(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2021 年 1 月 7 日 (07.01.2021)

WIPO | PCT

(10) 国际公布号
WO 2021/000970 A1

(51) 国际专利分类号:
G06F 8/41 (2018.01)

(21) 国际申请号: PCT/CN2020/111068

(22) 国际申请日: 2020 年 8 月 25 日 (25.08.2020)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201910596132.8 2019年7月3日 (03.07.2019) CN

(71) 申请人: 安徽寒武纪信息科技有限公司 (**ANHUI CAMBRICON INFORMATION TECHNOLOGY CO., LTD.**) [CN/CN]; 中国安徽省合肥市高新区习友路3333号中国（合肥）国际智能语音产业园研发中心楼611-194室, Anhui 231283 (CN)。

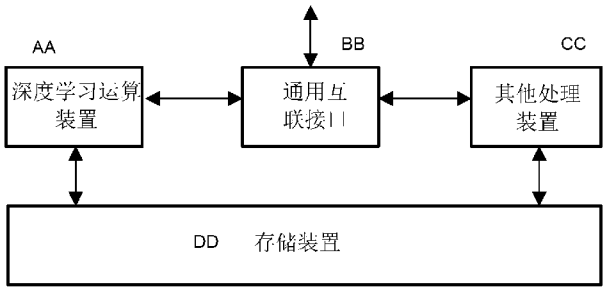
(72) 发明人: 陈黎明 (**CHEN, Liming**); 中国安徽省合肥市高新区习友路3333号中国（合肥）国际智能语音产业园研发中心楼611-194室, Anhui 231283 (CN)。 吴林阳 (**WU, Linyang**); 中国安徽省合肥市高新区习友路3333号中国（合肥）国际智能语音产业园研发中心楼611-194室, Anhui 231283 (CN)。 王子毅 (**WANG, Ziyi**); 中国安徽省合肥市高新区习友路3333号中国（合肥）国际智能语音产业园研发中心楼611-194室, Anhui 231283 (CN)。

(74) 代理人: 北京林达刘知识产权代理事务所（普通合伙）(**LINDA LIU & PARTNERS**); 中国北京市东城区北三环东路36号北京环球贸易中心C座16层, Beijing 100013 (CN)。

(81) 指定国(除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG,

(54) **Title:** DEEP LEARNING ALGORITHM COMPILING METHOD, DEVICE, AND RELATED PRODUCT.

(54) 发明名称: 深度学习算法的编译方法、装置及相关产品



AA Deep learning operation device
BB Universal connection interface
CC Other processing device
DD Storage device

图31

(57) **Abstract:** The present disclosure relates to an deep learning algorithm compiling method and a device and a related product, the product comprising a controller unit, and the controller unit comprising: an instruction cache unit, an instruction processing unit and a queue-storing unit. The instruction cache unit is used to store computing instructions associated with artificial neural network operations. The instruction processing unit is used to resolve the computing instructions to obtain a plurality of operation instructions. The queue-storing unit is used to store an instruction queue, which comprises: a plurality of operation instructions or computing instructions to be executed according to the front-to-rear sequence of the queue. By means of the described method, the present disclosure may improve the operation efficiency of the related product when carrying out neural network model operations.

(57) 摘要: 本公开涉及深度学习算法的编译方法、装置及相关产品，所述产品包括控制器单元，所述控制器单元包括：指令缓存单元、指令处理单元和存储队列单元；所述指令缓存单元，用于存储所述人工神经网络运算关联的计算指令；所述指令处理单元，用于对所述计算指令解析得到多个运算指令；所述存储队列单元，用于存储指令队列，该指令队列包括：按该队列的前后顺序待执行的多个运算指令或计算指令。通过以上方法，本公开可以提高相关产品在进行神经网络模型的运算时的运算效率。

WO 2021/000970 A1

[见续页]

BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) **指定国**(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布：

— 包括国际检索报告(条约第21条(3))。
— 在修改权利要求的期限届满之前进行，在收到该修改后将重新公布(细则48.2(h))。
— 包括关于请求恢复一项或多项优先权要求的信息(细则26之二.3和48.2(b)(vii))。

深度学习算法的编译方法、装置及相关产品

技术领域

本公开涉及深度学习领域，尤其涉及一种深度学习算法的编译方法、装置及相关产品。

背景技术

在人工智能技术领域，神经网络算法是最近非常流行的一种机器学习算法，在各种领域中都取得了非常好的效果，比如图像识别，语音识别，自然语言处理等。随着神经网络算法的发展，算法的复杂度也越来越高，为了提高识别度，模型的规模也在逐渐增大。

发明内容

有鉴于此，本公开提出了一种深度学习算法的编译方法、装置及相关产品，能够提升深度学习算法针对相应硬件平台的性能优化效果。

根据本公开的第一方面，提供了一种深度学习算法的编译方法，所述方法包括：接收深度学习编程库接口传递的操作数据；获取所述操作数据包括的操作指令；判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

根据本公开的第二方面，提供了一种深度学习算法的编译装置，包括：操作数据接收模块，用于接收深度学习编程库接口传递的操作数据；操作指令获取模块，用于获取所述操作数据包括的操作指令；编译模块，用于判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

根据本公开的第三方面，提供了一种深度学习运算装置，所述深度学习运算装置包括如上述第二方面所述的深度学习算法的编译装置，所述深度学习运算装置用于完成设定的深度学习运算。

根据本公开的第四方面，提供了一种组合运算装置，所述组合运算装置包括如上述第三方面所述的深度学习运算装置，通用互联接口和其他处理装置；所述深度学习运算装置与所述其他处理装置进行交互，共同完成用户指定的计算操作。

根据本公开的第五方面，提供了一种深度学习芯片，所述深度学习芯片包括：如上述第二方面所述的深度学习算法的编译装置；或者，如上述第三方面所述的深度学习运算装置；或者，如上述第四方面所述的组合运算装置。

根据本公开的第六方面，提供了一种电子设备，所述电子设备包括：如上述第二方面所述的深度学习算法的编译装置；或者，如上述第三方面所述的深度学习运算装置；或者，如上述第四方面所述的组合运算装置；或者，如上述第五方面所述的深度学习芯片。

通过接收深度学习编程库接口传递的操作数据，根据操作数据中操作指令的指令类型，执行与指令类型相应的编译操作，得到深度学习算法的二进制代码，根据本公开的各方面实施例的深度学习算法的编译方法、装置及相关产品，能够使得编译过程根据操作指令的类型不同而适应性地发生改变，从而大大提升编译的灵活性和编译效率，有效的提升深度学习算法针对相应硬件平台的性能优化效果，继而提升深度学习处理器的处理性能。

根据下面参考附图对示例性实施例的详细说明，本公开的其它特征及方面将变得清楚。

附图说明

包含在说明书中并且构成说明书的一部分的附图与说明书一起示出了本公开的示例性实施例、特征和方面，并且用于解释本公开的原理。

图1示出根据本公开一实施例的深度学习算法的编译方法的流程图。

图2示出根据本公开一实施例的神经演算编程库接口的总体架构示意图。

图3示出根据本公开一实施例的张量数据的属性、分类以及含义之间的对应关系图。

图4示出根据本公开一实施例的张量数据与形状符号之间的对应关系图。

图5示出根据本公开一实施例的NCLAPI支持的build-in操作指令图。
图6示出根据本公开一实施例的精度自动选优的实现方式示意图。
图7示出根据本公开一实施例的计算模型示意图。
图8示出根据本公开一实施例的定制操作指令的生成原理图。
图9示出根据本公开一实施例的操作融合结果示意图。
图10示出根据本公开一实施例的操作融合结果示意图。
图11示出根据本公开一实施例的操作融合的相关编程接口的示意图。
图12示出根据本公开一实施例的创建融合操作的过程示意图。
图13示出根据本公开一实施例的三层计算模型的数据流示意图。
图14示出根据本公开一实施例的混合式编程模型的实现框图。
图15示出根据本公开一实施例的离线模式和在线模式的区别示意图。
图16示出根据本公开一实施例的离线接口的示意图。
图17示出根据本公开一实施例的TensorFlow的架构图。
图18示出根据本公开一实施例的NCLAPI与主流深度学习编程库接口的对比示意图。
图19示出根据本公开一实施例的NCLA的整体架构示意图。
图20示出根据本公开一实施例的静态操作池的实现方式示意图。
图21示出根据本公开一实施例的CDUCA的架构示意图。
图22示出根据本公开一实施例的原始计算图的形式。
图23示出根据本公开一实施例的计算图引擎的工作流程图。
图24示出根据本公开一实施例的图像分类网络内包含子结构的结构示意图。
图25示出根据本公开一实施例的深度学习算法的编译方法的流程图。
图26示出根据本公开一实施例的指令流水示意图。
图27示出根据本公开一实施例的对模型数据进行优化的实现方式图。
图28示出根据本公开一实施例的对数据拆分后优化的示意图。
图29示出根据本公开一实施例的运行时系统的模块及功能示意图。
图30示出根据本公开一实施例的深度学习算法的编译装置的框图。
图31示出根据本公开一实施例的组合处理装置的框图。

具体实施方式

以下将参考附图详细说明本公开的各种示例性实施例、特征和方面。附图中相同的附图标记表示功能相同或相似的元件。尽管在附图中示出了实施例的各种方面，但是除非特别指出，不必按比例绘制附图。

在这里专用的词“示例性”意为“用作例子、实施例或说明性”。这里作为“示例性”所说明的任何实施例不必解释为优于或好于其它实施例。

另外，为了更好的说明本公开，在下文的具体实施方式中给出了众多的具体细节。本领域技术人员应当理解，没有某些具体细节，本公开同样可以实施。在一些实例中，对于本领域技术人员熟知的方法、手段、元件和电路未作详细描述，以便于凸显本公开的主旨。

为了缓解恶化的存储墙问题，深度学习处理器通常会在靠近运算单元的位置设计片上存储（on-chip memory），访问片上存储的延时远低于访问片外存储的延时，因此合理地利用片上存储是发挥深度学习处理器性能的关键。然而，片上存储不具备数据预取、数据替换、处理数据冲突等cache所具备的 功能，这些工作必须由编程人员编写指令来完成。另一方面，片上存储容量非常有限，编程人员必须对运算和数据同时进行拆分，这导致运算和数据之间紧密耦合。以某一深度学习处理器为例，其片上神经元存储（NBin）容量只有2KB，而一幅1080P的RGB图像以半精度表示需要占用12MB空间，因此仅输入数据就需要拆分成6K次进行load，数据拆分导致运算也要跟着拆分，比如循环嵌套的层数会增加，内层循环的迭代次数会减少。

片上存储的特点（显示管理，容量有限）和深度学习算法的特点（处理层复杂多变、运算数据类型多样化、处理高维张量等）共同导致了程序优化对算法和硬件变动极其敏感。以2D卷积操作为例，它至少包含9种形状参数 （N，CI，HI，WI，CO，Kh，Kw，Sh，Sw）、7重嵌套循环、多种运算精度 （half，fxm.b，Qts.b，intx等），上述参数的组合或者片上存储的容量发生变化都会影响数据和运算的最优拆分策略，导致生成的二进制代码不同。

通过上述原因可以看出，由于深度学习算法的特殊性以及深度学习处理器架构的特殊性，使得深度学习领域的程序优化有两个重要特点，一是对算法和硬件的变动极其敏感，二是运算和数据间耦合度极高。由于深度学习领域的程序优化对算法和硬件变动极其敏感，我们很难通过提前编译优化（AOT）来满足不同算法、不同硬件平台下的性能需求。因此，如何基于深度学习算法的特殊性和深度学习处理器架构的特殊性，提出一种面向深度学习算法的编译方法，成为一个亟待解决的问题。

图1示出根据本公开一实施例的深度学习算法的编译方法的流程图。如图所示，该方法可以包括：

步骤S11，接收深度学习编程库接口传递的操作数据。

步骤S12，获取操作数据包括的操作指令。

步骤S13，判断操作指令的指令类型，根据判断结果执行与指令类型相应的编译操作，得到深度学习算法的二进制代码。

上述公开实施例中，二进制代码为用于指导硬件设备执行深度学习算法的硬件指令，具体指导哪些硬件设备，以及硬件指令具体的内容在本公开实施例中均不受限制，可以根据实际情况灵活选择。

通过接收深度学习编程库接口传递的操作数据，根据操作数据中操作指令的指令类型，执行与指令类型相应的编译操作，得到深度学习算法的二进制代码，根据本公开的各方面实施例的深度学习算法的编译方法、装置及相关产品，能够使得编译过程根据操作指令的类型不同而适应性地发生改变，从而大大提升编译的灵活性和编译效率，有效的提升深度学习算法针对相应硬件平台的性能优化效果，继而提升深度学习处理器的处理性能。

步骤S11中接收的深度学习编程库接口传递的操作数据，其具体的实现来源不受限定。在一种可能的实现方式中，操作数据可以根据深度学习编程库接口接收的用户指令创建或调用。

通过根据深度学习编程库接口接收的用户指令，创建或调用操作数据，且该操作数据可以用于后续得到深度学习算法的二进制代码，通过上述过程，可以为用户提供一种通用的编程接口，实现用户指令与机器指令之间的有效转换。

上述步骤中，深度学习编程库接口的实现方式不受限定，可以根据实际情况进行灵活选择。在一种可能的实现方式中，深度学习编程库接口可以是神经演算编程库接口（NCLAPI，neurological calculus Library API），该接口的具体实现方式可以根据实际情况进行确定，不局限于下述公开实施例。图2示出根据本公开一实施例的神经演算编程库接口的总体架构示意图，如图所示，在一个示例中，该NCLAPI接口的实现方式可以为：通过模拟神经演算来使NCLAPI接口具备良好的深度学习建模能力；通过设计可重塑操作和相应的操作规则来灵活地支持各种性能优化；通过设计混合式编程模型来提高编程模型的灵活性；简化数据结构和接口的设计，将硬件细节隐藏在数据结构和接口内部。

上述公开实施例中，神经演算是一种函数式的深度学习建模方法，在一个示例中，神经演算可以用张量来表示输入层的输入数据、输出数据以及模型参数，用函数来表示深度学习处理层，函数可以按照一定的规则进行函数组合，从而构造出各种深度学习计算模型。由于函数本身就具备可组合性和可重用性，因此神经演算可以很好地表达深度学习算法的可组合性和可重用性。按照上述思路设计的神经演算具有强大的深度学习建模能力。目前已知的Tensor、Mxnet等深度学习框架都采用有向图对深度学习计算模型进行建模，通过实验可以证明：任意有向图能够映射成函数组合，任意函数组合能够映射成有向无环图，因此神经演算具备与有向图同等的深度学习建模能力。由于在一个示例中，可以通过模拟神经演算来使NCLAPI接口具备良好的深度学习建模能力，因此，在适当的模拟方式下，NCLAPI可以具备与有向图同等的学习建模能力。

通过图2可以看出，在一个示例中，NCLAPI可以存在两种数据结构，分别为张量（nclTensor）和可塑操作（nclOperator），nclTensor用来描述深度学习处理层的输入数据、输出数据以及模型参数，

nclOperator用来描述深度学习处理层。这两种数据结构模拟自神经演算的张量和函数，并根据实际的编程模型进行了一些修改和扩展。

通过步骤S11可以看出，本公开实施例提出的深度学习算法的编译方法，首先需要接收深度学习编程库接口传递的操作数据，而上述公开实施例中又提出，在一个示例中，深度学习编程库接口可以是NCLAPI ，NCLAPI中可以存在张量和可塑操作两种数据结构，因此，在一种可能的实现方式中，深度学习编程库接口传递的操作数据中可以为与nclTensor数据结构对应的张量数据，也可以为与nclOperator数据结构对应的操作指令，也可以同时包含张量数据和操作指令。

在一种可能的实现方式中，NCLAPI中的张量数据，是对多维数据的抽象表示，可以用来表示深度学习处理层的输入数据、输出数据以及模型参数，在一个示例中，卷积层的输入、输出以及权值数据都可以表示成张量数据。因此，在一种可能的实现方式中，张量数据可以具有如下特点：包含多种属性；可以描述标量、向量、矩阵、张量等多维数据；遵循一定的命名规则；可以描述多种数据类型。

上述公开实施例中提出张量数据遵循一定的命名规则，这一命名规则可以根据实际情况进行灵活设定，并不局限于下述公开实施例，在一种可能的实现方式中，张量数据遵循的命名规则可以为：只能由字母、数字、下划线组成；第一个字符必须是英文字母；不可以包含标点符号和类型说明符。

由于深度学习计算模型通常处理固定大小的数据，以图像分类模型AlexNet为例，它每一个处理层的输入和输出数据形状都是固定的，而数据的取值会随着输入频繁改变，因此，数据的属性和数据的值具有完全不同的更新频率。从数据结构复用以及编程灵活性两个角度出发，应该把数据的属性和数据的值进行解耦合，因此，在一种可能的实现方式中，本公开实施例中的张量数据仅用来描述数据的属性，而数据的值可以用指向内存区域的指针来描述数据的值，通过张量数据和指针的组合来完整地映射神经演算张量。

上述公开实施例中已经提出，张量数据的特点为可以包含多种属性，具体包含哪些属性，可以根据实际情况进行灵活设定和选择。在一种可能的实现方式中，张量数据可以包括形状属性（shape）、逻辑数据类型属性（dtype）、物理数据类型属性（pdtype）和物理布局属性（layout）。

通过对张量数据设定形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性这四种属性，可以足够并且很好地描述深度学习算法中的各种数据，从而使得编译方法可以较好地适应于深度学习算法中的各种数据情况，提升编译方法的通用性。

在应用过程中，还可以基于对张量数据的使用情况，对上述属性进行进一步的分类，具体的分类方式同样可以根据实际情况进行设定。图3示出根据本公开一实施例的张量数据的属性、分类以及含义之间的对应关系图。如图所示，在一个示例中，可以将上述四种属性分为两类：分别为可见属性和不可见属性，其中，形状属性和逻辑数据类型属性可以归类为可见属性，实际使用过程中，可以通过张量赋值接口，对可见属性进行设置；物理数据类型属性和物理布局属性可以归类为不可见属性，可以在编程库的内部进行维护、修改以及使用，继而可以对外界屏蔽硬件细节，降低编程复杂程度。

通过图3可以看出，物理数据类型属性可以用以表明数据在硬件设备内存中存储的精度，而逻辑数据类型属性可以用以表明数据在主机内存中存储的精度，因此，物理数据类型属性和逻辑数据类型属性，二者代表的精度可以相同也可以不同。在一种可能的实现方式中，物理数据类型数据和逻辑数据类型属性可以不同，此时可以设定编译过程实现精度自动选优功能，即在编译过程中，可以自动选择运行速度最快的数据类型进行运算，且该过程可以对用户透明。精度自动选优功能的具体实现过程可以根据实际情况进行确定，在后续公开实施例中会具体说明。

上述公开实施例中同样提出，张量数据可以描述多种数据类型，具体可以描述哪些数据类型可以根据实际情况灵活决定，在一种可能的实现方式中，张量数据可以描述低位宽、量化等数据类型。为了使得张量数据可以支持低位宽和量化，本公开实施例为张量数据设计了不同的数据类型（包括逻辑数据类型和物理数据类型），包括：

双精度浮点：double； 单精度浮点：float；半精度浮点：half；定点：fxm.b（m表示整数位数，b表示总位数）；量化：Qts.b（s表示张量的缩放因子scale，b表示张量的bias）；整型：intx；无符号整型：uintx。

深度学习算法可以支持对图像进行按通道量化，每个通道的scale和bias都可以不同。虽然按通道量化无法用Qts.b来描述，但是可以使用NCLAPI提供的scale和add操作来替代实现，因此NCLAPI对于量化的表达能力仍然是完备的。此外，考虑到将来还可能出现其它数据类型，NCLAPI还可以支持对张量数据描述的数据类型进行扩展。

上述公开实施例中已经提出，nclTensor用来描述深度学习处理层的输入数据、输出数据以及模型参数，由于张量数据与nclTensor相对应，而最常见深度学习处理层是卷积、池化、RNN，它们的输入数据、输出数据以及模型参数都是高维数据，图4示出根据本公开一实施例的张量数据与形状符号之间的对应关系图，如图所示，在一种可能的实现方式中，可以根据图示的对应关系，对深度学习中操作的张量数据形状进行约定。

上述公开实施例中已经提出，操作数据可以根据深度学习编程库接口接收的用户指令创建或调用，而张量数据作为操作数据的一种可能的实现方式，因此可以被创建或调用，张量数据在使用前必须先创建，具体的创建和调用的过程可以根据实际情况进行灵活设定。在一种可能的实现方式中，调用过程可以为对张量数据进行赋值。在一种可能的实现方式中，创建过程可以为在创建张量数据时初始化其可见属性。在一种可能的实现方式中，由于Tensor等深度学习框架把数据对象创建和属性设置进行了解耦合，为了避免在不破坏深度学习框架代码结构的前提下被深度学习框架集成进去，创建过程可以为首先创建一个未初始化的张量数据，后续再调用张量赋值接口（nclSet-TensorAttr）进行属性赋值。

在一种可能的实现方式中，NCLAPI中的操作指令，是对变换的抽象表示，它可以用来表示深度学习处理层，也可以用来表示通用计算，在一个示例中，操作指令可以用来表示深度学习处理层，比如，卷积、池化、全连接等。在本公开实施例中，操作指令执行的操作，可以被统一称为可塑操作。

