The Deep Learning Compiler- A Comprehensive Survey 深度学习编译器综述 （一）

The Deep Learning Compiler: A Comprehensive Survey

在不同的深度学习（DL）硬件上部署各种DL模型的困难推动了社区对DL编译器的研究和开发。从工业界和学术界都提出了几个DL编译器，例如Tensorflow XLA和TVM。类似地，DL编译器以不同的DL框架中描述的DL模型作为输入，然后生成适用于各种DL硬件的优化代码作为输出。然而，目前尚没有一篇综述全面地分析DL编译器的独特设计架构。在本文中，我们通过详细剖析常用的设计，特别关注DL导向的多级IR（中间表示）和前后端优化，对现有的DL编译器进行了全面调查。我们对多级IR的设计进行了详细的分析，并介绍了常用的优化技术。最后，我们还强调了一些意见，作为DL编译器潜在的研究方向。这是第一篇专注于DL编译器设计架构的综述论文，我们希望它能为未来的DL编译器研究铺平道路。

1. INTRODUCTION

深度学习（DL）的发展在各个科学领域产生了深远的影响。它不仅在人工智能领域，如自然语言处理（NLP）[64]和计算机视觉（CV）[26]中展示出了显著的价值，而且在更广泛的应用中也取得了巨大成功，如电子商务[36]、智能城市[68]和药物研发[15]。随着卷积神经网络（CNN）[54]、循环神经网络（RNN）[80]、长短期记忆网络（LSTM）[38]和生成对抗网络（GAN）[29]等多功能深度学习模型的出现，为了实现广泛应用，简化各种深度学习模型的编程变得至关重要。

在行业和学术界的持续努力下，已经提出了几个流行的深度学习框架，如TensorFlow [1]、PyTorch [75]、MXNet [16]和CNTK [81]，以简化各种深度学习模型的实现。尽管上述DL框架在设计上存在着优势和劣势，但在支持新兴的DL模型时，提供互操作性变得十分重要，以减少冗余的工程工作。为了提供互操作性，ONNX [66]被提出，它定义了一种统一的DL模型表示格式，以便于在不同的DL框架之间进行模型转换。

与此同时，诸如矩阵乘法等独特的计算特性激发了芯片架构师们设计定制的DL加速器，以提高效率。互联网巨头（例如，Google TPU [44]，Hisilicon NPU [56]，Apple Bonic [49]），处理器供应商（例如，NVIDIA Turing [72]，Intel NNP [41]），服务提供商（例如，Amazon Inferentia [8]，Alibaba Hanguang [7]），甚至初创公司（例如，Cambricon [57]，Graphcore [43]）都在投入大量的人力和资金开发DL芯片，以提高DL模型的性能。通常，DL硬件可以分为以下几类：1）具有软硬件协同设计的通用硬件，2）专门为DL模型量身定制的定制硬件，和3）受生物脑科学启发的神经形态硬件。例如，通用硬件（例如CPU，GPU）已经添加了特殊的硬件组件，如AVX512向量单元和张量核心，以加速DL模型。而对于像Google TPU这样的专用硬件，已经设计了特定的集成电路（例如矩阵乘法引擎和高带宽内存），以将性能和能源效率提升到极致。可预见的未来，DL硬件的设计将变得更加多样化。

为了适应硬件多样性，高效地将计算映射到DL硬件上非常重要。在通用硬件上，高度优化的线性代数库（例如BLAS库）（如MKL和cuBLAS）作为有效计算DL模型的基础。以卷积操作为例，DL框架将卷积转换为矩阵乘法，然后调用BLAS库中的GEMM函数。此外，硬件供应商还发布了专门针对DL计算进行优化的库（例如MKL-DNN和cuDNN），包括前向和反向卷积、池化、归一化和激活等。还开发了更先进的工具来进一步加速DL操作。例如，TensorRT

[73]支持图优化（如层融合）和低比特量化，并提供大量高度优化的GPU核函数。在专用的DL硬件上，也提供了类似的库[43, 57]。然而，依赖于这些库的缺点是它们通常滞后于DL模型的快速发展，因此无法有效利用DL芯片。

