级ARC MetaWare编译器作为支撑，也可从高级C/C++ MLI代码库代码为其他ARC处理器生成高效代码。

从用户的角度看，embARC MLI库提供了易用性，允许他们无需深入了解ARC ISA和软件优化细节即可构建高效的机器学习推理引擎。embARC MLI库提供了一组全面的优化函数，允许用户专注于使用他们所熟悉的高级结构来编写应用和嵌入式代码，以进行机器学习推理。

示例应用和基准

embARC MLI库以及演示该库诸多用例的示例应用，均可从embarc.org [5]获得。例如：

- CIFAR-10低分辨率对象分类器：CNN图
- 人脸检测：CNN图
- 人类活动识别(HAR)：基于LSTM的网络
- 关键字识别：使用Google语音命令数据集训练的、具有CNN和LSTM图层的图

CIFAR-10 [8]示例应用基于Caffe [9]教程。CIFAR-10数据集是一组6万个低分辨率RGB图像（32x32像素），共由“cat”、“dog”和“ship”等10类对象组成。该数据集在机器学习和计算机视觉中被广泛用作“Hello World”示例。其目的是使用5万个此类图像来训练分类器，以便对数据集中的其他1万个图像进行高精度分类。我们使用图9显示的CIFAR-10 CNN图执行训练与推理。此图与Caffe教程中的CIFAR-10图相匹配，包括末尾处的两个全连接层。

图9：CIFAR-10示例应用的CNN图

您可为处理器支持的所有embARC MLI库数据类型构建CIFAR-10示例应用：8位特征数据和权重，16位特征数据和权重，16位特征数据和8位权重。您应对上述每一个选项执行适当的编译时量化。我们使用特征数据和权重均为8位的CIFAR-10示例应用在ARC EM9D处理器上对机器学习推理的性能开展基准测试。此CIFAR-10应用的代码使用embARC MLI库所构建，如图10所示。

```
mli_krn_permute_fx8(&input, &permute_hwc2chw_cfg, &ir_Y);

ir_X.el_params.fx.frac_bits = CONV1_OUT_FRAQ;
mli_krn_conv2d_chw_fx8_k5x5_str1_krnpad(&ir_Y, &L1_conv_wt, &L1_conv_b, &conv_cfg, &ir_X);
mli_krn_maxpool_chw_fx8_k3x3(&ir_X, &pool_cfg, &ir_Y);

ir_X.el_params.fx.frac_bits = CONV2_OUT_FRAQ;
mli_krn_conv2d_chw_fx8_k5x5_str1_krnpad(&ir_Y, &L2_conv_wt, &L2_conv_b, &conv_cfg, &ir_X);
mli_krn_avepool_chw_fx8_k3x3_krnpad(&ir_X, &pool_cfg, &ir_Y);

ir_X.el_params.fx.frac_bits = CONV3_OUT_FRAQ;
mli_krn_conv2d_chw_fx8_k5x5_str1_krnpad(&ir_Y, &L3_conv_wt, &L3_conv_b, &conv_cfg, &ir_X);
mli_krn_avepool_chw_fx8_k3x3_krnpad(&ir_X, &pool_cfg, &ir_Y);

ir_X.el_params.fx.frac_bits = FC4_OUT_FRAQ;
mli_krn_fully_connected_fx8(&ir_Y, &L4_fc_wt, &L4_fc_b, &ir_X);

ir_Y.el_params.fx.frac_bits = FC5_OUT_FRAQ;
mli_krn_fully_connected_fx8(&ir_X, &L5_fc_wt, &L5_fc_b, &ir_Y);
mli_krn_softmax_fx8(&ir_Y, &output);
```

图10：CIFAR-10推理应用程序的MLI代码

如图10中的代码所示，图中的每一层都是通过调用embARC MLI库中的函数来实现的。在执行第一个卷积层之前，我们从embARC MLI库中调用了置换(permute)函数，以便将RGB图像转换为CHW格式，从而确保相邻的数据元素来自相同颜色的平面。该代码进一步表明，具有两个缓冲器ir_X和ir_Y的乒乓方案适用于缓冲输入和输出映射。

有些公司曾使用极为相似的CIFAR-10 CNN图在其嵌入式处理器上对机器学习推理的性能开展了基准测试，并将测试结果发布在[10]和[11]中。表3列出了我们使用的CIFAR-10 CNN图的模型参数，以及ARC EM9D处理器和其他两个嵌入式处理器的性能测试数据。