在一种可能的实现方式中，操作指令可以由三部分组成，分别为输入参数（input params）、输出参数（output params）以及操作类型（OpType），其中，输入参数与该变换的输入张量集合对应，即输入参数可以为所有的输入数据对应的nclTensor和指针。输出参数与该变换的输出张量集合对应，即输出参数可以为所有的输出数据对应的nclTensor和指针。输入参数和输出参数的实现方式不受限定，在一个示例中，可以规定一个操作指令允许以零个或多个（张量数据，指针）作为输入参数，以一个或多个（张量数据，指针）作为输出参数。

操作类型可以用来指定操作指令进行何种数据变换。用户可以在创建操作指令时指定不同的操作类型，这些操作类型可以表达值变换、属性变换、空变换三种类型的数据变换，因此，操作指令不仅可以描述深度学习处理层，还可以描述如数据分割、数据拼接、尺寸缩放等通用计算。

对于深度学习编程库接口来说，其自身可以提供一系列事先定义好的自有操作指令，在本公开实施例中，对于NCLAPI接口来说，这些操作指令可以被称为build-in操作指令（build-in operators），图5示出根据本公开一实施例的NCLAPI支持的build-in操作指令图，从图中可以看出，支持原地算法的操作指令即为NCLAPI支持的build-in操作指令。

操作指令的性质可以根据实际情况灵活设定，在一种可能的实现方式中，为了使得程序的行为更易分析和预测，除了build-in操作指令外，其余的操作指令均可以具有单向性、非相交性和幂等性，其中，单向性表明操作指令不改变输入参数（包括张量数据和指针指向的数据），非相交性表明操作指令的输入参数和输出参数不得同名，幂等性表明操作指令调用的结果只取决于输入参数，不受调用次数的影响。

上述公开实施例中已经提出，操作数据可以根据深度学习编程库接口接收的用户指令创建或调用，而操作指令作为操作数据的一种可能的实现方式，因此可以被创建或调用，具体的创建和调用的过程可以根据实际情况进行灵活设定。在一种可能的实现方式中，操作指令需要先创建再调用，操作指令调用是指将一个操作指令映射到深度学习处理器上执行，操作指令默认支持操作参数运行时可变，具体来说，操作指令创建只需要进行一次，而操作指令调用可以反复进行，且每次操作指令调用时可以指定不同的输入参数和输出参数。

在一种可能的实现方式中，为了优化程序性能，对于NCLAPI接口来说，在执行操作指令调用时接口可以支持两个重要功能，分别为异步执行和自动精度选优。

在一个示例中，异步执行是指操作指令调用函数被主机侧调用后会立即返回，CPU可以在深度学习处理器进行计算的同时执行其它操作，从而提高系统的整体利用率和程序性能。为了确保异步调用执行完成，NCLAPI提供了设备同步接口nclSyncDevice ，它会阻塞CPU的执行，直到设备执行完操作。

图6示出根据本公开一实施例的精度自动选优的实现方式示意图，如图所示，在一个示例中，精度自动选优是指操作指令在设备上执行之前，编程库会自动选择执行时间最短的数据类型，将原始数据转成最优的格式后再进行运算。精度自动选优会协同考虑数据格式转换和操作指令执行两者的时间开销，确保操作的整体执行时间最短。此外，为了满足操作的单向性和幂等性，编程库会申请临时空间来完成数据格式转换，确保不会覆盖原始输入数据。

在一种可能的实现方式中，操作指令还可以支持操作连接，操作连接是指将一个操作指令A的输出参数作为另一个操作指令B的输入参数，B在A完成计算后接着处理A的输出数据。两个操作指令A和B可以连接的充分必要条件是：它们至少各自存在一个输出张量T1和一个输入张量T2，T1和T2的属性完全一致。操作连接具有方向性，连接的方向是从提供数据的操作到使用数据的操作。

在一种可能的实现方式中，深度学习计算模型可以表示成函数组合，一个函数组合是一个只存在单向函数连接（在函数组合中，函数连接的方向只能从左到右）的函数序列。由于操作指令可以由函数映射得到，因此一个深度学习计算模型可以表示成一个只存在单向操作连接的操作指令序列（在操作指令序列中，操作指令连接的方向只能从左到右），在本公开实施例中，将这种操作指令序列称为单向操作序列。可以按照一定的算法将有向图转换成单向操作序列，此外，为了避免原地算法破坏操作的单向性，可以使用张量别名技术来消除原地操作。

在一个示例中，将有向图转换成单向操作指令序列的算法可以为：先将有向图g转换成有向无环图g'。然后对有向无环图g'进行拓扑排序，得到：

$$g''\left(V\left\{vertex_1, vertex_2, ..., vertex_n\right\}, E\right)$$

将图g"中的顶点按顺序映射成操作指令序列；将g"中的边映射成张量，将流出顶点的有向边加入该顶点的输出参数，将流入顶点的边加入该顶点的输入参数。

在一个示例中，张量别名技术的实现形式可以为：

$$func:\{t\}\rightarrow\{t\}\Rightarrow func:\{t\}\rightarrow\{t'\}$$

图7示出根据本公开一实施例的计算模型示意图，如图所示，可以将该计算模型表示成（conv，pool，bn，relu，add）和（conv，pool，relu，bn，add）两种单向操作指令序列，按单向操作指令序列中操作指令出现的顺序对操作指令进行调用，就可以完成一次计算模型执行，需要注意的是，由于relu和add操作存在从右到左的操作连接，因此上述计算模型不可以表示为（conv，pool，add，bn，relu）的操作指令序列。

在一种可能的实现方式中，由于深度学习算法通常处理固定大小的数据，因此可以通过固定操作指令的参数来实现性能优化。操作指令默认支持所有参数在运行时可变，为了利用固定参数来优化操作性能，本公开实施例为操作指令设计了参数绑定（parameter binding）和操作特例化（operator specialization）功能。参数绑定是指固定一个操作指令的部分输入参数或者所有输入参数；操作特例化是指将一个进行过参数绑定的操作指令转换成一个新的操作指令，则该新的操作指令可以被称为特例化操作指令。特例化操作指令仍然满足操作指令的定义、性质，且支持操作指令的所有功能（支持特例化、融合等）。特例化指令的分类方式可以根据实际情况灵活设定，在一种可能的实现方式中，可以根据参数绑定的数量将操作特例化操作指令进行划分，在一个示例中，特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，

完全特例化操作指令包括，对操作指令绑定所有输入参数后转换得到的操作指令；

部分特例化操作指令包括，对操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于操作指令的输入参数数量的正整数；

伪特例化操作指令包括，对操作指令不绑定输入参数后直接转换得到的操作指令。

通过上述公开实施例可以看出，在本公开示例中，特例化操作指令被分为三类：绑定一个操作的所有输入参数是完全特例化操作指令，绑定部分输入参数是部分特例化操作指令，不绑定任何输入参数是伪特例化操作指令。被绑定的参数可以从操作指令的输入参数中删除，用户在调用被绑定参数的操作指令时不需要指定被绑定的参数，因此完全特例化操作指令可以不指定任何输入参数。

通过参数绑定和特例化操作指令可以使得在编译过程中，通过对程序进行部分求值优化，从而减少操作指令的运行时间。参数绑定和特例化操作指令的具体实现方式不受限定，在一种可能的实现方式中，可以通过nclSpecializeOperator接口负责实施特例化操作指令，它可以对操作进行即时编译优化，返回硬件设备执行时间更短的特例化操作指令。在一个示例中，可以存在某一卷积操作，该卷积操作的输入、输出和权值的属性以及取值完全确定，此时可以用参数绑定和操作特例化来生成一个运行速度更快的卷积操作。参数绑定和特例化操作指令可以广泛应用于真实的计算模型中，在一个示例中，深度学习计算模型通常处理固定大小的数据，因此可以对张量的形状进行参数绑定， 然后进行操作特例化，得到特例化操作指令，从而优化程序的性能。在一个示例中，对于推理应用场景，权值是提前训练好的常量，因此可以对操作的权值数据进行参数绑定，然后进行操作特例化，得到特例化操作指令，从而优化程序性能。

在一种可能的实现方式中，深度学习编程库通常只支持使用频率高、耗时长的处理层 （比如卷积、全连接、RNN、池化、激活），导致编程库无法很好地支持端到端执行。为了解决上述问题，本公开实施例为操作指令设计了操作定制功能，使其具备可定制特性。操作定制是指用领域专用编程语言编写一个操作，然后以二进制代码的形式插入到编程库中，在本公开实施例中，将这种操作称为定制操作指令（customized operator）。定制操作指令仍然满足操作指令的定义、性质，且支持操作指令的所有功能（支持特例化、融合等）。

由于定制操作指令以二进制代码的形式插入到编程库中，因此表明定制操作指令需要预先进行编译生成二进制代码，定制操作指令对应的二进制代码的生成过程可以根据NCLAPI和所在的深度学习编程库的实际情况灵活决定，图8示出根据本公开一实施例的定制操作指令的生成原理图，如图所示，在一种可能的实现方式中，定制操作指令对应的二进制代码的生成过程，可以包括：

根据操作指令的接口和数据结构定义，对操作指令对应的用户指令进行封装，得到封装后的用户指令；

对封装后的用户指令进行编译，得到编译结果；

将编译结果以动态链接或静态链接的方式，插入至静态操作池内，得到定制操作指令对应的二进制代码。

上述公开实施例中的静态操作池为深度学习编程库中的存储区域，其具体的实现方式在后续公开实施例中进行具体阐述。

通过对封装的用户指令进行编译得到编译结果，并将编译结果以动态链接或静态链接的方式插入到深度学习编程库的静态操作池内，得到定制操作指令对应的二进制代码，通过这一过程，可以生成预先编译好的定制操作指令，并保存该定制操作指令的编译结果，可以将多次重复出现的非库内自带操作指令转为一个封装好的定制操作指令，从而在实现某一深度学习算法时，可以直接通过调用定制操作指令实现需要执行的操作，而避免多次重复无用的指令编辑，而且由于该定制操作指令的编译结果已保存在深度学习编程库内，因此编译时可以直接调用与该定制操作指令对应的二进制代码，而无需多次重复编译，有效提升了编译的效率，缩短编译时间。

在一个示例中，根据上述公开实施例中提出的定制操作指令对应的二进制代码的生成过程，实现的操作定制的具体过程可以为：用编程语言实现定制变换，得到待插入代码（insert code）；按操作指令的接口和数据结构定义对待插入代码进行封装，并完成数据格式转换等工作；编译待插入代码，以动态链接或者静态链接的方式插入到深度学习编程库中，完成操作定制；如操作指令一样正常使用定制操作指令。 需要注意的是，定制操作指令的名称由用户指定，但不能与build-in操作的操作名称冲突。

在一种可能的实现方式中，本公开实施例提出的操作指令还可以支持操作融合功能，操作融合

（operator fusion）是指将多个可塑操作按调用顺序组合成一个新的可塑操作，该新操作可以被称为融合操作指令（fusion operator）。融合操作指令仍然满足操作指令的定义、性质，且支持操作指令的所有功能（支持特例化、融合等）。在一个示例中，操作融合形式化表示如下：

$$op_{fused} = Fuse\left(op_1, op_2, \ldots, op_n\right)$$

本公开实施例中，操作融合满足弱变换等价性：融合操作指令的计算结果和原始操作指令序列的计算结果在误差允许的范围内可以看做相等，形式化表示为 error < epsilon，epsilon由应用过程对精度的敏感程度来决定。在一个示例中，融合操作指令可以再次参与操作融合，称为高阶融合（high-order fusion），高阶融合得到的输出仍然是可塑操作，表示如下：

$$op_{fused2} = Fuse\left(op_1, op_2, \ldots, op_{fused}, \ldots, op_n\right)$$

操作融合可以带来两个好处：优化性能和简化编程。在优化性能上，编程库可以在融合操作内部进行计算图级别的编译优化（比如可以通过线性变换、常量折叠等优化技术来减少整体的运算量和访存量），从而减少操作在设备上的执行时间；在简化编程上，可以用单个融合操作指令来表示深度学习算法中常用的功能块（比如ResNet中的残差块）甚至整个计算模型，这些高度抽象的组件可以被反复使用，从而提高程序开发效率。

在一种可能的实现方式中，操作融合需要满足一定的条件，在一个示例中，这一条件可以为：待融合的操作指令可以表示成单向操作指令序列中的连续子序列。如图7示出的计算模型示意图，上述公开实施例中已经提出，在一个示例中，该计算模型可以表示为以下两种单向操作指令序列，分别为seq1：（conv，pool，bn，relu，add）和seq2：（conv，pool，relu，bn，add）。这两个操作指令序列中的任意子序列都可以进行操作融合，图9示出根据本公开一实施例的操作融合结果示意图，如图所示，在一个示例中，可以融合seq1中的conv，pool，bn这三个操作指令，此时可以得到计算模型（fusion，relu，add）；图10示出根据本公开一实施例的操作融合结果示意图，如图所示，在一个示例中，可以融合seq2中的bn，add这两个操作指令，此时可以得到计算模型（conv，pool，relu，fusion）。在一个示例中，无法融合pool，relu，add这三个操作指令，因为它们既不是seq1也不是seq2的连续子序列，如果将其强行融合成操作fusion，那么无论fusion插入序列（conv，bn）中的哪个位置，都会存在循环数据依赖，即bn依赖于fusion的输出结果，fusion也依赖于bn的输出结果，因此此时无法进行操作融合。

根据上述操作融合的原理，在一种可能的实现方式中，融合操作指令的创建过程可以包括：

创建融合操作指令的名称。

根据待融合的操作指令确定融合操作子指令。

根据待融合的操作指令的调用顺序，确定融合操作子指令之间的操作连接关系。

根据操作连接关系，连接融合操作子指令，得到连接结果。

根据融合操作指令对应的用户指令，设置融合操作指令的输入参数和输出参数。

将名称、连接结果、输入参数和输出参数进行打包，得到融合操作指令。

通过上述融合操作指令的创建过程，可以较为便捷的将多个操作指令融合为一个融合操作指令，从而有效优化编译性能，减少操作指令在设备上的执行时间，同时也可以通过单个融合操作指令表示深度学习算法中常用的功能块甚至整个计算模型，这些抽象的指令可以被反复使用，从而提高程序开发效率。

操作融合的具体编程接口可以根据实际情况灵活设定，图11示出根据本公开一实施例的操作融合的相关编程接口的示意图，在一个示例中，基于图中示出的编程接口，结合上述融合操作指令的创建过程，可以得到创建一个融合操作的过程，图12示出根据本公开一实施例的创建融合操作的过程示意图，如图所示，在一个示例中，创建一个融合操作的步骤可以包括：

创建融合操作，指定融合操作的操作类型名（不得与build-in操作类型冲突）。

调用nclAddFusionOperator/nclSetFusionOperators接口指定所有待融合的子操作。

调用nclLinkOperator接口指定子操作之间的操作连接关系。

调用nclAddFusionInput以及nclAddFusionOutput接口设置融合操作的输入和输出参数。

调用nclFuseOperator接口完成操作融合。

上述步骤中对子操作进行函数连接的目的是构建计算图，nclFuseOperator接口会对计算图进行及时编译优化，从而加速操作执行。

通过上述公开实施例可以看出，本公开实施例提出的操作指令，可以包括操作融合指令，也可以包括如build-in操作指令、特例化操作指令等其他类型的操作指令，针对不同类型的操作指令，实现的编程模型同样具有区别。在本公开实施例中，NCLAPI采用的是混合编程模型，即既支持命令式编程，也支持声明式编程。在一种可能的实现方式中，可以基于操作融合来设计混合式编程模型，即不使用融合操作的编程模型是命令式编程模型，使用融合操作的编程模型是声明式编程模型，两种编程模型可以混合使用。

编程模型的实现方式可以根据实际情况进行灵活设定，在一种可能的实现方式中，可以基于三个因素设计编程模型，分别为：数据流、执行流和控制流。 在数据流上，为了完成主机和设备之间的数据搬运，本公开实施例为NCLAPI设计了数据拷贝接口nclMemcpy。在控制流上，为了控制设备执行以及进行主机和设备间同步，本公开实施例为NCLAPI设计了操作调用接口nclInvokeOperator以及设备同步接口 nclSyncDevice。在执行流上，本公开实施例对计算模型的执行方式进行了三类划分，分别为：逐层调用：逐个调用计算模型中的所有操作；融合调用：对整个计算模型进行操作融合，然后调用融合操作；分段融合调用：对计算模型进行分段操作融合，然后分段调用融合操作。并以这三类执行方式来区分NCLAPI编程模型：逐层调用的执行方式对应命令式编程模型；融合调用的执行方式对应声明式编程模型；分段融合调用的执行方式对应混合式编程。上述公开实施例已经提出，在一种可能的实现方式中，操作数据根据深度学习编程库接口接收的用户指令创建或调用，在一个示例中，基于上述公开实施例的三种计算模型的执行方式，可知根据用户指令触发调用深度学习编程库接口中的操作指令，可以包括：根据用户指令，逐个调用所有对应的操作指令；或者，根据用户指令，对所有对应的操作指令进行融合，得到融合操作指令，调用融合操作指令；或者，根据用户指令，对所有对应的操作指令进行分段，得到分段结果，对每一个分段结果分别进行融合，得到对应的分段融合操作指令，依次调用分段融合操作指令。

在一种可能的实现方式中，这三种调用方式可以用来区分NCLAPI编程模型，逐层调用的执行方式可以对应命令式编程模型；融合调用的执行方式可以对应声明式编程模型；分段融合调用的执行方式可以对应混合式编程。

图13示出根据本公开一实施例的三层计算模型的数据流示意图，如图所示，在一个示例中，基于图中的编程模型，通过不同的调用方式调用操作指令的过程可以为：逐层调用：调用三次nclInvokeOperator接口分别执行conv、pool、fc操作；融合调用：先将conv、pool、fc三个操作融合成单个操作，然后调用一次nclInvokeOperator接口执行融合操作；分段融合调用：融合conv、pool两个操作，然后调用两次nclInvokeOperator接口分别执行融合操作和fc操作。

图14示出根据本公开一实施例的混合式编程模型的实现框图，如图所示，在一个示例中，混合式编程模型通过命令式编程模型和声明式编程模型共同构成。在一个示例中，命令式编程模型的完整编程过程可以为：初始化：初始化设备和运行环境；创建操作：创建单个操作，选择性地进行参数绑定、操作特例化；调用操作：准备操作参数（包括创建Tensor和分配设备地址），拷贝主机侧输入数据到设备内存，进行操作调用，设备同步，从设备内存读取输出结果；释放操作资源：销毁前两步中不再使用的资源，包括张量、操作、内存等；重复创建、调用及释放操作直到计算模型中的所有操作执行完成；退出：关闭设备，销毁运行环境。在一个示例中，声明式编程模型的完整编程过程可以为：初始化：初始化设备和运行环境；创建子操作：创建所有需要参与融合的子操作，选择性地对子操作进行参数绑定；创建融合操作：创建融合操作，添加待融合的子操作，指定子操作之间的操作连接关系，设置融合操作的输入、输出参数，进行操作融合，选择性地进行操作特例化；调用融合操作：准备操作参数（包括创建Tensor和分配设备地址），拷贝主机侧输入数据到设备内存，进行操作调用，设备同步，从设备内存读取输出结果；释放操作资源：释放子操作资源，释放融合操作资源；退出：关闭设

备，销毁运行环境。

上述公开实施例中已经提出，特例化操作指令和融合操作指令可以优化编译性能，然而在一种可能的实现方式中，这两种操作指令的编译均需通过即时编译进行实现，这可能导致主机侧运行时间的增加，程序的总运行时间变长。因此，在一种可能的实现方式中，可以通过离线模式，进一步优化节省编译的时间，提升编译效率。

由于深度学习算法具有高度的可重用性，优化过的计算模型可以反复用于推理。因此对于同一个计算模型来说，特例化操作指令和融合操作指令通常只需要进行一次创建，而可以反复进行操作指令调用，操作调用的次数越多，特例化操作和操作融合的收益越高。因此，在一种可能的实现方式中，本公开实施例为NCLAPI提出了一种离线模式（offline mode），以消除操作特例化和操作融合在主机侧的二次编译开销。离线模式的实现方式可以根据实际情况进行灵活设定，在一个示例中，可以在一个单独的程序中使用操作特例化或操作融合提前优化好操作指令，在另一个程序中直接使用。在一种可能的实现方式中，提前优化好的操作指令可以被称为离线操作（offline operator）。与离线模式对应的是在线模式（online mode），图15示出根据本公开一实施例的离线模式和在线模式的区别示意图，如图所示，在一个示例中，在线模式可以为同一个程序中先进行操作特例化或操作融合，然后再调用特例化操作指令或融合操作指令。