为了解决DL库和工具的缺点，以及减轻手动优化每个DL硬件上的DL模型的负担，DL社区已经转向了领域特定编译器。迅速地，业界和学术界提出了几个流行的DL编译器，例如TVM [17]，Tensor Comprehension [91]，Glow [79]，nGraph [21]和XLA [53]。DL编译器以DL框架中描述的模型定义作为输入，并生成适用于各种DL硬件的高效代码实现作为输出。模型定义和具体代码实现之间的转换经过了针对模型规范和硬件架构的高度优化。具体而言，它们结合了面向DL的优化，如层和操作融合，从而实现高效的代码生成。此外，现有的DL编译器还利用了通用编译器成熟的工具链（例如LLVM [51]），可以在不同的硬件架构之间提供更好的可移植性。与传统编译器类似，DL编译器也采用了分层设计，包括前端、中间表示（IR）和后端。然而，DL编译器的独特之处在于多层次IR的设计和针对DL的特定优化。

在本文中，我们通过将编译器设计分为**前端、多层IR和后端**来对现有的DL编译器进行全面调查，并特别强调**IR设计和优化方法**。据我们所知，这是第一篇对DL编译器设计进行全面调查的论文。具体而言，本文有以下贡献：

1. 我们对现有DL编译器常用的设计架构进行了分析，并详细分析了关键设计组件，如多层IR、前端优化（包括节点级、块级和数据流级优化）和后端优化（包括硬件特定优化、自动调优和优化的内核库）。
2. 我们从各个方面提供了对现有DL编译器的全面分类，这对应于本调查中描述的关键组件。这个分类的目标是为从业人员提供关于选择DL编译器的指导，考虑到他们的需求，同时也为研究人员提供DL编译器的全面总结。
3. 我们对CNN模型中的DL编译器进行了定量性能比较，包括完整的模型和轻量级模型。我们比较了端到端和每层（只考虑了卷积层因为它们占据了主要推理时间）的性能，以展示优化的效果。评估脚本和结果已经开源[1]，供参考使用。
4. 我们强调了DL编译器未来发展的几个重要方向，包括动态形状和前/后处理、高级自动调优、多面体模型、子图划分、量化、统一优化、可微分编程和隐私保护等。我们希望这些方向能够推动DL编译器领域的研究进展。

论文的剩余部分组织如下。第2节介绍了DL编译器的背景，包括DL框架、DL硬件，以及针对硬件（FPGA）的特定DL代码生成器。第3节描述了DL编译器的通用设计架构。第4节讨论了DL编译器的关键组件，包括多层IR、前端优化和后端优化。第5节提供了一个全面的分类。第6节提供了定量性能比较。第7节强调了DL编译器研究的未来方向。

2. BACKGROUND

2.1 Deep Learning Frameworks

在本节中，我们概述了流行的DL框架。讨论可能不是详尽无遗的，但旨在为DL从业者提供指导。图1展示了DL框架的格局，包括当前流行的框架、历史框架以及支持ONNX的框架。

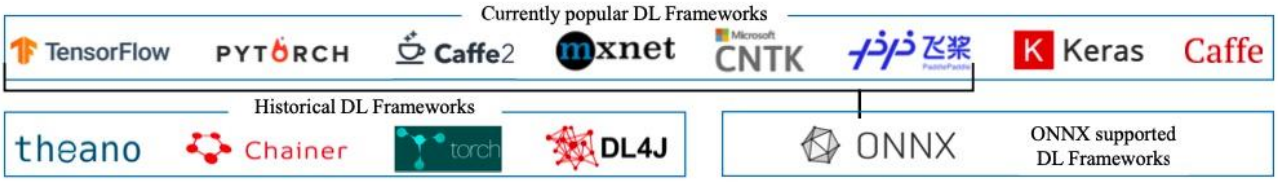
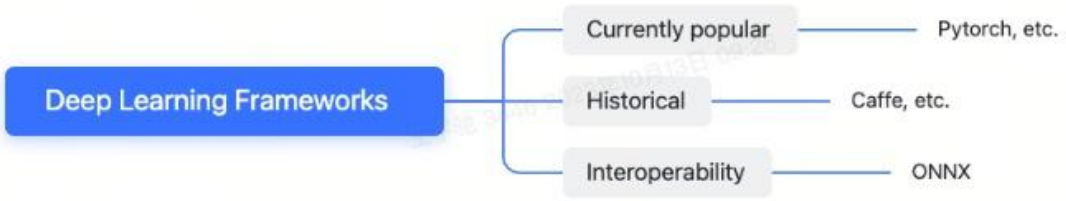
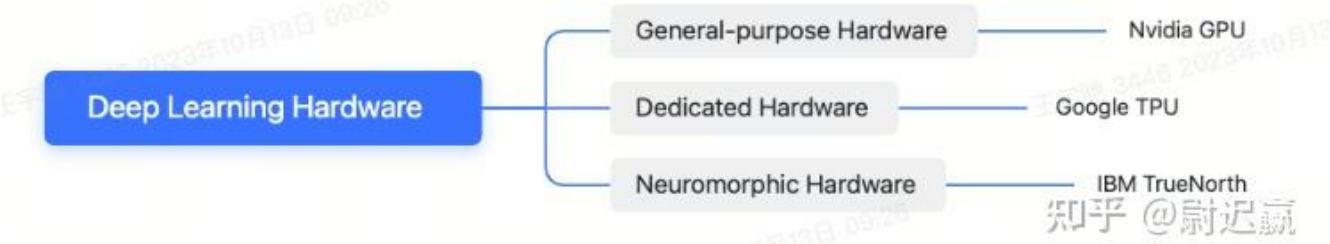