编号	图层类型	权重、张量、形状	输出、张量、形状	系数	ARC EM9D [Mcycles]	处理器A [Mcycles]	处理器B (RISC-V ISA) [Mcycles]
0	置换	-	3x32x32	0	0.01	-	-
1	卷积	32x3x5x5	32x32x32 (32K)	2400	1.61	6.78	-
2	最大池化	-	32x16x16 (8K)	0	0.16	0.34	-
3	卷积	32x32x5x5	32x16x16 (8K)	25600	3.45	9.25	-
4	平均池化	-	32x8x8 (2K)	0	0.1	0.09	-
5	卷积	64x32x5x5	64x8x8 (4K)	51200	1.75	4.88	-
6	平均池化	-	64x4x4 (1K)	0	0.07	0.04	-
7	全连接	64x1024	64	65536	0.03	0.02	-
8	全连接	10x64	10	640	0.001		-
	总计	**-**	**-**	**145376**	**7.1**	**21.4**	**12.3**

表3：CIFAR-10 CNN图的模型参数，提供面向ARC EM9D和其他嵌入式处理器的循环计数

作为以216 MHz时钟频率运行的处理器，处理器A的性能数据以毫秒为单位发布在[10]中。表3中显示的处理器A的时钟周期数是这个毫秒数与时钟频率的乘积。[10]中发布的CIFAR-10 CNN图，其卷积和池化层均与表3所列的相同，但却使用具有10x64x4x4滤波器形状的单个全连接层将64x4x4输入映射直接转换为10个输出值。Caffe CNN图的这种变化将大大减小权重数据的大小，但却需要重新训练该图。这种变化对总时钟周期数的影响很小。

发布于[11]中的RISC-V处理器性能数据显示，在高度并行化的8核RISC-V架构上执行CIFAR-10图总共需要1.5个Mcycles。为了计算单个RISC-V内核的总循环数，我们假设性能主要取决于5x5卷积所消耗的循环 - 该卷积构成了此图中98%以上的计算操作。对于这些5x5卷积，据参考文献[11]显示，从1核系统到8核系统的加速是18.5/2.2 = 8.2。因此，经合理估算，我们认为单个RISC-V内核的总循环数应是1.5 x 8.2 = 12.3个Mcycles。

从表3中，我们可以得出这样的结论：在不使用任何特定加速器的情况下执行相同的机器学习推理任务时，ARC EM9D处理器所消耗的时钟周期数比处理器A少3倍，比RISC-V内核（处理器B）少1.7倍。良好的执行效率导致ARC EM9D处理器的时钟频率更低，有助于降低智能物联网设备的功耗。

结语

与用户智能互动的智能物联网设备在许多领域均已浮出水面。这些设备越来越多地应用机器学习技术来处理捕获的传感器数据，从而基于识别到的模式采取明智的行动。在本书中，我们介绍了旨在高效实现低/中端机器学习推理的可编程处理器和相关软件库。我们以ARC EM9D处理器为例，展示了对于高效实施机器学习推理引擎至关重要的几项处理器功能，例如矢量MAC指令以及带有高级地址生成单元的XY存储器。这些功能，再加上其他的DSP和控制处理功能，导致ARC EM9D成为低功耗物联网应用的通用核心。完整且经过高度优化处理的embARC MLI库，允许您有效利用ARC EM9D处理器来高效支持各类低/中端机器学习应用。我们还通过CIFAR-10出色的基准测试结果证明了这种效率。总之，具有“感知”、“观察”和“倾听”能力的智能机器已经开始进入我们的生活，Synopsys已经做好准备，随时与您一起欢迎它们的到来。因此，让我们对这些机器说“欢迎”吧！

References

[1] Samuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 3 (3): 210–229.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[3] www.synopsys.com/ev

[4] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. arXiv preprint arXiv:1712.05877, 2017.

[5] https://embarc.org/

[6] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., “Deep speech 2: End-to-end speech recognition in english and mandarin”, in International Conference on Machine Learning, 2016, pp. 173–182.

[7] https://en.wikipedia.org/wiki/Q_(number_format)

[8] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." 2009.

[9] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor. "Caffe: Convolutional Architecture for Fast Feature Embedding." arXiv preprint arXiv:1408.5093. 2014: http://caffe.berkeleyvision.org/

[10] L. Lai, N. Suda, and V. Chandra, “CMSIS-NN: Efficient neural network kernels for ARM Cortex-M CPUs,” arXiv preprint arXiv:1801.06601, 2018. [Online]. Available: https://arxiv.org/abs/1801.06601

[11] https://content.riscv.org/wp-content/uploads/2018/07/Shanghai-1325_GreenWaves_Shanghai-2018-MC-V2.pdf

06/18/19.CS357660975-Machine Learning White Paper-CN.

Pub: May 2019