通过上述公开实施例可以看出，在一种可能的实现方式中，离线操作指令及其编译得到的二进制代码，需要提前进行保存，从而便于后续的直接使用，因此，在本公开实施例中，还为NCLAPI设计了离线缓存（offline cache）。离线缓存的实现方式可以根据实际情况灵活决定，在一种可能的实现方式中，离线缓存包括离线文件和索引表；其中，离线文件，用于保存离线操作指令的预先编译结果；索引表，用于指示离线操作指令的预先编译结果在所述离线文件中的位置。离线文件和索引表的具体实现方式也可以根据实际情况灵活选择，在一个示例中，离线操作被保存到离线文件中，并通过索引表来索引在离线文件中的位置。索引表采用键值对（Key，Value）实现，Key是离线操作的名称，Value是指针，用于指向离线文件中该离线操作对应的二进制代码。离线操作指令的具体接口可以根据实际情况进行设置，图16示出根据本公开一实施例的离线接口的示意图，如图所示，在一个示例中，离线操作指令的接口实现方式可以为：nclSaveOperator接口将指定操作指令保存到离线缓存中，并以op_type 指定的字符串作为该操作的名称和索引 Key。离线操作指令的使用方法和build-in操作指令完全一样，只是操作类型不同，因此用户仍然可以使用nclCreateOperator接口创建离线操作指令，NCLAPI可以根据给定的操作名称优先匹配build-in操作指令，如果没匹配上，再到离线缓存中查找是否存在对应的离线操作指令。

通过上述各公开实施例可以看出，NCLAPI可以传递操作数据，且对于深度学习算法具有较好的适应性，因此，在一种可能的实现方式中，NCLAPI可以集成于深度学习框架。在一种可能的实现方式中，可以对深度学习框架进行类扩展，将张量数据与操作指令封装于深度学习框架的数据内，实现深度学习框架与深度学习编程库接口的集成。在实际应用中，随着深度学习框架的具体实现方式不同，将NCLAPI集成于深度学习框架的实现方式也可能随之发生变化。

在一个示例中，深度学习框架可以为Caffe，由于Caffe包含三个关键数据结构：Blob、Layer、Net。Blob主要用于存储数据、完成主机和设备间的数据拷贝、提供数据访问接口。Layer用来表示操作（如卷积、池化等操作），它以Blob为输入和输出。Caffe为Layer设计了继承体系，不同的操作可以通过编写Layer的子类来实现。Layer有三个关键方法：Setup、Forward和Backward，它们分别负责操作初始化、前向计算以及反向计算。为了支持不同的设备，同一个Layer子类可以包含多个 Forward、Backward方法。

Net保存了所有的Blob和Layer，它用Layer构成的有向无环图来表达完整的计算模型。Net有三个关键方法：Init、Forward和Backward。Init方法把NetParameter（由prototxt转换而得）定义的计算模型转换成Blob和Layer，并调用Setup方法对所有的Layer进行初始化。Forward方法对整个计算模型进行前向推理，Backward方法对计算模型进行反向训练。Caffe使用prototxt对深度学习计算模型建模，用户按照prototxt的语法描述处理层、数据以及处理层连接关系，Caffe接收prototxt文件，将其转换成Blob、

Layer、Net后执行。

根据Caffe的组成，在一个示例中，本公开实施例采用以下方式将NCLAPI集成到Caffe中：

拓展Blob类，可以是将nclTensor数据结构以及相关接口（比如nclCreateTensor、nclSetTensorAttr、nclMemcpy）封装到Blob中；

拓展Layer类，可以是将NCLAPI的nclOperator数据结构以及相关接口封装到Layer中，具体来说，将nclCreateOperator、nclSpecializeOperator、nclBlindOutputTensor等接口封装到Layer的Setup方法中，将nclInvokeOperator封装到Layer的Forward和Backward方法中。为了在不破坏已有Layer的前提下支持新的设备和算子，本公开实施例采用为Layer新增子类的实现方法；

拓展Net类，可以是将NCLAPI的操作融合接口封装到Net中，由于在Net中能够拿到所有的Layer，因此Net最适合作为操作融合的载体，本公开实施例为Net新增了操作融合模块，可以对计算模型进行分段融合或者完全融合。

在一个示例中，深度学习框架可以为TensorFlow，图17示出根据本公开一实施例的TensorFlow的架构图，如图所示，在一个示例中，TensorFlow在设计时很好地考虑了架构的可扩展性，它预留了算子添加、设备注册机制，并给出了详细的官方说明文档，因此TensorFlow本身就比较容易集成第三方深度学习编程库和深度学习处理器。由于TensorFlow的分布式主控（Distributed master）负责计算子图的划分、任务分配，因此在一个示例中，可以将NCLAPI的操作融合集成到分布式主控模块，对子图进行操作融合。

在一个示例中，本公开实施例采用以下方式将NCLAPI集成到TensorFlow中：

拓展Tensor类，将nclTensor数据结构以及相关接口（比如nclCreateTensor、 nclSetTensorAttr、nclMemcpy）封装到Tensor中；

按照TensorFlow官方文档注册新的设备（深度学习处理器）和NCLAPI算子；

将操作融合功能集成到分布式主控（Distributed master）模块。

通过上述各公开实施例可以看出，NCLAPI使用张量来表示标量、向量、矩阵等多维数据，使用操作指令来表示深度学习处理层。操作指令支持操作融合、操作定制、操作特例化、操作参数运行时可变、离线优化以及混合式编程模型（命令式+声明式），从而很好的解决性能优化和编程灵活性问题。在一个示例中，NCLAPI的一个具体实现可以部署到DaDianNao深度学习处理器平台上，同时它还被集成到了Caffe、TensorFlow等主流的深度学习框架中。实践证明，NCLAPI能够运行包括图像分类、目标检测、自然语言处理在内的主流深度学习算法，它具有很强的通用性和灵活性。此外，NCLAPI在数据结构和接口的设计上模拟了神经演算，从而证明：神经演算可以作为指导深度学习编程库设计的理论基础。

图18示出根据本公开一实施例的NCLAPI与主流深度学习编程库接口的对比示意图，从图中可以看出，NCLAPI与主流的深度学习编程库接口相比，可以支持混合式编程模型，能同时满足性能优化和编程灵活性需求；可以支持操作定制，具备很强的操作扩展性，能更好地支持端到端执行性能优化；可以支持操作融合、操作特例化以及离线模式，能够从多个角度优化程序的性能。同时，集成NCLAPI的Tensorflow能够端到端地支持（不使用任何CPU操作）大量的深度学习计算模型。

根据上述各公开实施例可以看出，本公开实施例提出的深度学习算法的编译方法，在一种可能的实现方式中，可以基于NCLAPI接口进行实现，在接收了NCLAPI传递的操作数据后，可以根据操作数据中包括的操作指令，判断指令类型，根据判断结果执行与指令类型相应的编译操作，得到深度学习算法的二进制代码。因此，基于这一编译方法，本公开实施例还提出了一种与该编译方法适应的深度学习编程库架构-神经演算编程库架构（neurological calculus library architecture，NCLA）。图19示出根据本公开一实施例的NCLA的整体架构示意图，如图所示，在一种可能的实现方式中，NCLA可以由即时编译系统（just-in-time compilation system，NCLCS）、静态操作池（static operator pool，NCLSOPP）以及运行时系统（runtie system，NCLRT）组成，而且NCLA可以与NCLAPI进行集成，根据NCLAPI传递的操作数据进行编译。即时编译系统可以在运行时对任意操作指令进行运算和数据协同编译优化，生成高效的二进制代码；静态操作池可以用于保存已经优化好的二进制代码，从而消除二次编译开销；

运行时系统可以提供设备管理、内存管理、操作执行、设备同步等基本功能，使得操作指令能够端到端地部署到深度学习处理器上执行。

上述公开实施例中已经提出，深度学习领域的程序优化对算法和硬件变动极其敏感，因此在实现过程中，很难通过提前编译优化（AOT）来满足不同算法、不同硬件平台下的性能需求。因此，在一种可能的实现方式中，本公开实施例可以采用即时编译优化（JIT）来设计NCLA。即时编译优化可以在运行时针对不同的算法、不同的硬件平台动态地调整优化策略，实现普适的性能优化。然而，即时编译会引入额外的运行时开销，缓解该问题的常用手段是引入即时编译缓存，深度学习算法具有高度的可重用性，因此缓存可以发挥巨大的作用，在一种可能的实现方式中，本公开实施例使用静态操作池作为即时编译系统的缓存。由于运算和数据间耦合度极高，单独对运算或者数据进行优化都无法充分发挥深度学习处理器的性能，因此在一种可能的实现方式中，可以采用一种运算和数据协同优化的编译框架来设计即时编译。

基于上述原理，可以将NCLAPI传递的操作数据中包含的操作指令进行分流，具体的分流方式可以根据实际情况灵活选择，在一种可能的实现方式中，可以将操作指令划分为静态操作指令和动态操作指令，动态操作指令可以触发NCLA进行即时编译，而静态操作指令可以触发NCLA执行查找操作。静态操作指令和动态操作指令具体包含哪些指令，可以根据深度编程接口传递的操作指令的实际情况进行确定，不局限于下述各公开实施例。

在一种可能的实现方式中，静态操作指令可以包括定制操作指令、build-in操作指令和离线操作指令中的一个或多个；其中，

定制操作指令包括，根据操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；

build-in操作指令包括，深度学习编程库接口中包括的自有操作指令；

离线操作指令包括，预先编译完成的动态操作指令，其中，预先编译的结果保存在离线缓存中。

定制操作指令、build-in操作指令以及离线操作指令的具体实现方式已在上述各公开实施例中进行阐述，在此不再赘述。

在一种可能的实现方式中，动态操作指令可以包括特例化操作指令和/或融合操作指令；其中，

特例化操作指令包括，对操作指令绑定输入参数后转换得到的操作指令；

融合操作指令包括，对多个操作指令根据调用顺序进行组合得到的操作指令。

特例化操作指令和融合操作指令的具体实现过程在上述各公开实施例中也已经阐述过，在此不再赘述。

因此，在一种可能的实现方式中，步骤S13可以包括：

步骤S131，判断操作指令的指令类型。

步骤S132，在指令类型为静态操作指令时，根据静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为深度学习算法的二进制代码。

基于上述公开实施例中提出的原理，可以看出，通过在指令类型为静态操作指令时，根据静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为深度学习算法的二进制代码，可以对具有重用性的操作指令，直接查找对应的二进制代码，避免多次重复编译，消除二次编译开销，提升编译效率。

进一步地，在一种可能的实现方式中，步骤S132可以包括：

根据静态操作指令的名称，在静态操作池中查找与名称对应的二进制代码。

在查找结果为成功时，返回二进制代码，作为深度学习算法的二进制代码。

在一种可能的实现方式中，步骤S132还可以包括：

在查找结果为失败时，将静态操作指令作为动态操作指令，进行实时编译。

上述各公开实施例已经提出，在一种可能的实现方式中，定制操作指令可以由用户使用深度学习领域编程语言编写，先对它进行提前编译生成二进制代码，然后以动态链接或者静态链接的方式插入到静态操作池中。离线操作指令的二进制代码可以由即时编译系统生成，用户通过调用nclSaveOperator接口将其插入到静态操作池中。build-in操作指令是NCLAPI自带的操作指令，考虑到深度学习领域的

程序优化对算法和硬件极其敏感，为了降低开发成本，本公开实施例并没有采用手工优化的方式来实现build-in操作指令，而是提前用即时编译系统对操作指令进行伪特例化（没有绑定任何输入参数的特例化）来生成build-in操作指令对应的二进制代码，然后插入到静态操作池中。

通过上述各公开实施例可以看出，静态操作池可以用来储存与静态操作指令对应的二进制代码，其具体的实现方式可以根据实际情况灵活设定，不局限于下述公开实施例。在一种可能的实现方式中，静态操作池可以包括静态操作源文件，静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，静态代码段用于保存build-in操作指令对应的二进制代码；动态代码段用于保存定制操作指令对应的二进制代码；静态数据段用于保存与build-in操作指令对应的张量数据；动态数据段用于保存与定制操作指令对应的张量数据。

上述公开实施例中还提出过，离线操作指令对应的二进制代码可以保持在离线缓存中，因此，在一种可能的实现方式中，静态操作池还可以包括离线缓存。

基于上述公开实施例，图20示出根据本公开一实施例的静态操作池的实现方式示意图，如图所示，在一个示例中，可以为深度学习编程库的源文件（.so）新增4种段：静态代码段、静态数据段、动态代码段、动态数据段。静态代码段和静态数据段用于存储build-in操作指令对应的二进制代码以及对应的常量数据，动态代码段、动态数据段用于存储定制操作指令对应的二进制代码和对应的常量数据。区分动态段和静态段的原因如下：定制操作指令由用户编写，它会不断扩展，因此本公开实施例为其设计了大小可变的动态段；而build-in操作指令由深度学习编程库自带，它不会发生变化，因此本公开实施例为其设计了大小不变的静态段。此外，本公开实施例并没有把离线操作指令对应的二进制代码嵌入到深度学习编程库的源文件（.so）中，因为离线操作指令通常是重量级的计算模型（AlexNet、ResNet）等，它占用的存储空间巨大，因此本公开实施例为离线操作单独设计 了离线缓存（offline cache），用文件系统来保存离线操作指令。离线缓存由索引表（index table）和离线文件组成，索引表采用键值对（Key，Value）实现，Key是离线操作指令的类型名，Value是离线操作指令对应的二进制代码引用指针。

基于上述公开实施例提出的静态操作池的实现方式，在一种可能的实现方式中，根据静态操作指令的名称，在静态操作池中查找与名称对应的二进制代码，可以包括：

根据静态操作指令被创建时指定的名称，在静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与名称对应的二进制代码。

在一个示例中，用户可以在调用操作创建接口（ nclCreateOperator）时指定操作的名称，NCLA以操作的名称为索引到静态操作池中查找对应的二进制代码，查找的顺序依次为定制操作指令、build-in操作指令、离线操作指令， 如果查找命中，则返回对应的二进制代码引用，否则触发即时编译。

上述各公开实施例阐明了在操作指令为静态操作指令时编译的具体方式，通过上述各公开实施例还可得知，操作指令还可以为动态操作指令，可以触发NCLA执行即时编译，因此，在一种可能的实现方式中，步骤S13还可以包括步骤S133：在指令类型为动态操作指令时，对动态操作指令进行实时编译，得到实时编译结果，作为深度学习算法的二进制代码。

通过上述过程可以看出，如果NCLAPI接口传递的为融合操作指令或者特例化操作指令这些动态操作指令，在调用操作融合或操作特例化接口（nclFuseOperator、nclSpecializeOperator）时会触发即时编译（也可称为实时编译）。即时编译系统为操作生成高度优化的二进制代码，在操作调用时交给运行时系统执行。在一种可能的实现方式中，用户还可以调用nclSaveOperator接口保存优化过的操作指令（比如融合操作指令），则该操作指令对应的二进制代码可以被保存到离线缓存中，并以其操作名作为查找的索引，否则，为了确保编程库的体积不会迅速膨胀，程序退出后未保存的操作指令对应的二进制代码会被丢弃。

对动态操作指令进行实时编译的具体过程不受限定，可以根据实际情况灵活选择，在本公开实施例中，涉及了一种运算和数据协同优化的深度学习编译框架（computation and data unified compilation architecture，CDUCA）来实现对动态操作指令的实时编译，图21示出根据本公开一实施例的CDUCA

的架构示意图，如图所示，在一种可能的实现方式中，CDUCA包含计算图引擎、代码生成器、数据优化器三个组件，它在多个层次对运算和数据进行协同优化，从而生成高效的二进制代码。三个组件的具体方式不唯一，在一种可能的实现方式中，三个组件可以实现的功能分别为：

计算图引擎：采用线性变换、常量折叠等优化技术，对原始的计算图和常量数据进行面向算法的高级优化，生成优化后的计算图和常量数据；

代码生成器：采用基于代价模型（cost model）的启发式搜索策略，对计算图和常量数据进行运算和数据协同编译优化，生成高效的目标平台代码以及数据描述符；

数据优化器：解析数据描述符，面向目标平台对常量数据进行拆分、重排序、精度转换等优化，然后将优化后的常量数据和目标平台代码进行打包（地址重定位等），生成最终二进制代码。

基于上述公开实施例的架构，在一种可能的实现方式中，步骤S133可以包括：

步骤S1331，根据动态操作指令，得到与动态操作指令对应的原始计算图和原始模型数据。

步骤S1332，根据原始计算图和原始模型数据，进行面向深度学习算法的协同处理，得到第一计算图和第一模型数据。

步骤S1333，根据第一计算图，生成硬件指令和数据描述符。

步骤S1334，根据数据描述符，对第一模型数据进行面向硬件平台的处理，得到第二模型数据。

步骤S1335，根据所述硬件指令和所述第二模型数据，得到深度学习算法的二进制代码。

通过上述公开实施例可以看出，CDUCA的输入是计算图和常量数据，因此在一种可能的实现方式中，需要通过步骤S1331，得到与动态指令对应的原始计算图和原始模型数据，步骤S1331的实现方式不受限定，在一种可能的实现方式中，步骤S1331可以包括：

通过解析动态操作指令，得到与动态操作指令对应的原始计算图。

根据动态操作指令的参数，得到原始模型数据。

基于上述公开实施例，在一个示例中，获取与动态指令对应的原始计算图和原始模型数据的方式可以为：根据动态操作指令的参数，直接得到原始模型数据，而原始计算图可以通过NCLA解析动态操作指令生成，具体的解析过程可以根据实际情况灵活决定，在此不受限定，图22示出根据本公开一实施例的原始计算图的形式，如图所示，在一个示例中，原始计算图中包含Tensor和Operator两种图节点，分别对应操作指令的输入输出数据和数据变换。

通过上述公开实施例可以看出，步骤S1332可以对应CDUCA中的计算图引擎部件，计算图引擎的实现方式不受限定，可以根据实际情况灵活决定。

计算图中的节点可以用于表示深度学习过程中执行的操作，深度学习算法中常见的操作可以有卷积、全连接、激活、池化、批规范化、缩放等。这些操作根据其具体的实现形式，可以被分为线性变换操作和非线性变化操作这两类，其中，所有的线性变换操作均可以表示成向量或矩阵进行乘加的形式，因此，线性变换操作的通用表达形式可以为：

$$Y = X * W + B$$

其中，X 和 Y 为变量，W 和 B 为模型数据常量。

任何线性变换操作均可以表示成上述的通用表达形式，因此，不可以被表达为上述通用表达形式的操作，为非线性变换操作。在一个示例中，上述列举的深度学习算法的常见操作内，卷积、全连接、批规范化、缩放操作是线性变换操作，池化、激活是非线性变换操作。在一个示例中，将全连接操作通过上述通用表达形式进行表达时，X 和 Y 分别表示全连接的输入神经元矩阵和输出神经元矩阵，W 表示权值矩阵，B 表示偏置矩阵。其他的线性变换操作通过通用表达形式的具体表达方式，在此不再赘述。

对于线性变换操作来说，如果存在两个连续的线性变换操作，分别为：

$$Y_1 = X_1 * W_1 + B_1$$

$$Y_2 = Y_1 * W_2 + B_2$$

由于线性变换操作满足分配律和结合律，且线性变换操作的通用表达形式内 W 和 B 均为常量，因此，两个线性变换操作分别可以做如下的等价线性变换：

$$Y_2=\left(X_1*W_1+B_1\right)*W_2+B_2$$

$$Y_2 = X_1*W_1*W_2 + B_1*W_2+B_2$$

$$W^{'}=W_1*W_2, B^{'} = B_1*W_2+B_2$$

$$Y_2 = X_1*W^{'} + B^{'}$$

通过上述等价线性变换，原始的线性变换操作，可以通过线性变换优化手段，以及常量折叠优化手段，进行整体优化，最终简化为一步线性变换操作。通过上述的线性变换优化手段和常量折叠优化手段进行优化，可以在减少运算量的同时，对模型数据进行了压缩，一方面可以减少模型数据的存储开销，另一方面可以减少运行时的访存量。

因此，在一种可能的实现方式中，面向深度学习算法的协同处理的具体实现形式，可以为线性变换和常量折叠。

图23示出根据本公开一实施例的计算图引擎的工作流程图，如图所示，在一种可能的实现方式中，步骤S1332可以包括：

步骤S13321，读取原始计算图和原始模型数据。

步骤S13322，识别原始计算图内的连续线性变换操作节点。

步骤S13323，通过线性变换和常量折叠，对连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

通过上述内容可知，2个连续的线性变换操作可以通过线性变换和常量折叠的方式，简化为1个线性变换操作，当存在3个连续的线性变换操作时，可以将前2个连续的线性变换操作通过线性变换和常量折叠的方式简化为1个线性变换操作，再将这个简化后的线性变换操作，与剩余的另一个线性变换操作，再次通过线性变换和常量折叠的方式，最终简化为1个线性变换操作，由此类推，当存在更多数量的连续的线性变换操作时，通过线性变换和常量折叠的方式，可以将这些线性变换操作进行合并，简化为至少1个线性变换操作。

由于每个线性变换操作在计算图中均对应一个相应的线性变换操作节点，因此，在一种可能的实现方式中，连续线性变换操作节点可以包括：至少2个连续的线性变换操作节点。这样连续线性变换操作节点可以对应至少2个连续的线性变换操作，连续线性变换操作节点的数量不受限定，可以根据计算图的实际情况进行确定。