Fig. 1. DL framework landscape: *1)* Currently popular DL frameworks; *2)* Historical DL frameworks; *3)* ONNX supported frameworks.



现在（2023）真正流行的框架实际只剩下了一种：Pytorch，TensorFlow也逐渐式微。
Historical中最经典的当属Caffe，很多人开始接触深度学习都是从Caffe始。
*ONNX的全称叫Open Neural Network Exchange，它的设计目标是为了实现不同深度学习框架之间的模型互操作性，即通过ONNX可以将一个框架训练的模型导出为ONNX格式，然后在另一个框架中加载和运行该模型。ONNX可以使**训练和推理解耦**，不管是什么框架训练的，只要能导出为ONNX格式就可以用ONNX Runtime来进行推理。*
然而，由于不同的深度学习框架支持的操作符和计算规则可能有所不同，为了胶合这么多的框架ONNX也付出了巨大的代价，定义了一系列的opset来描述不同操作符和计算规则的行为。目前onnx的opset版本已经到了19 = =!。

2.2 Deep Learning Hardware



根据通用性，DL硬件可以分为三类：1) 通用硬件，通过硬件和软件优化来支持DL工作负载；2) 专用硬件，专注于通过全面定制的电路设计加速DL工作负载；3) 神经形态硬件，通过模拟人脑的方式来实现功能。

该文把AI芯片分为了以下3种：通用硬件，专用硬件，神经形态硬件。

1. 通用硬件

代表性的通用DL模型硬件是图形处理单元（GPU），它通过众核（many-core）架构实现了高度并行计算。例如，自Volta架构以来，Nvidia GPU引入了张量核心（tensor cores)。张量核心可以在并行计算中加速混合精度的矩阵乘法和累加计算，这在DL模型的训练和推理过程中广泛使用。针对硬件进行了优化，NVIDIA还推出了高度优化的DL库和工具，如cuDNN [18]和TensorRT [73]，以进一步加速DL模型的计算。

最具代表性的就是通用图形处理单元（GPGPU，General-Purpose Graphic Processing Unit）。GPGPU是通用计算在GPU上的应用。GPGPU利用了GPU的并行计算能力，将其应用于通用计算领域，例如科学计算、深度学习、数据分析等。深度学习只是GPGPU的一个应用。

2. 专用硬件

专用硬件完全定制用于DL计算，以提高性能和能效。DL应用和算法的快速扩展促使许多初创公司开发专用的DL硬件（如Graphcore GC2、Cambricon MLU270）。此外，传统硬件公司（如Intel NNP，Qualcomm Cloud AI 100）和云服务提供商（如Google TPU，Amazon Inferentia和Alibaba Hanguang）也投资于这个领域。最著名的专用DL硬件是Google的TPU系列。TPU包括矩阵乘法单元（MXU），统一缓冲区（UB）和激活单元（AU），它由主处理器驱动使用CISC指令。MXU主要由系统阵列组成，用于进行矩阵乘法计算，以实现功耗和面积的高效优化。与CPU和GPU相比，TPU仍然可编程，但使用矩阵作为基本元素，而不是向量或标量。Amazon Inferentia最近也引起了关注。这个芯片有四个NeuroCore，专为张量级操作设计，并且拥有大型片上缓存，以避免频繁访问主内存。

也叫做DLA（Deep Learning Accelerator），是专门为深度学习任务设计和优化的硬件加速器。目的是提供高性能、低功耗和高效能的深度学习计算。由于深度学习任务通常涉及大量的矩阵计算和并行处理，传统的通用处理器（如CPU和GPU）可能无法充分发挥其优势。因此，DLA通过定制化的硬件设计和优化的指令集，能够更好地满足深度学习