在一种可能的实现方式中，步骤S13323的具体过程可以包括：

将原始计算图内的连续线性变换操作节点进行线性变换和常量折叠，将连续线性变换操作节点合并，得到第一计算图。

将连续线性变换操作节点对应的模型数据合并，得到第一模型数据。

在一个示例中，原始计算图内的连续线性变换操作节点可以为依次连接的节点1、节点2、节点3和节点4，对应的模型数据组合可以为模型数据组1，通过线性变换和常量折叠，可以将节点1、节点2、节点3和节点4合并为1个，最终得到节点5，在这一过程中，模型数据组1由于发生了常量折叠，因此内部包含的模型数据可能发生了合并，最终得到了一份合并后的模型数据组，可以称为模型数据组2；在一个示例中，原始计算图内的连续线性变换操作节点可以为依次连接的节点1、节点2、节点3和节点4，对应的模型数据组合可以为模型数据组1，通过线性变换和常量折叠，可以将节点1、节点2和节点3合并为1个节点6，而不将节点6和节点4进行合并，此时可以最终得到节点4和节点6，在这一过程中，由于节点1、节点2和节点3发生了常量折叠，因此模型数据组1内与之对应的模型数据组合可能发生了合并，最终得到了一份合并后的模型数据组，可以称为模型数据组3，而原有的节点4对应的模型

数据由于未发生常量折叠，因此将模型数据组3和节点4对应的模型数据进行组合，则可以得到与当前整体线性变换操作对应的模型数据组4。由上述两个示例类推，在原始计算图内的连续线性变换操作节点的数量发生改变时，步骤S13323的具体过程也可以随之有相应的变化，在此不再一一列举。在一个示例中，深度学习算法对应的神经网络可以是经典图像分类网络Resnet，图24示出根据本公开一实施例的图像分类网络内包含子结构的结构示意图，从图中可以看出，在Resnet这一网络中，可以包含有卷积+批规范化+缩放的子结构，由于卷积、批规范化和缩放均为线性变换操作，因此在Resnet这一网络对应的计算图中，可以将这一子结构所对应的3个操作节点，通过如上述内容中线性变换操作的方式，进行线性变换和常量折叠，最终合并为1个操作节点，并将这3个操作节点中涉及到的模型数据也随之合并。

通过上述任一形式的过程，可以对深度学习算法的运算过程进行优化，得到面向深度学习算法进行协同处理后的第一计算图和第一模型数据，从而可以减小运行这一深度学习算法时的访存量，也可以减少模型数据在存储时的存储开销。

基于第一计算图和第一模型数据，可以通过步骤S1333，生成硬件指令和数据描述符。步骤S1333的具体实现形式不受限定，任何可以基于计算图来生成硬件指令的过程均可以作为步骤S1333的实现形式。

步骤S1333的主要目的在于基于步骤S1332优化后的第一计算图，生成与之相应的硬件平台可读的硬件指令。为了优化硬件平台的访存性能，往往会在硬件平台中靠近计算位置的部分设计片上缓存（on-chip memory），在一个示例中，这一部分可以是靠近深度学习处理器计算单元的位置。在对硬件平台进行访问时，访问片上缓存的速度，往往比访问其他位置的速度快，在一个示例中，其他位置可以是片外双倍数据速率同步动态随机存取存储器（DDR，Double Data Rate Synchronous Dynamic Random Access Memory）。受片上缓存位置和功能的影响，片上缓存的容量有限，因此对片上缓存的利用率，可以直接影响深度学习算法针对相应硬件平台的性能优化效果。然而，通过步骤S1332中面向深度学习算法的协同处理方法，可以实现对于运算的优化，却无法提高片上缓存的利用率。在一种可能的实现方式中，可以通过优化生成硬件指令的过程，来提高片上缓存的利用率，继而提升深度学习算法针对相应硬件平台的性能优化效果。因此，在一种可能的实现方式中，步骤S1333可以包括：根据代价模型对第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

根据代价模型对第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符的具体过程可以根据实际情况灵活选择。图25示出根据本公开一实施例的深度学习算法的编译方法的流程图，如图所示，在一种可能的实现方式中，步骤S1333可以包括：

步骤S13331，通过代价模型对第一计算图进行建模，生成搜索空间和目标函数。

步骤S13332，通过启发式搜索策略，在搜索空间内进行搜索，当目标函数达到阈值时，生成面向硬件平台的硬件指令和数据描述符。

基于计算图生成硬件指令的方式可以存在多种实现形式，如可以将计算图作为输入，通过可以生成硬件指令的模型，并在模型的输出结果中进行寻优，可以得到最终的面向硬件平台的硬件指令。具体应用哪个可以生成硬件指令的模型，并不受限定，在一种可能的实现方式中，可以通过代价模型（cost model），来对第一计算图进行建模。代价模型对操作在深度学习处理器上执行的总时间（包括数据格式转换等开销）进行估计，它考虑的主要因素是操作的访存时间、运算时间以及两者的重叠率，这三个因素直接决定了程序的性能。具体来说，深度学习处理器包含独立运算单元和访存单元，计算指令和访存指令之间可以流水执行夯重叠执行，图26示出根据本公开一实施例的指令流水示意图，如图所示。为了使程序的总运行时间最短，必须减少计算时间、访存时间以及增大两者执行的重叠率。因此，在将第一计算图通过代价模型建模后，生成的硬件指令实际上是多种可能的硬件指令所组合而成的搜索空间，搜索空间内包含的硬件指令，其内容可以指示出硬件平台在操作过程中的多种选择方式：在一个示例中，硬件指令可以指示硬件平台将一块完整的数据分成几次加载，放在片上缓存；在一个示例中，硬件指令可以指示硬件平台将一块完整的数据分成几次加载，并在片上缓存和片外DDR之间根据需求进行换入换出；在一个示例中，硬件指令可以指示硬件平台在进行一次向量运算处理时，处理

的数据量需要多大。由于搜索空间内存在多种硬件指令，具体最终应用哪些应用指令，需要通过对搜索空间内进行搜索，寻找其中最优的指令组合，来作为最终得到的硬件指令。

从上述内容中可知，代价模型可以给出硬件指令的运行估计时间，因此，在将第一计算图通过代价模型后，还可以生成相应的目标函数，目标函数的具体形式在此不做限定，可以根据实际情况进行灵活设定，此目标函数可以表明在搜索空间内进行搜索，最终生成的硬件指令在运行时耗时的多少，因此，在目标函数达到阈值时，表明生成的硬件指令达到了运行时间的要求，继而表明生成的硬件指令可以提升硬件指令在硬件平台中运行时的性能。由于目标函数的具体实现形式不受限定，因此与之相对的目标函数的阈值同样不受限定，可以根据实际情况进行灵活设置。

如何在生成的搜索空间内进行搜索同样也不受限定，在一种可能的实现方式中，可以直接通过暴力搜索的方式来进行搜索，但是这样的搜索过程耗时过长，可能会延长编译过程的耗时，继而可能降低深度学习算法针对相应硬件平台的性能优化效果。在一种可能的实现方式中，可以通过启发式搜索策略来在搜索空间内进行搜索。

启发式搜索策略的目的在于提高在搜索空间内搜索的效率，因此并不限定于某种具体的搜索方式，可以根据实际情况进行灵活选择。在一种可能的实现方式中，启发式搜索策略可以包括：提高硬件平台内片上缓存的利用率的搜索策略；或者，在保证硬件平台内运算单元和访问单元的利用率的基础上，减小运算粒度和访存粒度的搜索策略。在一个示例中，启发式搜索策略的目的可以在于尽量将片上缓存用满，因此此时可以将搜索策略定为提高硬件平台内片上缓存的利用率的搜索策略；在一个示例中，启发式搜索策略的目的可以在于在保证硬件平台中运算单元和访存单元利用率的前提下，尽可能选择运算和访问粒度比较小的指令组合，从而提高计算和访存的覆盖率，因此此时可以将搜索策略定为在保证硬件平台内运算单元和访问单元的利用率的基础上，减小运算粒度和访存粒度的搜索策略；在一个示例中，启发式搜索策略也可以是上述两种策略的均衡策略，即使得上述两种策略达到综合最优情况下的搜索策略。

通过上述任一形式的搜索策略，可以在搜索空间内搜索后生成面向硬件平台的硬件指令，由于搜索空间内包含的硬件指令，其内容可以指示出硬件平台在操作过程中的多种选择方式，因此在搜索空间内进行搜索时，除了可以考虑上述硬件指令的运行性能以外，同样也可以考虑通过生成的硬件指令来指导硬件平台对于模型数据进行优化处理，来进一步提高硬件平台的运算访存速度和访存效率，因此，基于此种考虑，步骤S13332在生成硬件指令的同时，同样可以生成数据描述符，用来指导如何对模型数据进行进一步优化。

基于步骤S1333同时生成的数据描述符，可以对第一模型数据进行进一步优化，来得到第二模型数据，数据描述符具体如何对模型数据进行优化的实现形式不受限定，图27示出根据本公开一实施例的对模型数据进行优化的实现方式图，如图所示，在一种可能的实现方式中，步骤S1334可以包括：根据数据描述符，按照硬件平台内的运算要求，对第一模型数据进行数据拆分；根据数据描述符，按照硬件平台内的运算要求，对第一模型数据进行数据对齐；或者，根据数据描述符，按照硬件平台内的运算要求，对第一模型数据进行维度变换；或者，根据数据描述符，按照硬件平台内的运算要求，对第一模型数据进行精度选择。

上述公开实施例中，在一种可能的实现方式中，对于数据拆分（tiling）来说，可以优化循环、片上存储管理以及实现指令流水；对于数据对齐（align）来说，深度学习处理器支持向量、矩阵运算，因此对数据对齐有一定的要求，对齐访问能够加快访存速度，也能减少DDR bank冲突；对于数据重排布（reorder）来说，卷积等操作的输入数据是多维数组，多维数组的排布顺序会影响访存的跳转次数，对数据进行重排布可以提高访存局部性，降低MMU miss率；对于精度转换（type convert）来说，深度学习处理器通常支持低精度运算，比如半精度浮点、夸比特量化，不同精度运算性能不同，对数据进行合适精度转换能够提高程序的整体性能；精度转换是NCLAPI精度自动选优功能的一种具体实现方式。

数据拆分的具体实现方式不受限定，图28示出根据本公开一实施例的对数据拆分后优化的示意图，如图所示，在一个示例中，可以对一个通道数为2、高和宽为4的常量数据进行优化。假设数据描述符

要求对数据进行如下变换：高和宽同时进行2拆分；数据的排布顺序由HWC调整为CHW；数据运算精度由float32调整为half。数据优化器按照上述描述，从DDR视角对数据实施拆分、重排布、精度转换、对齐等物理变换，最终得到优化后的数据。

在一个示例中，硬件平台在根据硬件指令进行运算的过程中，可能会对数据的对齐有一定要求，如果这一过程在硬件平台内进行操作，则会大大降低硬件平台运行时的性能，因此，可以根据数据描述符，在编译过程中提前对待对齐的数据进行对齐，从而加快硬件平台工作时的访存速度，具体对齐的标准和方式在此不做限定，根据硬件平台的实际要求进行确定即可；在一个示例中，硬件平台在根据硬件指令进行运算的过程中，部分算法，如卷积算法，可能需要将数据解释成多维数组，而多维数组的排布顺序则会影响硬件平台内访存的跳转次数，从而影响硬件平台运行时的性能，因此，可以根据数据描述符，在编译过程中提前按照运算要求对模型数据进行维度变换，来提高访存局部性，从而尽量减少硬件平台的访存跳转，具体进行何种维度变换和变换的方式在此不做限定，根据硬件平台的实际要求进行确定即可；在一个示例中，由于不同的运算精度对应的运算速度不同，硬件平台可能支持的运算精度要求也不同，在一个示例中，硬件平台可能更加支持低精度运算，如果使用高精度运算，可能会降低硬件平台的运行速度，因此，可以根据数据描述符，在编译过程中提前根据硬件平台的精度要求，选择优选的模型数据精度，具体选择何种精度的数据再次不做限定，根据硬件平台的需求确定即可，在一个示例中，优选的精度可以是16比特量化精度，在一个示例中，优选的精度可以是8比特量化精度。

上述将第一模型数据通过处理优化成第二模型数据的过程，可以是上述四种方式的任一种方式，也可以是上述四种方式的任意组合形式，也可以包含更多有利于提升硬件平台速度的其他方式，在此不再列举。通过在编译过程中提前完成对模型数据的优化，可以大大提升硬件平台在运行时的性能，如提高访存速度和访存效率等。

基于上述过程，可以得到深度学习算法的二进制代码，这一二进制代码所包含的内容可以存在多种形式，在一种可能的实现方式中，步骤S1335可以包括：将硬件指令和第二模型数据打包，得到深度学习算法的二进制代码。

在一个示例中，硬件指令可以是根据第一计算图生成的硬件指令，第二模型数据可以是原始模型数据分别经过面向深度学习算法的协同处理，和面向硬件平台的处理所得到的模型数据，即最终得到的深度学习算法的二进制代码，是依次经过步骤S1332、步骤S1333和步骤1334得到的内容所得到的。在一个示例中，硬件指令可以是直接根据原始计算图生成的硬件指令，第二模型数据可以是原始模型数据只经历过面向硬件平台的处理所得到的模型数据，即最终得到的深度学习算法的可执行文件，是只依次经过步骤S1333和步骤S1334所得到的。在一个示例中，硬件指令可以是根据第一计算图生成的硬件指令，第二模型数据可以就是第一模型数据，即最终得到的深度学习算法的可执行文件，是只依次经过步骤S1332和步骤S1333得到的。在一个示例中，硬件指令可以是直接根据原始计算图生成的硬件指令，第二模型数据可以是原始模型数据，即最终得到的深度学习算法的可执行文件，可以是只经过步骤S1333得到的。从上述示例可以看出，步骤S1332、步骤S1333和步骤S1334可以不同时存在，可以根据实际情况进行灵活组合。

通过上述各公开实施例实现的CDUCA，可以集成内存复用、操作融合、延迟隐藏、线性代数变换、公共子表达式消除、常量传播、死代码消除、数据并行等前述编译优化技术。此外，CDUCA层次化的设计结构具有很强的扩展性，开发者能够在CDUCA各个模块集成各种编译优化技术。比如，可以在计算图引擎模块集成操作聚合技术，可以在代码生成器模块集成多面体编译优化技术。通过CDUCA对动态操作指令进行实时编译，可以有效的提升编译效率，从而提升硬件设备的运行速度。

通过上述各公开实施例可以看出，通过即时编译和静态查找两种方式，可以生成深度学习算法的二进制代码，然而通过上述公开实施例还可以看出，在NCLA的整体架构中，还包括运行时系统，因此在一种可能的实现方式中，本公开实施例提出的方法还包括：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

运行时系统的实现方式不受限定，图29示出根据本公开一实施例的运行时系统的模块及功能示意

图，如图所示，运行时系统可以负责主机与深度学习处理器之间的交互，它可以对设备驱动接口进行了封装，为上层提供设备管理、内存管理、操作执行、设备同步等功能。

通过上述各公开实施例，可以实现通过NCLA，实现对深度学习算法的编译，通过实验可以证实，目前NCLA的一个具体实现可以部署到深度学习处理器平台上，并且能够支持包括图像分类、目标检测、自然语言处理在内的主流深度学习算法。本公开实施例使用TensorFlow对几类常用的深度学习应用进行了实验，平均来说，NCLA即时编译系统生成的二进制代码性能可以达到手工优化代码性能的83.24%，在最好情况下，NCLA至少能够发挥72.61%的硬件峰值性能。此外，NCLA的成功案例进一步证实了神经演算以及 上述公开实施例中提到的NCLAPI的通用性。

图30示出根据本公开一实施例的深度学习算法的编译装置的框图，如图所示，该装置20包括：操作数据接收模块21，用于接收深度学习编程库接口传递的操作数据；操作指令获取模块22，用于获取所述操作数据包括的操作指令；编译模块23，用于判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

在一种可能的实现方式中，操作指令根据所述深度学习编程库接口接收的用户指令创建或调用。

在一种可能的实现方式中，编译模块包括：判断单元，用于判断所述操作指令的指令类型；静态查找单元，用于在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码。

在一种可能的实现方式中，静态查找单元用于：根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码；在查找结果为成功时，返回所述二进制代码，作为所述深度学习算法的二进制代码。

在一种可能的实现方式中，静态查找单元还用于：在查找结果为失败时，将所述静态操作指令作为动态操作指令，进行实时编译。

在一种可能的实现方式中，静态操作指令包括定制操作指令、build-in操作指令和离线操作指令中的一个或多个；其中，所述定制操作指令包括，根据所述操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；所述build-in操作指令包括，所述深度学习编程库接口中包括的自有操作指令；所述离线操作指令包括，预先编译完成的动态操作指令，其中，所述预先编译的结果保存在离线缓存中。

在一种可能的实现方式中，定制操作指令对应的二进制代码的生成过程，包括：根据所述操作指令的接口和数据结构定义，对所述操作指令对应的用户指令进行封装，得到封装后的用户指令；对所述封装后的用户指令进行编译，得到编译结果；将所述编译结果以动态链接或静态链接的方式，插入至所述静态操作池内，得到所述定制操作指令对应的二进制代码。

在一种可能的实现方式中，静态操作池包括静态操作源文件，所述静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，所述静态代码段用于保存所述build-in操作指令对应的二进制代码；所述动态代码段用于保存所述定制操作指令对应的二进制代码；所述静态数据段用于保存与所述build-in操作指令对应的张量数据；所述动态数据段用于保存与所述定制操作指令对应的张量数据。

在一种可能的实现方式中，静态操作池还包括离线缓存，所述离线缓存包括离线文件和索引表；其中，所述离线文件，用于保存离线操作指令的预先编译结果；所述索引表，用于指示所述离线操作指令的预先编译结果在所述离线文件中的位置。

在一种可能的实现方式中，静态查找单元进一步用于：根据所述静态操作指令被创建时指定的名称，在所述静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与所述名称对应的二进制代码。

在一种可能的实现方式中，编译模块还包括动态编译单元，用于：在所述指令类型为动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码。

在一种可能的实现方式中，动态编译单元包括：原始数据获取子单元，用于根据所述动态操作指

令，得到与所述动态操作指令对应的原始计算图和原始模型数据；协同处理子单元，用于根据所述原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据；硬件指令生成子单元，用于根据所述第一计算图，生成硬件指令和数据描述符；模型数据处理子单元，用于根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据；二进制代码生成子单元，用于根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码。

在一种可能的实现方式中，原始数据获取子单元用于：通过解析所述动态操作指令，得到与所述动态操作指令对应的原始计算图；根据所述动态操作指令的参数，得到所述原始模型数据。

在一种可能的实现方式中，协同处理子单元用于：读取所述原始计算图和所述原始模型数据；识别所述原始计算图内的连续线性变换操作节点；通过线性变换和常量折叠，对所述连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

在一种可能的实现方式中，硬件指令生成子单元用于：根据代价模型对所述第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

在一种可能的实现方式中，模型数据处理子单元用于：根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行数据对齐；或者，根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行维度变换；或者，根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行精度选择。

在一种可能的实现方式中，二进制代码生成子单元用于：将所述硬件指令和所述第二模型数据打包，得到所述深度学习算法的二进制代码。

在一种可能的实现方式中，动态操作指令包括特例化操作指令和/或融合操作指令；其中，所述特例化操作指令包括，对所述操作指令绑定输入参数后转换得到的操作指令；所述融合操作指令包括，对多个所述操作指令根据调用顺序进行组合得到的操作指令。

在一种可能的实现方式中，特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，所述完全特例化操作指令包括，对所述操作指令绑定所有输入参数后转换得到的操作指令；所述部分特例化操作指令包括，对所述操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于所述操作指令的输入参数数量的正整数；所述伪特例化操作指令包括，对所述操作指令不绑定输入参数后直接转换得到的操作指令。

在一种可能的实现方式中，融合操作指令的创建过程包括：创建所述融合操作指令的名称；根据待融合的操作指令确定融合操作子指令；根据所述待融合的操作指令的调用顺序，确定所述融合操作子指令之间的操作连接关系；根据所述操作连接关系，连接所述融合操作子指令，得到连接结果；根据所述融合操作指令对应的用户指令，设置所述融合操作指令的输入参数和输出参数；将所述名称、连接结果、输入参数和输出参数进行打包，得到所述融合操作指令。

在一种可能的实现方式中，装置还包括执行模块，用于：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

在一种可能的实现方式中，操作数据还包括张量数据，其中，所述张量数据包括形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性。

在一种可能的实现方式中，装置还用于：对深度学习框架进行类扩展，将所述张量数据与所述操作指令封装于所述深度学习框架的数据内，实现所述深度学习框架与所述深度学习编程库接口的集成。

在一种可能的实现方式中，本公开还提出了一种深度学习运算装置，其包含了上述中任意一种可能的深度学习算法的编译装置，深度学习运算装置用于完成设定的深度学习运算。

图31示出根据本公开一实施例的组合处理装置的框图，如图所示，所述组合处理装置，包括上述的深度学习运算装置，通用互联接口，和其他处理装置。

深度学习运算装置与其他处理装置进行交互，共同完成用户指定的操作。其他处理装置，包括中央处理器CPU、图形处理器GPU、神经网络处理器等通用/专用处理器中的一种或以上的处理器类型。其他处理装置所包括的处理器数量不做限制。其他处理装置作为深度学习运算装置与外部数据和控制

的接口，包括数据搬运，完成对本深度学习运算装置的开启、停止等基本控制；其他处理装置也可以和深度学习运算装置协作共同完成运算任务。通用互联接口，用于在所述深度学习运算装置与其他处理装置间传输数据和控制指令。该深度学习运算装置从其他处理装置中获取所需的输入数据，写入深度学习运算装置片上的存储装置；可以从其他处理装置中获取控制指令，写入深度学习运算装置片上的控制缓存；也可以读取深度学习运算装置的存储模块中的数据并传输给其他处理装置。

组合处理装置还可以包括存储装置，存储装置分别与所述深度学习运算装置和所述其他处理装置连接。存储装置用于保存在所述深度学习运算装置和所述其他处理装置的数据，尤其适用于所需要运算的数据在本深度学习运算装置或其他处理装置的内部存储中无法全部保存的数据。

该组合处理装置可以作为手机、机器人、无人机、视频监控设备等设备的SOC片上系统，有效降低控制部分的核心面积，提高处理速度，降低整体功耗。此情况时，该组合处理装置的通用互联接口与设备的某些部件相连接。某些部件譬如摄像头，显示器，鼠标，键盘，网卡，wifi接口。

在一种可能的实现方式中，本公开还提供深度学习芯片，其包括了上述深度学习运算装置或组合处理装置。

在一种可能的实现方式中，本公开还提供芯片封装结构，其包括了上述芯片。

在一种可能的实现方式中，本公开还提供板卡，其包括了上述芯片封装结构。

在一种可能的实现方式中，本公开还提供电子设备，其包括了上述板卡。

电子设备包括数据处理装置、机器人、电脑、打印机、扫描仪、平板电脑、智能终端、手机、行车记录仪、导航仪、传感器、摄像头、服务器、云端服务器、相机、摄像机、投影仪、手表、耳机、移动存储、可穿戴设备、交通工具、家用电器、和/或医疗设备。

所述交通工具包括飞机、轮船和/或车辆；所述家用电器包括电视、空调、微波炉、冰箱、电饭煲、加湿器、洗衣机、电灯、燃气灶、油烟机；所述医疗设备包括核磁共振仪、B超仪、和/或心电图仪。

需要说明的是，对于前述的各方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本公开并不受所描述的动作顺序的限制，因为依据本公开，某些步骤可以采用其他顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于可选实施例，所涉及的动作和模块并不一定是本公开所必须的。

在上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详述的部分，可以参见其他实施例的相关描述。

在本公开所提供的几个实施例中，应该理解到，所揭露的装置，可通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或单元的间接耦合或通信连接，可以是电性或其它的形式。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

另外，在本公开各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件程序模块的形式实现。

所述集成的单元如果以软件程序模块的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储器中。基于这样的理解，本公开的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储器中，包括若干指令用以使得一台计算机设备 (可为个人计算机、服务器或者网络设备等)执行本公开各个实施例所述方法的全部或部分步骤。而前述的存储器包括：U盘、只读存储器(ROM，Read-Only Memory)、随机存取存储器 (RAM，Random Access Memory)、移动硬盘、磁碟或者光盘等各种可以

存储程序代码的介质。

本领域普通技术人员可以理解上述实施例的各种方法中的全部或部分步骤是可以通过程序来指令相关的硬件来完成，该程序可以存储于一计算机可读存储器中，存储器可以包括：闪存盘、只读存储器(英文：Read-Only Memory，简称：ROM)、随机存取器(英文：Random Access Memory，简称：RAM)、磁盘或光盘等。

以上对本公开实施例进行了详细介绍，本文中应用了具体个例对本公开的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本公开的方法及其核心思想；同时，对于本领域的一般技术人员，依据本公开的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本公开的限制。

这里参照根据本公开实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本公开的各个方面。应当理解，流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合，都可以由计算机可读程序指令实现。

附图中的流程图和框图显示了根据本公开的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，所述模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

以上已经描述了本公开的各实施例，上述说明是示例性的，并非穷尽性的，并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实施例的原理、实际应用或对市场中的技术改进，或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

依据以下条款可更好地理解前述内容：

条款A1、一种深度学习算法的编译方法，所述方法包括：

接收深度学习编程库接口传递的操作数据；

获取所述操作数据包括的操作指令；

判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

条款A2、根据条款A1所述的方法，所述操作数据根据所述深度学习编程库接口接收的用户指令创建或调用。

条款A3、根据条款A2所述的方法，所述判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码，包括：

判断所述操作指令的指令类型；

在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码。

条款A4、根据条款A3所述的方法，所述在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码，包括：

根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码；

在查找结果为成功时，返回所述二进制代码，作为所述深度学习算法的二进制代码。

条款A5、根据条款A4所述的方法，所述在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码，还包括：

在查找结果为失败时，将所述静态操作指令作为动态操作指令，进行实时编译。

条款A6、根据条款A3所述的方法，所述静态操作指令包括定制操作指令、build-in操作指令和离

线操作指令中的一个或多个；其中，

所述定制操作指令包括，根据所述操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；

所述build-in操作指令包括，所述深度学习编程库接口中包括的自有操作指令；

所述离线操作指令包括，预先编译完成的动态操作指令，其中，所述预先编译的结果保存在离线缓存中。

条款A7、根据条款A6所述的方法，所述定制操作指令对应的二进制代码的生成过程，包括：

根据所述操作指令的接口和数据结构定义，对所述操作指令对应的用户指令进行封装，得到封装后的用户指令；

对所述封装后的用户指令进行编译，得到编译结果；

将所述编译结果以动态链接或静态链接的方式，插入至所述静态操作池内，得到所述定制操作指令对应的二进制代码。

条款A8、根据条款A6所述的方法，所述静态操作池包括静态操作源文件，所述静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，

所述静态代码段用于保存所述build-in操作指令对应的二进制代码；

所述动态代码段用于保存所述定制操作指令对应的二进制代码；

所述静态数据段用于保存与所述build-in操作指令对应的张量数据；

所述动态数据段用于保存与所述定制操作指令对应的张量数据。

条款A9、根据条款A8所述的方法，所述静态操作池还包括离线缓存，所述离线缓存包括离线文件和索引表；其中，

所述离线文件，用于保存离线操作指令的预先编译结果；

所述索引表，用于指示所述离线操作指令的预先编译结果在所述离线文件中的位置。

条款A10、根据条款A6所述的方法，所述根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码，包括：

根据所述静态操作指令被创建时指定的名称，在所述静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与所述名称对应的二进制代码。

条款A11、根据条款A3所述的方法，所述判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码，还包括：

在所述指令类型为动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码。

条款A12、根据条款A1至A11中任意一项所述的方法，所述在所述操作指令包括动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码，包括：

根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据；

根据所述原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据；

根据所述第一计算图，生成硬件指令和数据描述符；

根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据；

根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码。

条款A13、根据条款A12所述的方法，所述根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据，包括：

通过解析所述动态操作指令，得到与所述动态操作指令对应的原始计算图；

根据所述动态操作指令的参数，得到所述原始模型数据。

条款A14、根据条款A12所述的方法，所述根据深度学习算法的原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据，包括：

读取所述原始计算图和所述原始模型数据；

识别所述原始计算图内的连续线性变换操作节点；

通过线性变换和常量折叠，对所述连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

条款A15、根据条款A12所述的方法，所述根据所述第一计算图，生成硬件指令和数据描述符，包括：

根据代价模型对所述第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

条款A16、根据条款A12所述的方法，所述根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据，包括：

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行数据对齐；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行维度变换；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行精度选择。

条款A17、根据条款A12所述的方法，所述根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码，包括：

将所述硬件指令和所述第二模型数据打包，得到所述深度学习算法的二进制代码。

条款A18、根据条款A11所述的方法，所述动态操作指令包括特例化操作指令和/或融合操作指令；其中，

所述特例化操作指令包括，对所述操作指令绑定输入参数后转换得到的操作指令；

所述融合操作指令包括，对多个所述操作指令根据调用顺序进行组合得到的操作指令。

条款A19、根据条款A18所述的方法，所述特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，

所述完全特例化操作指令包括，对所述操作指令绑定所有输入参数后转换得到的操作指令；

所述部分特例化操作指令包括，对所述操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于所述操作指令的输入参数数量的正整数；

所述伪特例化操作指令包括，对所述操作指令不绑定输入参数后直接转换得到的操作指令。

条款A20、根据条款A18所述的方法，所述融合操作指令的创建过程包括：

创建所述融合操作指令的名称；

根据待融合的操作指令确定融合操作子指令；

根据所述待融合的操作指令的调用顺序，确定所述融合操作子指令之间的操作连接关系；

根据所述操作连接关系，连接所述融合操作子指令，得到连接结果；

根据所述融合操作指令对应的用户指令，设置所述融合操作指令的输入参数和输出参数；

将所述名称、连接结果、输入参数和输出参数进行打包，得到所述融合操作指令。

条款A21、根据条款A1所述的方法，所述方法还包括：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

条款A22、根据条款A1所述的方法，所述操作数据还包括张量数据，其中，所述张量数据包括形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性。

条款A23、根据条款A22所述的方法，所述方法还包括：

对深度学习框架进行类扩展，将所述张量数据与所述操作指令封装于所述深度学习框架的数据内，实现所述深度学习框架与所述深度学习编程库接口的集成。

条款A24、一种深度学习算法的编译装置，包括：

操作数据接收模块，用于接收深度学习编程库接口传递的操作数据；

操作指令获取模块，用于获取所述操作数据包括的操作指令；

编译模块，用于判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

条款A25、根据条款A24所述的装置，所述操作指令根据所述深度学习编程库接口接收的用户指

令创建或调用。

条款A26、根据条款A25所述的装置，所述编译模块包括：

判断单元，用于判断所述操作指令的指令类型；

静态查找单元，用于在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码。

条款A27、根据条款A26所述的装置，所述静态查找单元用于：

根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码；

在查找结果为成功时，返回所述二进制代码，作为所述深度学习算法的二进制代码。

条款A28、根据条款A27所述的装置，所述静态查找单元还用于：

在查找结果为失败时，将所述静态操作指令作为动态操作指令，进行实时编译。

条款A29、根据条款A26所述的装置，所述静态操作指令包括定制操作指令、build-in操作指令和离线操作指令中的一个或多个；其中，

所述定制操作指令包括，根据所述操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；

所述build-in操作指令包括，所述深度学习编程库接口中包括的自有操作指令；

所述离线操作指令包括，预先编译完成的动态操作指令，其中，所述预先编译的结果保存在离线缓存中。

条款A30、根据条款A29所述的装置，所述定制操作指令对应的二进制代码的生成过程，包括：

根据所述操作指令的接口和数据结构定义，对所述操作指令对应的用户指令进行封装，得到封装后的用户指令；

对所述封装后的用户指令进行编译，得到编译结果；

将所述编译结果以动态链接或静态链接的方式，插入至所述静态操作池内，得到所述定制操作指令对应的二进制代码。

条款A31、根据条款A29所述的装置，所述静态操作池包括静态操作源文件，所述静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，

所述静态代码段用于保存所述build-in操作指令对应的二进制代码；

所述动态代码段用于保存所述定制操作指令对应的二进制代码；

所述静态数据段用于保存与所述build-in操作指令对应的张量数据；

所述动态数据段用于保存与所述定制操作指令对应的张量数据。

条款A32、根据条款A31所述的装置，所述静态操作池还包括离线缓存，所述离线缓存包括离线文件和索引表；其中，

所述离线文件，用于保存离线操作指令的预先编译结果；

所述索引表，用于指示所述离线操作指令的预先编译结果在所述离线文件中的位置。

条款A33、根据条款A29所述的装置，所述静态查找单元进一步用于：

根据所述静态操作指令被创建时指定的名称，在所述静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与所述名称对应的二进制代码。

条款A34、根据条款A26所述的装置，所述编译模块还包括动态编译单元，用于：

在所述指令类型为动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码。

条款A35、根据条款A24至A34中任意一项所述的装置，所述动态编译单元包括：

原始数据获取子单元，用于根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据；

协同处理子单元，用于根据所述原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据；

硬件指令生成子单元，用于根据所述第一计算图，生成硬件指令和数据描述符；

模型数据处理子单元，用于根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据；

二进制代码生成子单元，用于根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码。

条款A36、根据条款A35所述的装置，所述原始数据获取子单元用于：

通过解析所述动态操作指令，得到与所述动态操作指令对应的原始计算图；

根据所述动态操作指令的参数，得到所述原始模型数据。

条款A37、根据条款A35所述的装置，所述协同处理子单元用于：

读取所述原始计算图和所述原始模型数据；

识别所述原始计算图内的连续线性变换操作节点；

通过线性变换和常量折叠，对所述连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

条款A38、根据条款A35所述的装置，所述硬件指令生成子单元用于：

根据代价模型对所述第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

条款A39、根据条款A35所述的装置，所述模型数据处理子单元用于：

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行数据对齐；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行维度变换；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行精度选择。

条款A40、根据条款A35所述的装置，所述二进制代码生成子单元用于：

将所述硬件指令和所述第二模型数据打包，得到所述深度学习算法的二进制代码。

条款A41、根据条款A34所述的装置，所述动态操作指令包括特例化操作指令和/或融合操作指令；其中，

所述特例化操作指令包括，对所述操作指令绑定输入参数后转换得到的操作指令；

所述融合操作指令包括，对多个所述操作指令根据调用顺序进行组合得到的操作指令。

条款A42、根据条款A41所述的装置，所述特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，

所述完全特例化操作指令包括，对所述操作指令绑定所有输入参数后转换得到的操作指令；

所述部分特例化操作指令包括，对所述操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于所述操作指令的输入参数数量的正整数；

所述伪特例化操作指令包括，对所述操作指令不绑定输入参数后直接转换得到的操作指令。

条款A43、根据条款A41所述的装置，所述融合操作指令的创建过程包括：

创建所述融合操作指令的名称；

根据待融合的操作指令确定融合操作子指令；

根据所述待融合的操作指令的调用顺序，确定所述融合操作子指令之间的操作连接关系；

根据所述操作连接关系，连接所述融合操作子指令，得到连接结果；

根据所述融合操作指令对应的用户指令，设置所述融合操作指令的输入参数和输出参数；

将所述名称、连接结果、输入参数和输出参数进行打包，得到所述融合操作指令。

条款A44、根据条款A24所述的装置，所述装置还包括执行模块，用于：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

条款A45、根据条款A24所述的装置，所述操作数据还包括张量数据，其中，所述张量数据包括形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性。

条款A46、根据条款A45所述的装置，所述装置还用于：

对深度学习框架进行类扩展，将所述张量数据与所述操作指令封装于所述深度学习框架的数据内，实现所述深度学习框架与所述深度学习编程库接口的集成。

条款A47、一种深度学习运算装置，所述深度学习运算装置包括一个或多个如条款A24-A46任一项所述的深度学习算法的编译装置，所述深度学习运算装置用于完成设定的深度学习运算。

条款A48、一种组合运算装置，所述组合运算装置包括一个或多个如条款A47任一项所述的深度学习运算装置，通用互联接口和其他处理装置；

所述深度学习运算装置与所述其他处理装置进行交互，共同完成用户指定的计算操作。

条款A49、一种深度学习芯片，所述深度学习芯片包括：

如条款A24-46任一项所述的深度学习算法的编译装置；或者，

如条款A47所述的深度学习运算装置；或者，

如条款A48所述的组合运算装置。

条款A50、一种电子设备，所述电子设备包括：

如条款A24-A46任一项所述的深度学习算法的编译装置；或者，

如条款A47所述的深度学习运算装置；或者，

如条款A48所述的组合运算装置；或者，

如条款A49所述的深度学习芯片。

以上对本公开实施例进行了详细介绍，本文中应用了具体个例对本公开的原理及实施方式进行了阐述，以上实施例的说明仅用于帮助理解本公开的方法及其核心思想。同时，本领域技术人员依据本公开的思想，基于本公开的具体实施方式及应用范围上做出的改变或变形之处，都属于本公开保护的范围。综上所述，本说明书内容不应理解为对本公开的限制。

权 利 要 求 书

1、一种深度学习算法的编译方法，其特征在于，所述方法包括：

接收深度学习编程库接口传递的操作数据；

获取所述操作数据包括的操作指令；

判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

2、根据权利要求1所述的方法，其特征在于，所述操作数据根据所述深度学习编程库接口接收的用户指令创建或调用。

3、根据权利要求2所述的方法，其特征在于，所述判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码，包括：

判断所述操作指令的指令类型；

在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码。

4、根据权利要求3所述的方法，其特征在于，所述在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码，包括：

根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码；

在查找结果为成功时，返回所述二进制代码，作为所述深度学习算法的二进制代码。

5、根据权利要求4所述的方法，其特征在于，所述在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码，还包括：

在查找结果为失败时，将所述静态操作指令作为动态操作指令，进行实时编译。

6、根据权利要求3所述的方法，其特征在于，所述静态操作指令包括定制操作指令、build-in操作指令和离线操作指令中的一个或多个；其中，

所述定制操作指令包括，根据所述操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；

所述build-in操作指令包括，所述深度学习编程库接口中包括的自有操作指令；

所述离线操作指令包括，预先编译完成的动态操作指令，其中，所述预先编译的结果保存在离线缓存中。

7、根据权利要求6所述的方法，其特征在于，所述定制操作指令对应的二进制代码的生成过程，包括：

根据所述操作指令的接口和数据结构定义，对所述操作指令对应的用户指令进行封装，得到封装后的用户指令；

对所述封装后的用户指令进行编译，得到编译结果；

将所述编译结果以动态链接或静态链接的方式，插入至所述静态操作池内，得到所述定制操作指令对应的二进制代码。

8、根据权利要求6所述的方法，其特征在于，所述静态操作池包括静态操作源文件，所述静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，

所述静态代码段用于保存所述build-in操作指令对应的二进制代码；

所述动态代码段用于保存所述定制操作指令对应的二进制代码；

所述静态数据段用于保存与所述build-in操作指令对应的张量数据；

所述动态数据段用于保存与所述定制操作指令对应的张量数据。

9、根据权利要求8所述的方法，其特征在于，所述静态操作池还包括离线缓存，所述离线缓存包括离线文件和索引表；其中，

所述离线文件，用于保存离线操作指令的预先编译结果；

所述索引表，用于指示所述离线操作指令的预先编译结果在所述离线文件中的位置。

10、根据权利要求6所述的方法，其特征在于，所述根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码，包括：

根据所述静态操作指令被创建时指定的名称，在所述静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与所述名称对应的二进制代码。

11、根据权利要求3所述的方法，其特征在于，所述判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码，还包括：

在所述指令类型为动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码。

12、根据权利要求1至11中任意一项所述的方法，其特征在于，所述在所述操作指令包括动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码，包括：

根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据；

根据所述原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据；

根据所述第一计算图，生成硬件指令和数据描述符；

根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据；

根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码。

13、根据权利要求12所述的方法，其特征在于，所述根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据，包括：

通过解析所述动态操作指令，得到与所述动态操作指令对应的原始计算图；

根据所述动态操作指令的参数，得到所述原始模型数据。

14、根据权利要求12所述的方法，其特征在于，所述根据深度学习算法的原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据，包括：

读取所述原始计算图和所述原始模型数据；

识别所述原始计算图内的连续线性变换操作节点；

通过线性变换和常量折叠，对所述连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

15、根据权利要求12所述的方法，其特征在于，所述根据所述第一计算图，生成硬件指令和数据描述符，包括：

根据代价模型对所述第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

16、根据权利要求12所述的方法，其特征在于，所述根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据，包括：

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行数据对齐；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行维度变换；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行精度选择。

17、根据权利要求12所述的方法，其特征在于，所述根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码，包括：

将所述硬件指令和所述第二模型数据打包，得到所述深度学习算法的二进制代码。

18、根据权利要求11所述的方法，其特征在于，所述动态操作指令包括特例化操作指令和/或融合操作指令；其中，

所述特例化操作指令包括，对所述操作指令绑定输入参数后转换得到的操作指令；

所述融合操作指令包括，对多个所述操作指令根据调用顺序进行组合得到的操作指令。

19、根据权利要求18所述的方法，其特征在于，所述特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，

所述完全特例化操作指令包括，对所述操作指令绑定所有输入参数后转换得到的操作指令；

所述部分特例化操作指令包括，对所述操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于所述操作指令的输入参数数量的正整数；

所述伪特例化操作指令包括，对所述操作指令不绑定输入参数后直接转换得到的操作指令。

20、根据权利要求18所述的方法，其特征在于，所述融合操作指令的创建过程包括：

创建所述融合操作指令的名称；

根据待融合的操作指令确定融合操作子指令；

根据所述待融合的操作指令的调用顺序，确定所述融合操作子指令之间的操作连接关系；

根据所述操作连接关系，连接所述融合操作子指令，得到连接结果；

根据所述融合操作指令对应的用户指令，设置所述融合操作指令的输入参数和输出参数；

将所述名称、连接结果、输入参数和输出参数进行打包，得到所述融合操作指令。

21、根据权利要求1所述的方法，其特征在于，所述方法还包括：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

22、根据权利要求1所述的方法，其特征在于，所述操作数据还包括张量数据，其中，所述张量数据包括形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性。