的计算需求。其中最具代表性的有Google的TPU（Tensor Processing Unit）和华为的Ascend芯片。

3. 神经形态硬件

神经形态芯片使用电子技术来模拟生物大脑。代表性的产品有IBM的TrueNorth和Intel的Loihi。神经形态芯片（如TrueNorth）的人工神经元之间具有非常高的连接性。神经形态芯片还复制了类似大脑组织的结构：神经元可以同时存储和处理数据。传统芯片将处理器和内存分布在不同的位置，而神经形态芯片通常有许多微处理器，每个处理器都有一小部分本地内存。与TrueNorth相比，Loihi的学习能力更接近大脑。Loihi引入了脉冲时间依赖型突触可塑性模型（STDP），这是一种通过前突触和后突触脉冲的相对时间来调节突触强度的机制。然而，神经形态芯片离大规模商业生产还有很长的距离。尽管如此，在计算机科学领域，神经形态芯片可以帮助捕捉快速、终身学习的过程，这是常规DL模型所忽视的；而在神经科学领域，它们有助于揭示大脑各个部分如何协同工作，创造思维、感受甚至意识

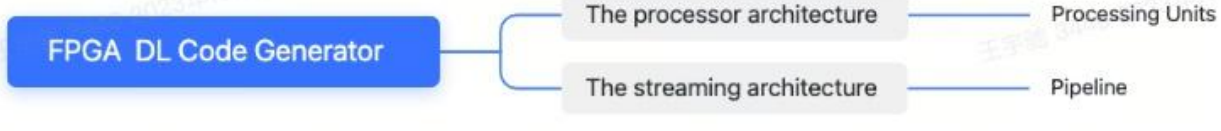
神经形态硬件（Neuromorphic Hardware）是一种通过模仿人脑神经系统的结构和功能来实现高效计算的硬件设备。它受到了神经科学和计算神经科学的启发，旨在模拟和复制生物神经元的工作方式。
看起来很好，然而商用还有哪些问题？

2.3 Hardware-specific DL Code Generator

Field Programmable Gate Arrays (FPGAs) 是一种可编程的集成电路。由于它可编程，低功耗和高性能，FPGA在很多领域比如通信，医疗，图像处理和ASIC原型设计等多个领域都有应用。在深度学习领域，高性能的CPUs和GPUs可编程性高，但功耗高，而低功耗的专用集成电路（ASICs)只能用于特定应用。然而，FPGA可以弥合CPU/GPU和ASIC之间的差距，这使得FPGA成为深度学习的一个有吸引力的平台。

高级综合（HLS）编程模型使得FPGA程序员能够使用高级语言如C和C++方便地生成有效的硬件设计。

针对FPGA的硬件特定代码生成器将DL模型或其领域特定语言（DSLs）作为输入，进行特定于领域（关于FPGA和DL）的优化和映射，然后生成HLS或Verilog/VHDL代码，最终生成比特流。根据生成的基于FPGA加速器的架构，它们可以分为两类：处理器架构和流式架构。



2.3.1 The processor architecture

处理器架构(The processor architecture)与通用处理器有相似之处。这种架构的FPGA加速器通常有多个处理单元（Processing Units (PUs)组成，其中包括片上缓冲区和多个较小的处理引擎(Processing engine, PEs)。它通常有一个虚拟指令集(ISA)，硬件的控制和执行的调度由软件来决定。
DNN Weaver， AngelEye， ALAMO， FP-DNN， SysArrayAccel是针对处理器架构的典型FPGA DL代码生成器。此外，PU和PE通常负责粗粒度的基本操作，如矩阵-向量乘法、矩阵-矩阵乘法、池化和一些元素相关的操作。

2.3.2 The streaming architecture

流架构（The streaming architecture）与管道（pipeline）有相似之处。该体系结构的FPGA加速卡由多个不同的硬件块组成，几乎每层输入DL模型都有一个硬件块。这种加速器以DL模型的输入数据为基础，通过不同的硬件块按相同的顺序分层处理数据。此外，通过流式输入数据，所有硬件块都可以以流水线方式充分利用。fpgaConvNet、DeepBurning、Haddoc2、AutoCodeGen是典型的对应DL代码生成器。