23、根据权利要求22所述的方法，其特征在于，所述方法还包括：

对深度学习框架进行类扩展，将所述张量数据与所述操作指令封装于所述深度学习框架的数据内，实现所述深度学习框架与所述深度学习编程库接口的集成。

24、一种深度学习算法的编译装置，其特征在于，包括：

操作数据接收模块，用于接收深度学习编程库接口传递的操作数据；

操作指令获取模块，用于获取所述操作数据包括的操作指令；

编译模块，用于判断所述操作指令的指令类型，根据判断结果执行与所述指令类型相应的编译操作，得到所述深度学习算法的二进制代码。

25、根据权利要求24所述的装置，其特征在于，所述操作指令根据所述深度学习编程库接口接收的用户指令创建或调用。

26、根据权利要求25所述的装置，其特征在于，所述编译模块包括：

判断单元，用于判断所述操作指令的指令类型；

静态查找单元，用于在所述指令类型为静态操作指令时，根据所述静态操作指令的名称，在静态操作池中查找对应的二进制代码，作为所述深度学习算法的二进制代码。

27、根据权利要求26所述的装置，其特征在于，所述静态查找单元用于：

根据所述静态操作指令的名称，在所述静态操作池中查找与所述名称对应的二进制代码；

在查找结果为成功时，返回所述二进制代码，作为所述深度学习算法的二进制代码。

28、根据权利要求27所述的装置，其特征在于，所述静态查找单元还用于：

在查找结果为失败时，将所述静态操作指令作为动态操作指令，进行实时编译。

29、根据权利要求26所述的装置，其特征在于，所述静态操作指令包括定制操作指令、build-in操作指令和离线操作指令中的一个或多个；其中，

所述定制操作指令包括，根据所述操作指令的编码方式和封装形式，实现的具有自定义功能的操作指令；

所述build-in操作指令包括，所述深度学习编程库接口中包括的自有操作指令；

所述离线操作指令包括，预先编译完成的动态操作指令，其中，所述预先编译的结果保存在离线缓存中。

30、根据权利要求29所述的装置，其特征在于，所述定制操作指令对应的二进制代码的生成过程，包括：

根据所述操作指令的接口和数据结构定义，对所述操作指令对应的用户指令进行封装，得到封装后的用户指令；

对所述封装后的用户指令进行编译，得到编译结果；

将所述编译结果以动态链接或静态链接的方式，插入至所述静态操作池内，得到所述定制操作指令对应的二进制代码。

31、根据权利要求29所述的装置，其特征在于，所述静态操作池包括静态操作源文件，所述静态操作源文件，包括静态代码段、静态数据段、动态代码段和动态数据段；其中，

所述静态代码段用于保存所述build-in操作指令对应的二进制代码；

所述动态代码段用于保存所述定制操作指令对应的二进制代码；

所述静态数据段用于保存与所述build-in操作指令对应的张量数据；

所述动态数据段用于保存与所述定制操作指令对应的张量数据。

32、根据权利要求31所述的装置，其特征在于，所述静态操作池还包括离线缓存，所述离线缓存包括离线文件和索引表；其中，

所述离线文件，用于保存离线操作指令的预先编译结果；

所述索引表，用于指示所述离线操作指令的预先编译结果在所述离线文件中的位置。

33、根据权利要求29所述的装置，其特征在于，所述静态查找单元进一步用于：

根据所述静态操作指令被创建时指定的名称，在所述静态操作池中，依次查找定制操作对应的二进制代码、build-in操作对应的二进制代码和离线操作对应的二进制代码，得到与所述名称对应的二进制代码。

34、根据权利要求26所述的装置，其特征在于，所述编译模块还包括动态编译单元，用于：

在所述指令类型为动态操作指令时，对所述动态操作指令进行实时编译，得到实时编译结果，作为所述深度学习算法的二进制代码。

35、根据权利要求24至34中任意一项所述的装置，其特征在于，所述动态编译单元包括：

原始数据获取子单元，用于根据所述动态操作指令，得到与所述动态操作指令对应的原始计算图和原始模型数据；

协同处理子单元，用于根据所述原始计算图和原始模型数据，进行面向所述深度学习算法的协同处理，得到第一计算图和第一模型数据；

硬件指令生成子单元，用于根据所述第一计算图，生成硬件指令和数据描述符；

模型数据处理子单元，用于根据所述数据描述符，对所述第一模型数据进行面向硬件平台的处理，得到第二模型数据；

二进制代码生成子单元，用于根据所述硬件指令和所述第二模型数据，得到所述深度学习算法的二进制代码。

36、根据权利要求35所述的装置，其特征在于，所述原始数据获取子单元用于：

通过解析所述动态操作指令，得到与所述动态操作指令对应的原始计算图；

根据所述动态操作指令的参数，得到所述原始模型数据。

37、根据权利要求35所述的装置，其特征在于，所述协同处理子单元用于：

读取所述原始计算图和所述原始模型数据；

识别所述原始计算图内的连续线性变换操作节点；

通过线性变换和常量折叠，对所述连续线性变换操作节点进行处理，得到第一计算图和第一模型数据。

38、根据权利要求35所述的装置，其特征在于，所述硬件指令生成子单元用于：

根据代价模型对所述第一计算图进行处理，结合启发式搜索策略，得到硬件指令和数据描述符。

39、根据权利要求35所述的装置，其特征在于，所述模型数据处理子单元用于：

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行数据对齐；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行维度变换；或者，

根据所述数据描述符，按照硬件平台内的运算要求，对所述第一模型数据进行精度选择。

40、根据权利要求35所述的装置，其特征在于，所述二进制代码生成子单元用于：

将所述硬件指令和所述第二模型数据打包，得到所述深度学习算法的二进制代码。

41、根据权利要求34所述的装置，其特征在于，所述动态操作指令包括特例化操作指令和/或融合操作指令；其中，

所述特例化操作指令包括，对所述操作指令绑定输入参数后转换得到的操作指令；

所述融合操作指令包括，对多个所述操作指令根据调用顺序进行组合得到的操作指令。

42、根据权利要求41所述的装置，其特征在于，所述特例化操作指令包括完全特例化操作指令、部分特例化操作指令和伪特例化操作指令；其中，

所述完全特例化操作指令包括，对所述操作指令绑定所有输入参数后转换得到的操作指令；

所述部分特例化操作指令包括，对所述操作指令绑定N个输入参数后转换得到的操作指令，其中N为小于所述操作指令的输入参数数量的正整数；

所述伪特例化操作指令包括，对所述操作指令不绑定输入参数后直接转换得到的操作指令。

43、根据权利要求41所述的装置，其特征在于，所述融合操作指令的创建过程包括：

创建所述融合操作指令的名称；

根据待融合的操作指令确定融合操作子指令；

根据所述待融合的操作指令的调用顺序，确定所述融合操作子指令之间的操作连接关系；

根据所述操作连接关系，连接所述融合操作子指令，得到连接结果；

根据所述融合操作指令对应的用户指令，设置所述融合操作指令的输入参数和输出参数；

将所述名称、连接结果、输入参数和输出参数进行打包，得到所述融合操作指令。

44、根据权利要求24所述的装置，其特征在于，所述装置还包括执行模块，用于：通过运行时系统，将所述深度学习算法的二进制代码在深度学习处理器中执行。

45、根据权利要求24所述的装置，其特征在于，所述操作数据还包括张量数据，其中，所述张量数据包括形状属性、逻辑数据类型属性、物理数据类型属性和物理布局属性。

46、根据权利要求45所述的装置，其特征在于，所述装置还用于：

对深度学习框架进行类扩展，将所述张量数据与所述操作指令封装于所述深度学习框架的数据内，实现所述深度学习框架与所述深度学习编程库接口的集成。

47、一种深度学习运算装置，其特征在于，所述深度学习运算装置包括一个或多个如权利要求24-46任一项所述的深度学习算法的编译装置，所述深度学习运算装置用于完成设定的深度学习运算。

48、一种组合运算装置，其特征在于，所述组合运算装置包括如权利要求47所述的深度学习运算装置，通用互联接口和其他处理装置；

所述深度学习运算装置与所述其他处理装置进行交互，共同完成用户指定的计算操作。

49、一种深度学习芯片，其特征在于，所述深度学习芯片包括：

如权利要求24-46任一项所述的深度学习算法的编译装置；或者，

如权利要求47所述的深度学习运算装置；或者，

如权利要求48所述的组合运算装置。

50、一种电子设备，其特征在于，所述电子设备包括：

如权利要求24-46任一项所述的深度学习算法的编译装置；或者，

如权利要求47所述的深度学习运算装置；或者，

如权利要求48所述的组合运算装置；或者，

如权利要求49所述的深度学习芯片。

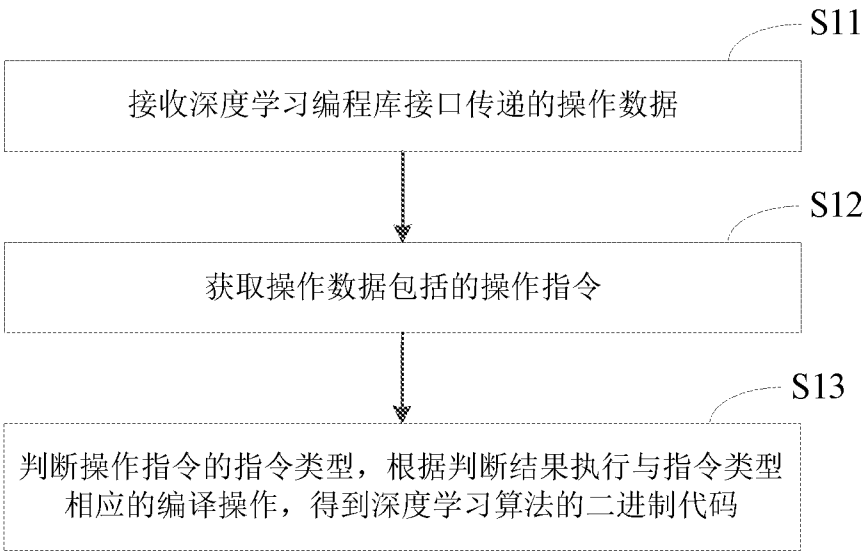


图 1

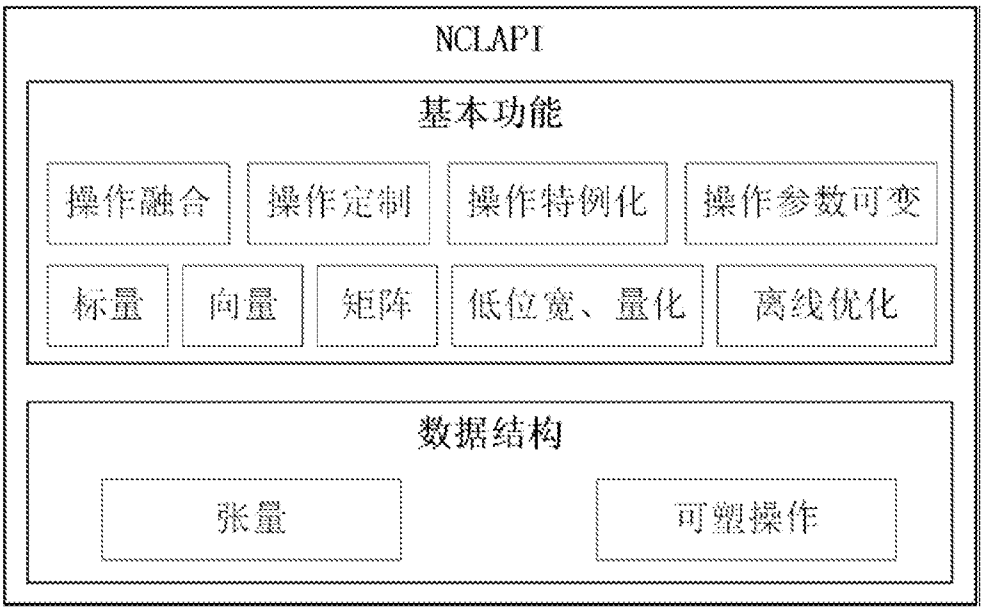


图 2

属性	分类	含义
形状	可见属性	多维数据的维度、每个维度的大小、各维度在主机内存中的排布顺序
逻辑数据类型	可见属性	数据在主机内存中存储的精度
物理数据类型	不可见属性	数据在设备内存中存储的精度
物理布局	不可见属性	数据在设备内存中的布局，包括对齐、拆分、排布顺序等

图 3

张量类型	形状符号约定	符号含义
卷积输入和输出	*N*	数据的批数
	C	特征图数量
	H	特征图高度
	W	特征图宽度
卷积权值	*CO*	输出特征图数量
	CI	输入特征图数量
	Kh	卷积核高度
	Kw	卷积核宽度
RNN 输入	*T*	输入序列的长度（比如一句话中单词的个数）
	N	输入序列的批数（比如多少句话）
	C	每个序列元素的大小（比如单个单词的长度）

图 4

操作类型	描述	支持原地算法	操作类型	描述	支持原地算法
conv	卷积	✗	conv3d	三维卷积	✗
maxpool	最大池化	✗	avgpool	均值池化	✗
lrn	LRN	✗	bn	批规范化	✓
active	激活	✓	upsample	上采样	✗
fc	全连接	✗	softmax	Softmax	✗
scale	像素缩放	✓	reshape	尺寸缩放	✗
concat	数据拼接	✗	split	数据拆分	✗
add	张量加法	✓	sub	张量减法	✓
mult	张量乘法	✓	divide	张量除法	✓
nop	空操作	✗	deconv	反卷积	✗
basicrnn	基本 RNN	✗	lstmcell	基本 LSTM 单元	✗
grucell	GRU 单元	✗	roipool	roi 池化	✗
transpose	转置	✗	depthwiseconv	深度卷积	✗

图 5

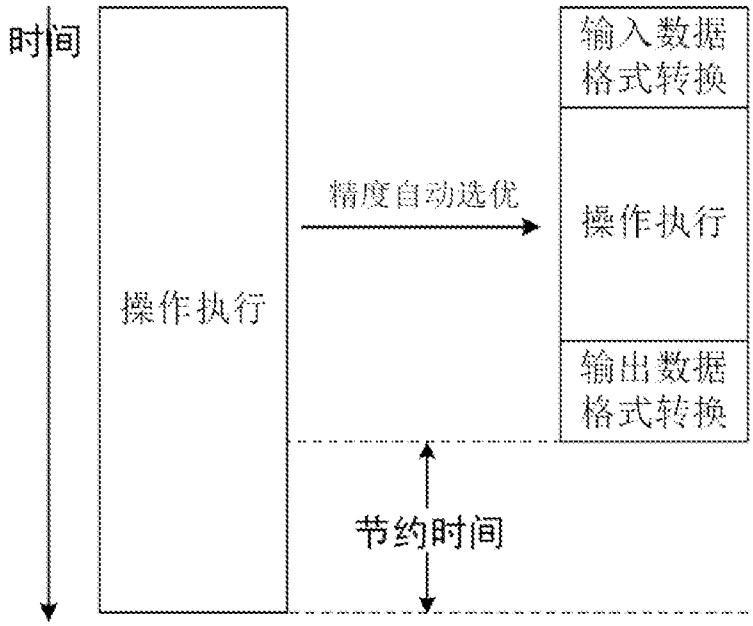

时间
操作执行
精度自动选优
输入数据格式转换
操作执行
输出数据格式转换
节约时间

图 6

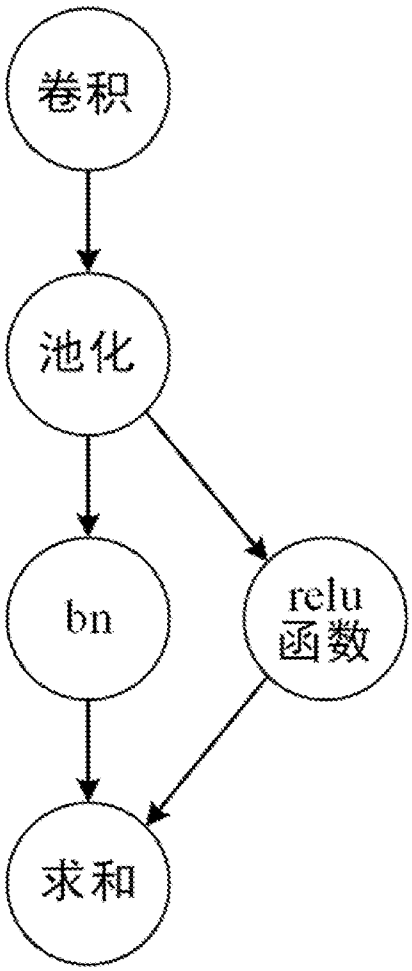

卷积
池化
bn
relu函数
求和

图 7

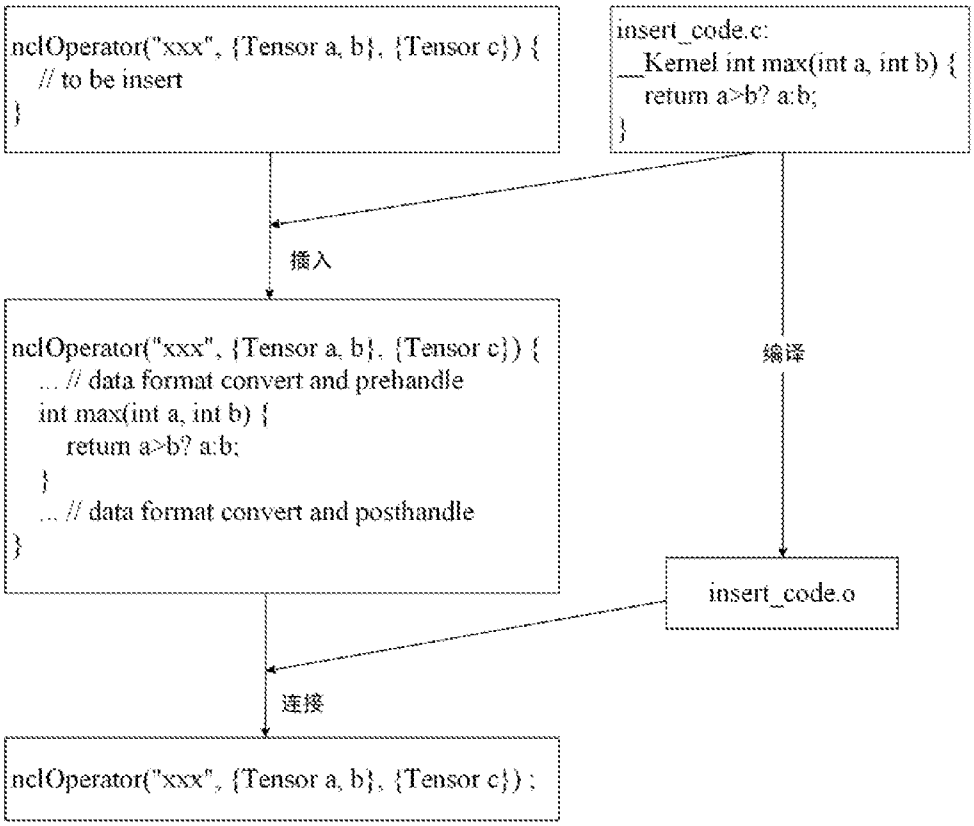

nclOperator("xxx", {Tensor a, b}, {Tensor c}) {
// to be insert
}
insert_code.c:
__Kernel int max(int a, int b) {
return a>b? a:b;
}
插入
编译
nclOperator("xxx", {Tensor a, b}, {Tensor c}) {
... // data format convert and prehandle
int max(int a, int b) {
return a>b? a:b;
}
... // data format convert and posthandle
}
insert_code.o
连接
nclOperator("xxx", {Tensor a, b}, {Tensor c}) ;

图 8

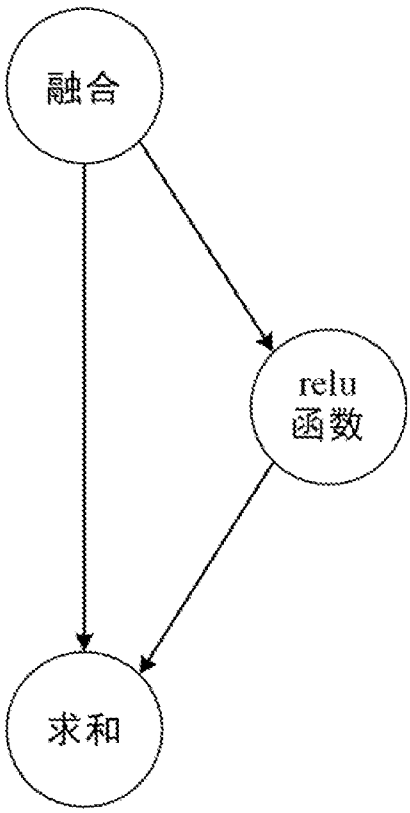

融合
relu
函数
求和

图 9

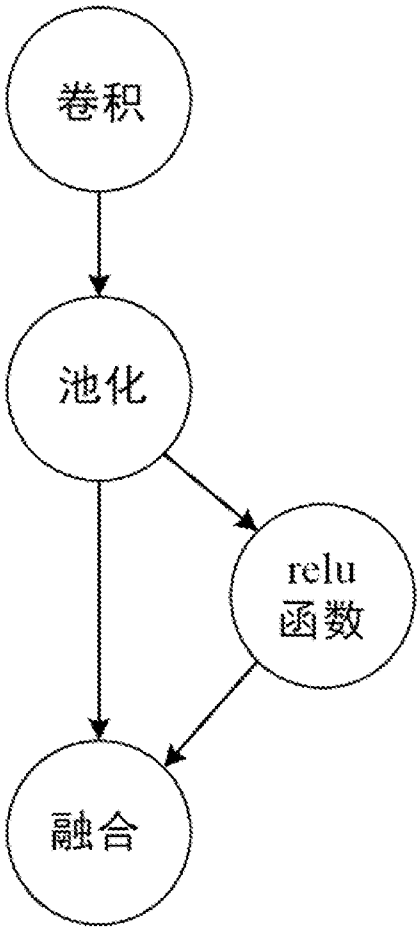


图 10

接口声明	接口功能
nclOperator nclCreateFusionOperator(string op_type_name)	创建融合操作
void nclAddFusionOperator(nclOperator fusion, nclOperator op)	逐个添加待融合的子操作
void nclSetFusionOperators(nclOperator fusion, nclOperator ops[])	一次性添加所有待融合的子操作
void nclAddFusionInput(nclOperator fusion, nclOperator op, int i)	把 op 的第 i 个输入参数作为融合操作的输入参数，i 从 0 开始计数
void nclAddFusionOutput(nclOperator fusion, nclOperator op, int i)	把 op 的第 i 个输出参数作为融合操作的输出参数，i 从 0 开始计数
void nclFuseOperator(nclOperator fusion)	进行操作融合

图 11

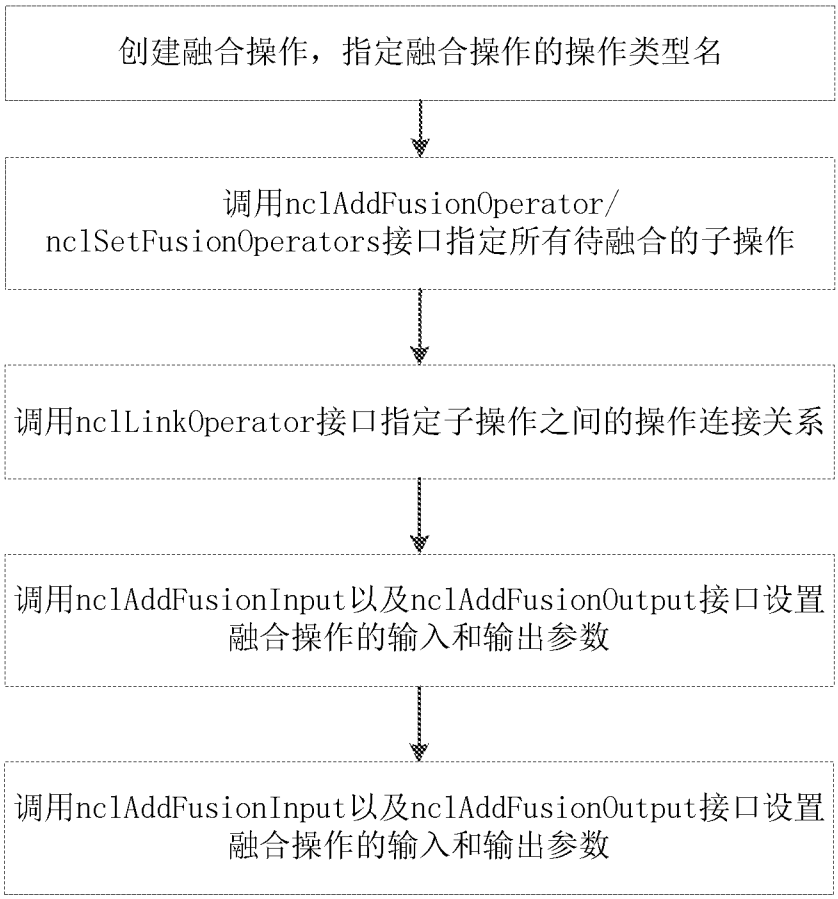

创建融合操作，指定融合操作的操作类型名
调用nclAddFusionOperator/
nclSetFusionOperators接口指定所有待融合的子操作
调用nclLinkOperator接口指定子操作之间的操作连接关系
调用nclAddFusionInput以及nclAddFusionOutput接口设置
融合操作的输入和输出参数
调用nclAddFusionInput以及nclAddFusionOutput接口设置
融合操作的输入和输出参数

图 12

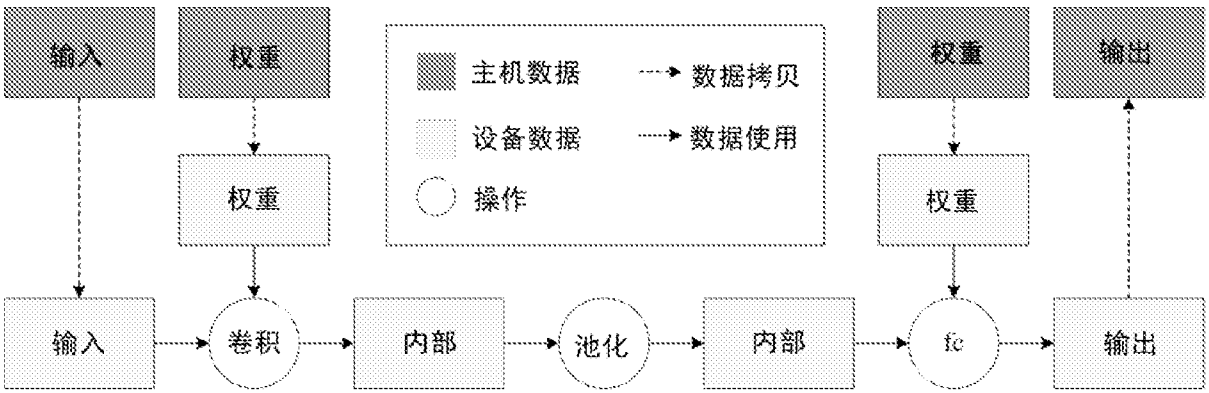
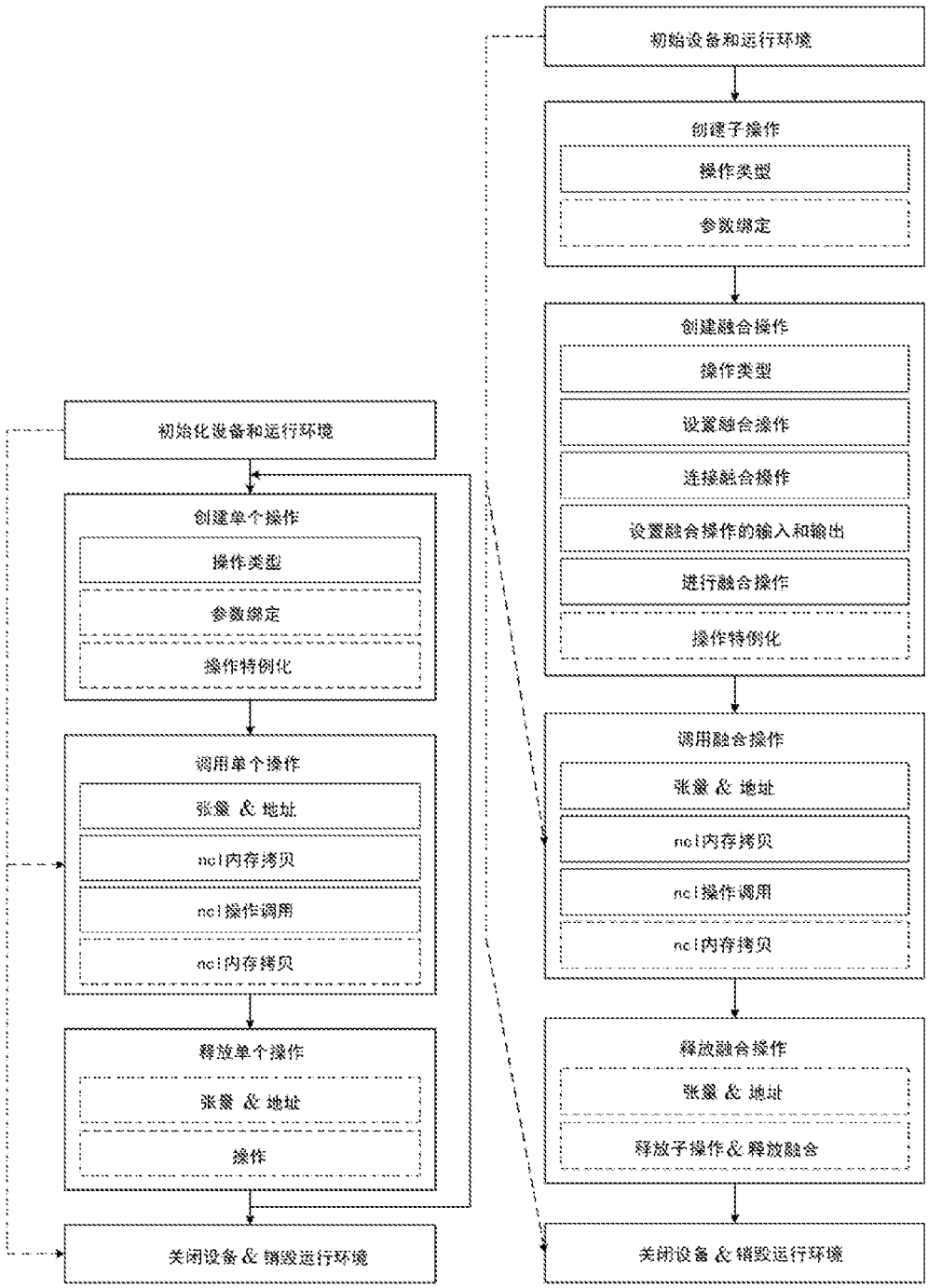

输入
权重
主机数据
数据拷贝
设备数据
数据使用
操作
权重
输出
权重
权重
输入
卷积
内部
池化
内部
fc
输出

图 13



（a）命令式编程模型 （b）声明式编程模型

图 14

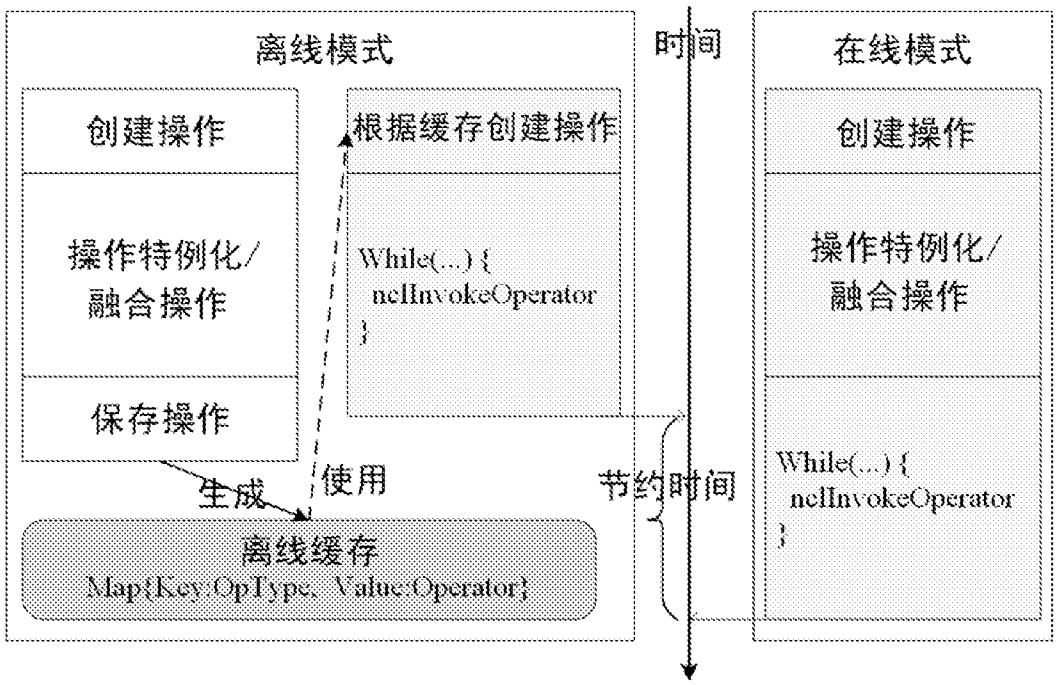


图 15

接口名	接口功能
void nclSaveOperator(nclOperator op, char op_type[])	保存离线操作
nclOperator nclCreateOperator(OpType type_string)	从离线缓存中创建操作

图 16

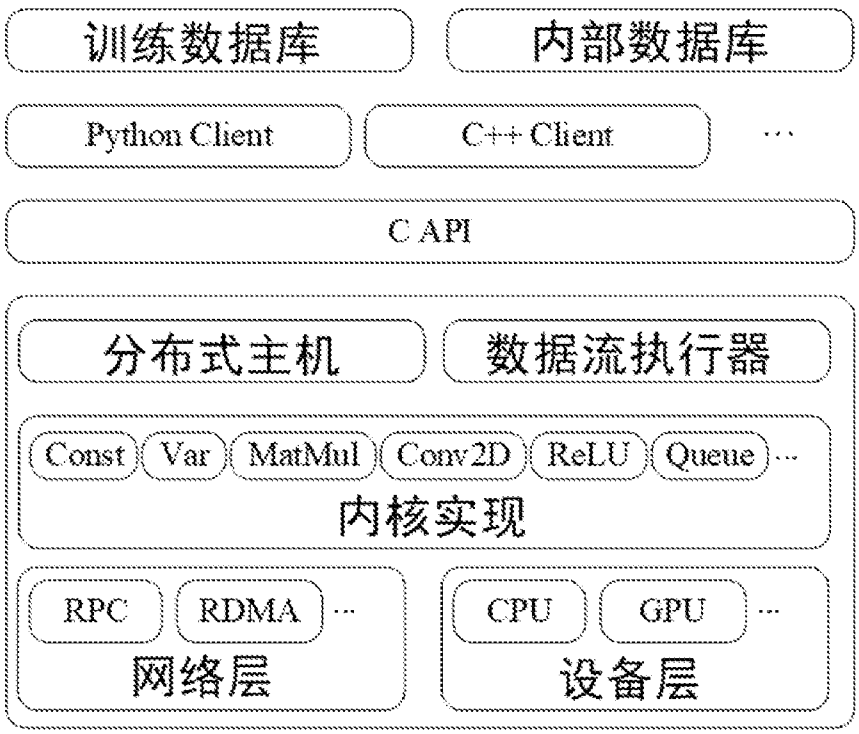


图 17

	NCLAPI	TensorRT	cuDNN	MIOpen	ACL	MKL-DNN
分类	混合	基于图	基于算子	基于算子	混合	基于图
编程范式	混合	声明式	命令式	命令式	混合式	声明式
支持操作融合	✓	✓	✗	✗	✓	✓
支持的操作数量	26	22+	18+	9+	100+	34+
支持操作定制	✓	✓	✗	✗	✗	✗
支持操作特例化	✓	✓	✗	✗	✓	✓
操作参数运行时可变	✓	✗	✓	✓	✗	✗
离线模式	✓	✓	✗	✗	✗	✗

图 18

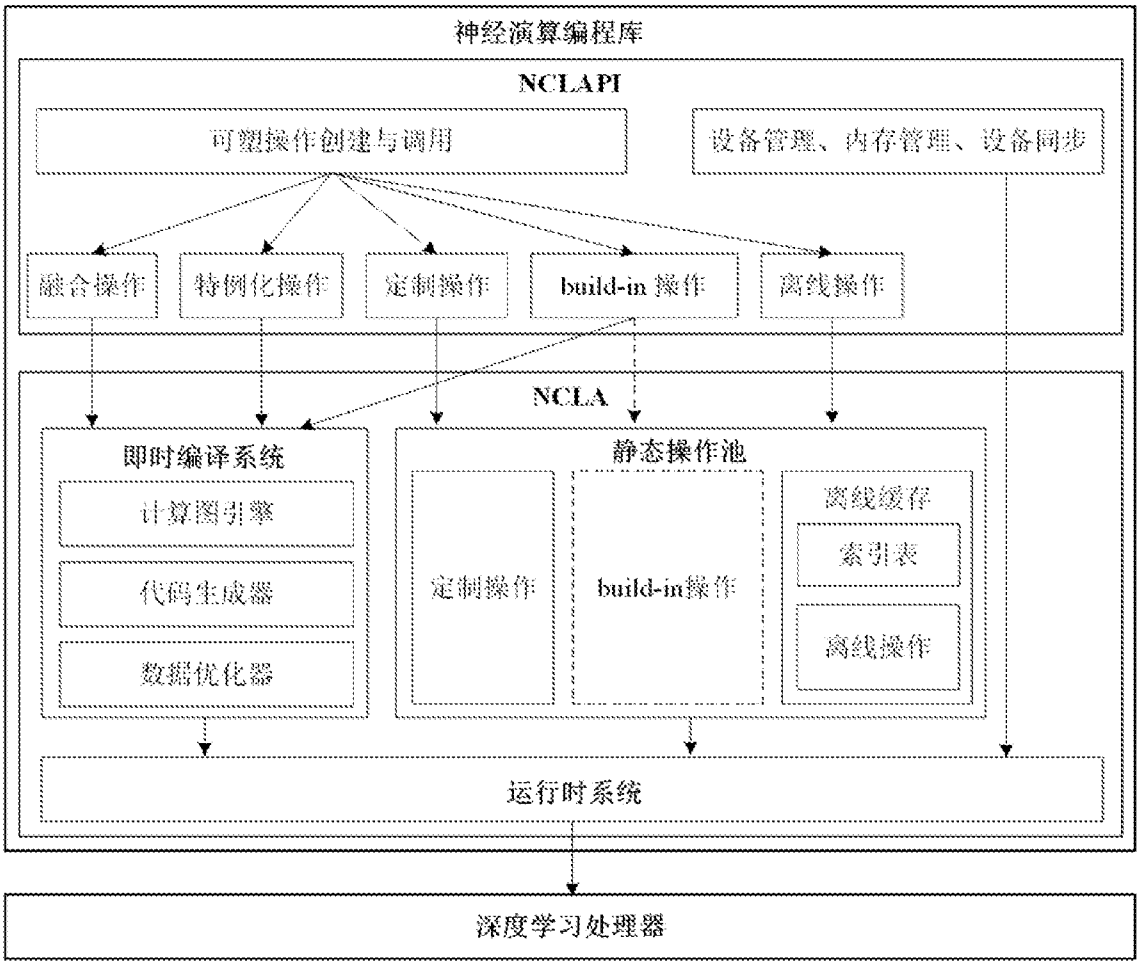


图 19

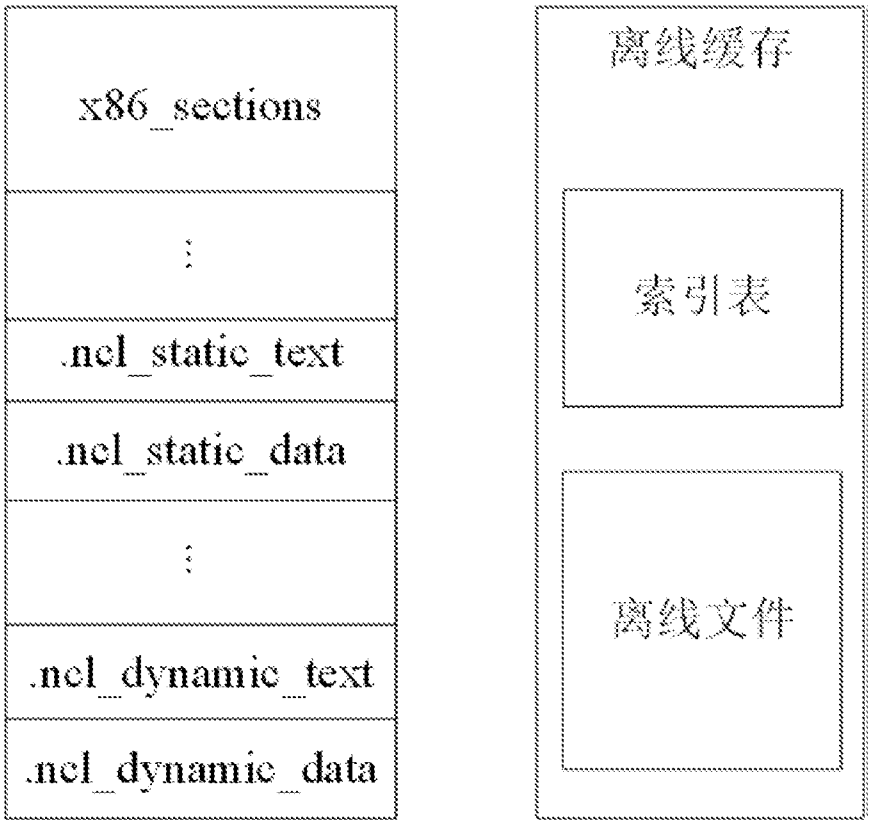

x86_sections
⋮
.ncl_static_text
.ncl_static_data
⋮
.ncl_dynamic_text
.ncl_dynamic_data
离线缓存
索引表
离线文件