3. COMMON DESIGN ARCHITECTURE OF DL COMPILERS

DL编译器的常见设计架构主要包含两个部分：编译器前端和编译器后端，如图2所示。中间表示（IR）在前端和后端之间分布。通常，IR是程序的抽象，用于程序的优化。具体而言，DL模型在DL编译器中被转换为多级IR，其中高级IR位于前端，低级IR位于后端。基于高级IR，编译器的前端负责硬件无关的转换和优化。基于低级IR，编译器的后端负责硬件特定的优化、代码生成和编译。需要注意的是，本调查主要关注DL编译器的设计原理。对于DL编译器的功能和实验比较，读者可以参考[55，102]

DL编译器一般包括两部分，前端（frontend）和后端（backend）。前端对应highl-level IR，后端对应low-level IR。
基于high-level IR，编译器前端负责hardware-indendent（硬件无关的）的转换和优化。基于low-level IR，编译器后端负责hardware-specific（硬件相关的）优化。

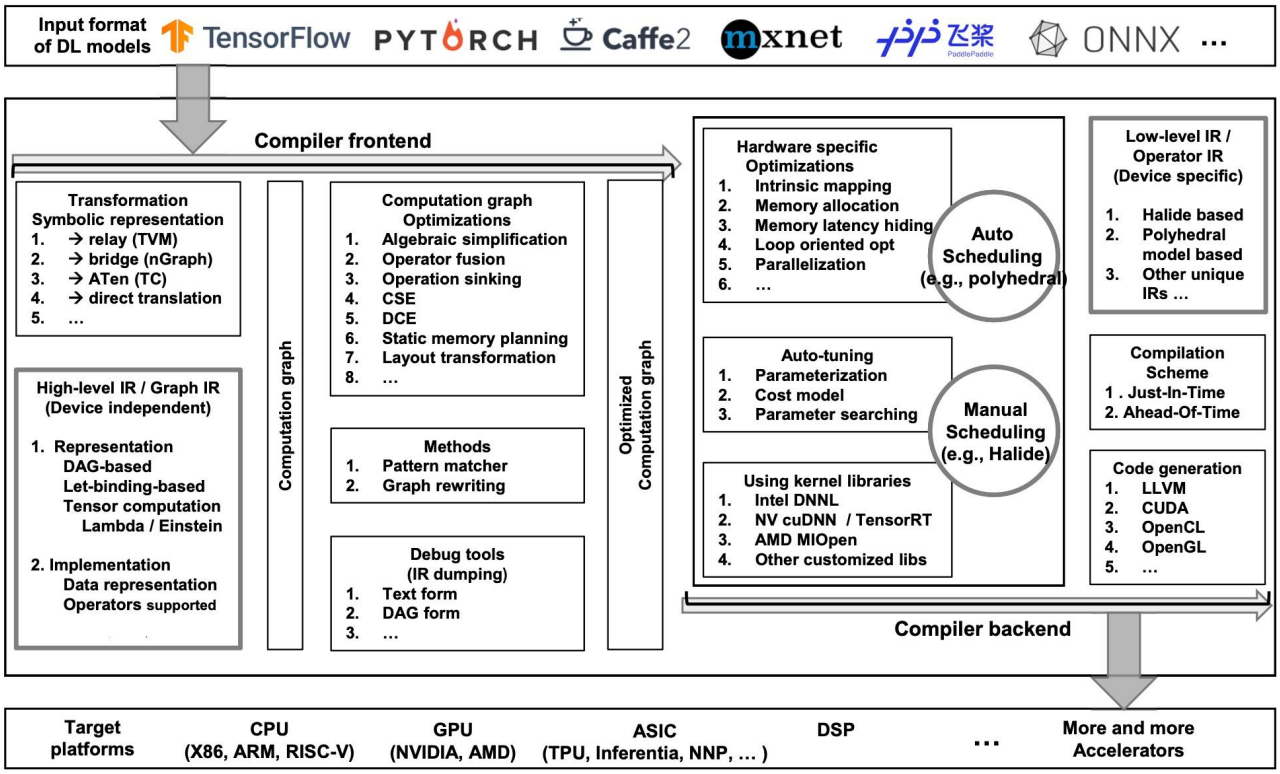
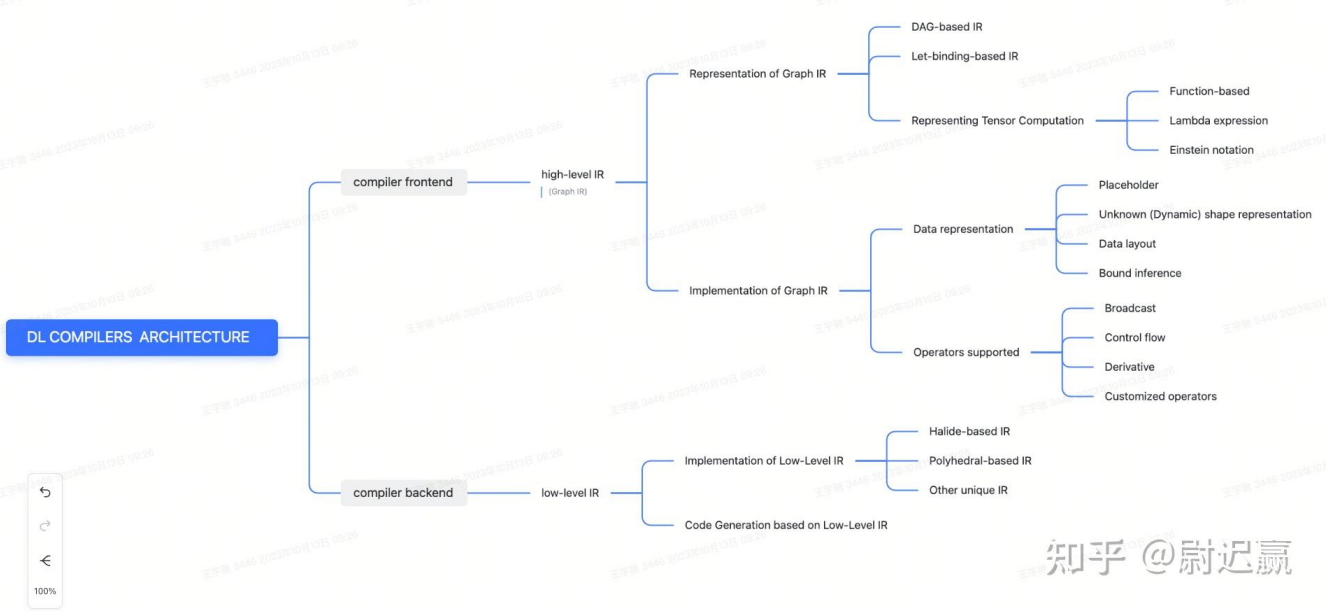