图 20


原始计算图 + 常量数据
计算图引擎
优化后计算图 + 常量数据'
代码生成器
目标平台代码 + 常量数据' + 数据描述符
数据优化器
二进制代码

图 21

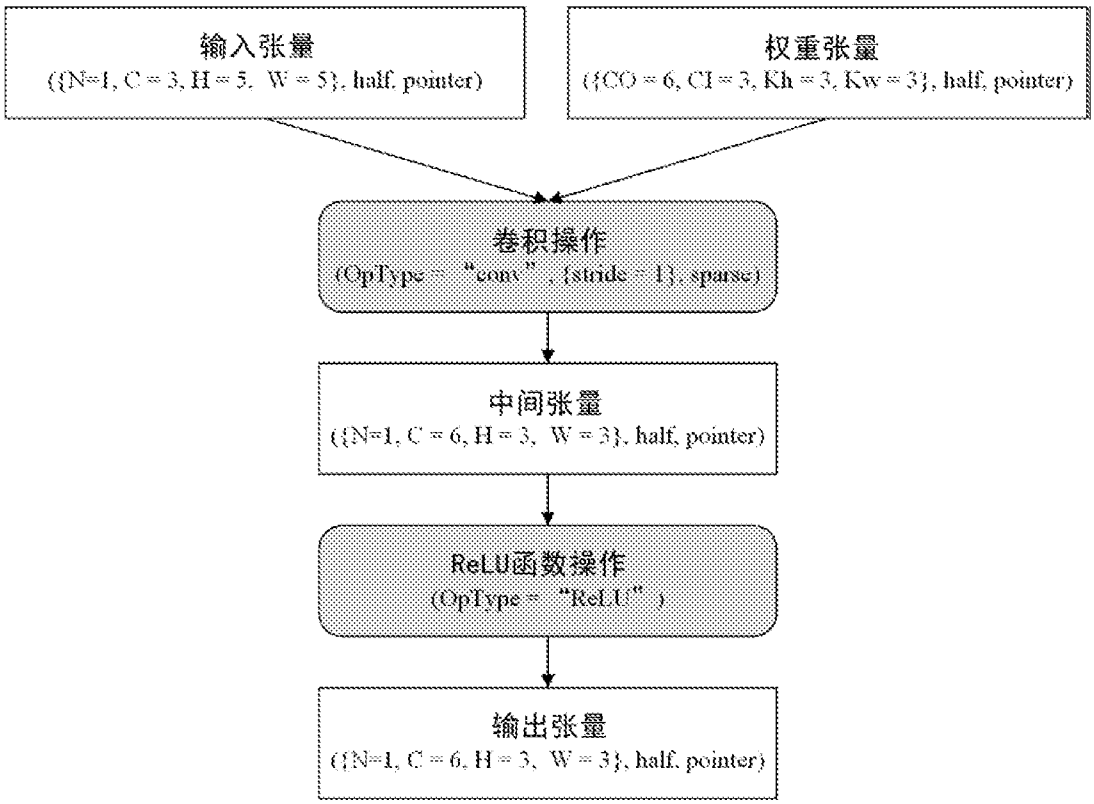

输入张量
({N=1, C = 3, H = 5, W = 5}, half, pointer)
权重张量
({CO = 6, CI = 3, Kh = 3, Kw = 3}, half, pointer)
卷积操作
(OpType = "conv", {stride = 1}, sparse)
中间张量
({N=1, C = 6, H = 3, W = 3}, half, pointer)
ReLU函数操作
(OpType = "ReLU")
输出张量
({N=1, C = 6, H = 3, W = 3}, half, pointer)

图 22

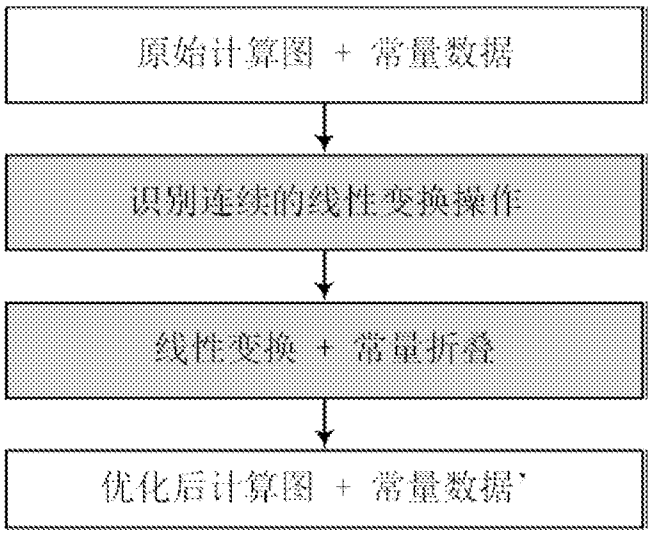

原始计算图 + 常量数据
识别连续的线性变换操作
线性变换 + 常量折叠
优化后计算图 + 常量数据'

图 23


卷积
批规范化
缩放

图 24

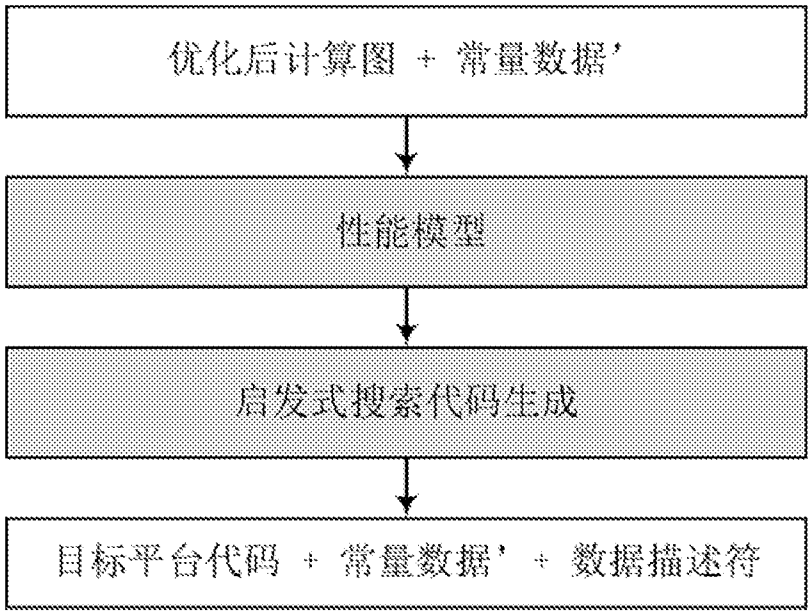

优化后计算图 + 常量数据'
性能模型
启发式搜索代码生成
目标平台代码 + 常量数据' + 数据描述符

图 25

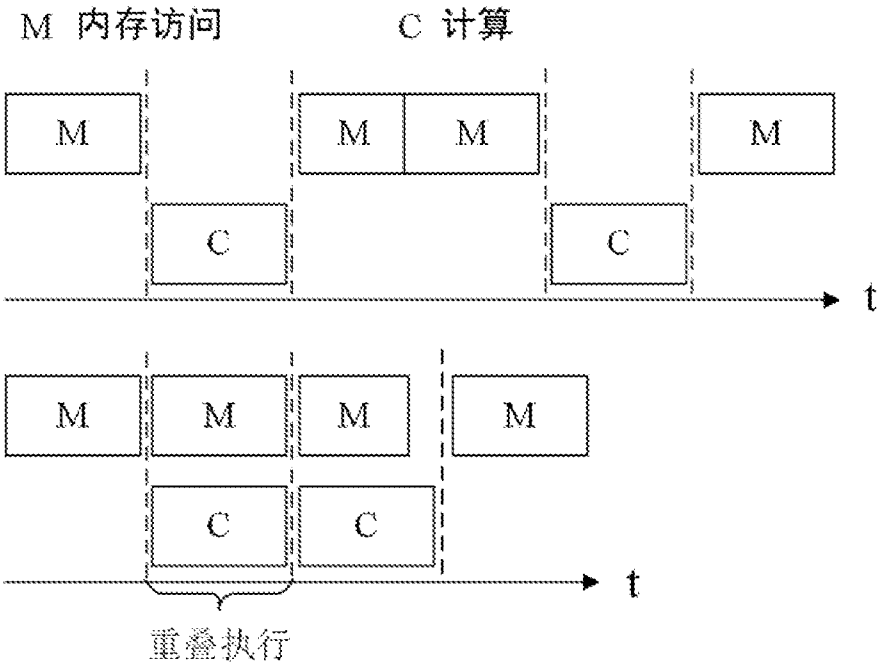

M 内存访问
C 计算
M
M
M
M
C
C
t
M
M
M
M
C
C
t
重叠执行

图 26

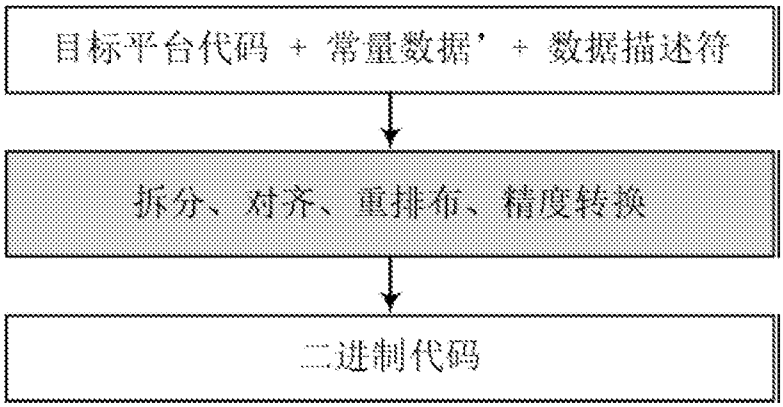

目标平台代码 + 常量数据' + 数据描述符
拆分、对齐、重排布、精度转换
二进制代码

图 27

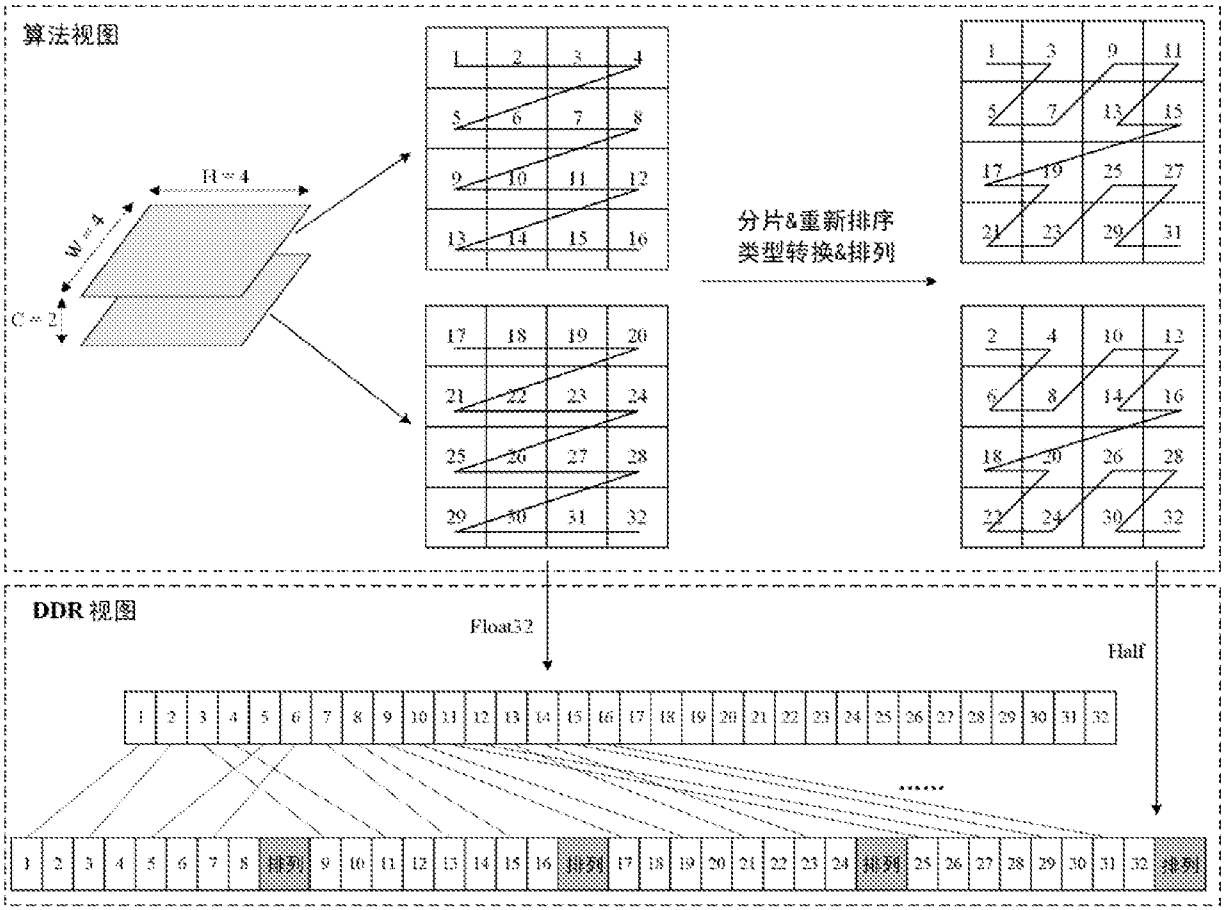


图 28

模块	功能	功能描述
内存管理	malloc	内存分配
	free	内存释放
	memcpy	内存拷贝
	lock	锁页内存
设备管理	open	打开设备
	close	关闭设备
	p2p	设备通讯
执行控制	invoke	异步执行
	sync	操作同步
	schedule	任务调度

图 29

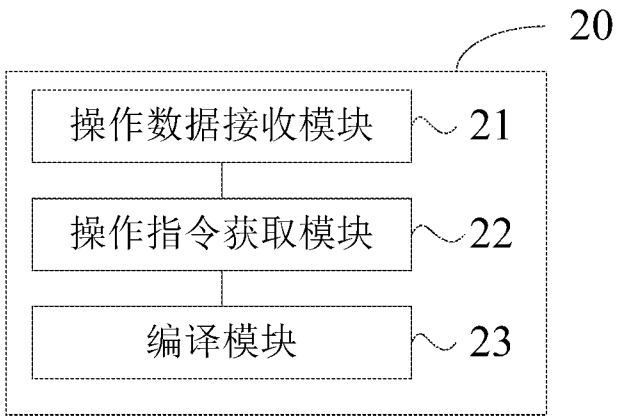

20
操作数据接收模块
21
操作指令获取模块
22
编译模块
23

图 30

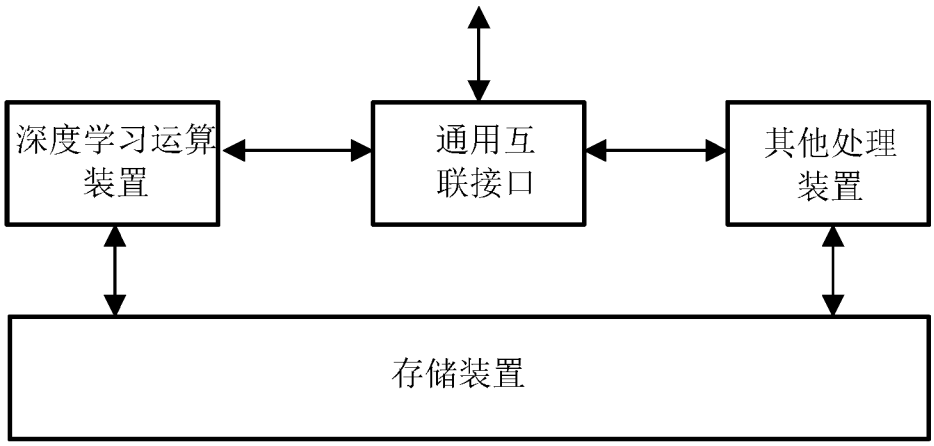

深度学习运算装置
通用互联接口
其他处理装置
存储装置

图 31

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/111068

A. CLASSIFICATION OF SUBJECT MATTER

G06F 8/41(2018.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

SIPOABS; DWPI; CNABS; CNTXT; IEEE; CNKI: LEARNING, MODEL, COMPILE, MACHINE, CODE, EXECUTE, GENERATE, RETRIEVAL, SEARCH, CACHE, BINARY, 学习, 模型, 编译, 机器, 代码, 执行, 生成, 检索, 搜索, 缓存, 二进制

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2018349109 A1 (APPLE INC.) 06 December 2018 (2018-12-06) claims 1-20, description, paragraphs [0021]-[0044] and [0050]-[0054], and figures 2 and 3	1-50
Y	CN 102187313 A (MICROSOFT CORPORATION) 14 September 2011 (2011-09-14) claims 1-10, and description, pages 3 and 4	1-50
Y	CN 108345937 A (GOOGLE INC.) 31 July 2018 (2018-07-31) claims 1-18, and description, pages 4-9	12-20, 35-43
A	CN 105556464 A (INTEL CORPORATION) 04 May 2016 (2016-05-04) entire document	1-50

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 November 2020	**01 December 2020**

Name and mailing address of the ISA/CN	Authorized officer
China National Intellectual Property Administration (ISA/CN) **No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088** **China**	
Facsimile No. **(86-10)62019451**	Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/111068

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2018349109 A1	06 December 2018	US 10606566 B2	31 March 2020
		CN 110603520 A	20 December 2019
		WO 2018222290 A1	06 December 2018
CN 102187313 A	14 September 2011	EP 2340481 B1	13 June 2018
		JP 2012506094 A	08 March 2012
		JP 5518085 B2	11 June 2014
		BR PI0919132 A2	08 December 2015
		CN 102187313 B	09 July 2014
		WO 2010045027 A3	01 July 2010
		RU 2520344 C2	20 June 2014
		WO 2010045027 A2	22 April 2010
		EP 2340481 A2	06 July 2011
		US 2010095284 A1	15 April 2010
		EP 2340481 A4	06 March 2013
		US 9250938 B2	02 February 2016
		RU 2011114807 A	20 October 2012
CN 108345937 A	31 July 2018	DE 102018100239 A1	12 July 2018
		CN 108345937 B	12 November 2019
		SG 10201800155 P A	30 August 2018
		US 9798527 B1	24 October 2017
		GB 2560410 A	12 September 2018
		GB 201800179 D0	21 February 2018
		WO 2018129327 A1	12 July 2018
		HK 1259126 A1	22 November 2019
CN 105556464 A	04 May 2016	EP 3060984 A4	31 May 2017
		US 10268497 B2	23 April 2019
		US 2015212836 A1	30 July 2015
		EP 3060984 A1	31 August 2016
		WO 2015060850 A1	30 April 2015

国际检索报告

国际申请号
PCT/CN2020/111068

A. 主题的分类

G06F 8/41(2018.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

SIPOABS; DWPI; CNABS; CNTXT; IEEE; CNKI: LEARNING, MODEL, COMPILE, MACHINE, CODE, EXECUTE, GENERATE, RETRIEVAL, SEARCH, CACHE, BINARY, 学习、模型、编译、机器、代码、执行、生成、检索、搜索、缓存、二进制

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
Y	US 2018349109 A1 (APPLE INC) 2018年 12月 6日 (2018-12-06) 权利要求1-20、说明书第21-44，50-54段、附图2-3	1-50
Y	CN 102187313 A (微软公司) 2011年 9月 14日 (2011-09-14) 权利要求1-10、说明书第3-4页	1-50
Y	CN 108345937 A (谷歌有限责任公司) 2018年 7月 31日 (2018-07-31) 权利要求1-18，说明书第4-9页	12-20，35-43
A	CN 105556464 A (英特尔公司) 2016年 5月 4日 (2016-05-04) 全文	1-50

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2020年 11月 18日	2020年 12月 1日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 刘欢 电话号码 86-010-62411658

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2020/111068

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
US	2018349109	A1	2018年 12月 6日	US	10606566	B2	2020年 3月 31日
				CN	110603520	A	2019年 12月 20日
				WO	2018222290	A1	2018年 12月 6日
CN	102187313	A	2011年 9月 14日	EP	2340481	B1	2018年 6月 13日
				JP	2012506094	A	2012年 3月 8日
				JP	5518085	B2	2014年 6月 11日
				BR	PI0919132	A2	2015年 12月 8日
				CN	102187313	B	2014年 7月 9日
				WO	2010045027	A3	2010年 7月 1日
				RU	2520344	C2	2014年 6月 20日
				WO	2010045027	A2	2010年 4月 22日
				EP	2340481	A2	2011年 7月 6日
				US	2010095284	A1	2010年 4月 15日
				EP	2340481	A4	2013年 3月 6日
				US	9250938	B2	2016年 2月 2日
				RU	2011114807	A	2012年 10月 20日
CN	108345937	A	2018年 7月 31日	DE	102018100239	A1	2018年 7月 12日
				CN	108345937	B	2019年 11月 12日
				SG	10201800155P	A	2018年 8月 30日
				US	9798527	B1	2017年 10月 24日
				GB	2560410	A	2018年 9月 12日
				GB	201800179	D0	2018年 2月 21日
				WO	2018129327	A1	2018年 7月 12日
				HK	1259126	A1	2019年 11月 22日
CN	105556464	A	2016年 5月 4日	EP	3060984	A4	2017年 5月 31日
				US	10268497	B2	2019年 4月 23日
				US	2015212836	A1	2015年 7月 30日
				EP	3060984	A1	2016年 8月 31日
				WO	2015060850	A1	2015年 4月 30日