Fig. 2. The overview of commonly adopted design architecture of DL compilers.



3.1 The high-level IR

高级IR，也称为图形IR，表示计算和控制流，并且与硬件无关。高级IR的设计挑战是能够抽象出计算和控制流，以捕捉和表达各种深度学习模型。高级IR的目标是建立操作符和数据之间的控制流和依赖关系，同时提供图级优化的接口。它还包含用于编译的丰富语义信息，并提供自定义操作符的可扩展性。

3.2 The low-level IR

低级IR旨在针对不同硬件目标进行硬件特定优化和代码生成。因此，低级IR应该足够细粒度，以反映硬件特性并表示硬件特定优化。它还应该允许在编译器后端中使用成熟的第三方工具链，如Halide、polyhedral model和LLVM。

3.3 The frontend

前端接收来自现有深度学习框架的深度学习模型作为输入，然后将模型转换为计算图表示（例如图形IR）。前端需要实现各种格式转换，以支持不同框架中的多样化格式。计算图优化综合了通用编译器和深度学习特定优化的技术，以减少冗余并提高图形IR的效率。这些优化可以分为**节点级优化（例如无操作消除和零维张量消除）、块级优化（例如代数化简、操作融合和操作下沉）和数据流级优化（例如CSE、DCE、静态内存规划和布局转换）**。在前端完成后，优化后的计算图生成并传递给后端。

3.4 The backend

后端将高级IR转换为低级IR，并执行硬件特定的优化。一方面，它可以直接将高级IR转换为第三方工具链（例如LLVM IR），以利用现有的基础设施进行通用优化和代码生成。另一方面，它可以利用对DL模型和硬件特性的先前了解，通过定制化编译通路进行更高效的代码生成。常用的硬件特定优化包括**硬件内在映射、内存分配和读取、内存延迟隐藏、并行化以及循环优化**。为了确定大型优化空间中的最佳参数设置，现有的DL编译器广泛采用两种方法，即**自动调度（例如多面体模型）和自动调优（例如AutoTVM）**。优化后的低级IR使用JIT或AOT进行编译，为不同的硬件目标生成代码。