(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 4 月 1 日 (01.04.2021)

WIPO | PCT

(10) 国际公布号
WO 2021/057722 A1

(51) 国际专利分类号:
G06N 3/063 (2006.01)

(21) 国际申请号: PCT/CN2020/116820

(22) 国际申请日: 2020 年 9 月 22 日 (22.09.2020)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201910910116.1 2019年9月24日 (24.09.2019) 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: 张潇 (**ZHANG, Xiao**); 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)。 周玉松 (**ZHOU, Yusong**); 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)。 孟小甫 (**MENG, Xiaofu**); 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: 广州三环专利商标代理有限公司 (**SCIHEAD IP LAW FIRM**); 中国广东省广州市越秀区先烈中路80号汇华商贸大厦1508室, Guangdong 510070 (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, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU,

(54) **Title:** METHOD OF PERFORMING SPLITTING IN NEURAL NETWORK MODEL BY MEANS OF MULTI-CORE PROCESSOR, AND RELATED PRODUCT

(54) 发明名称: 用多核处理器实现神经网络模型拆分方法及相关产品

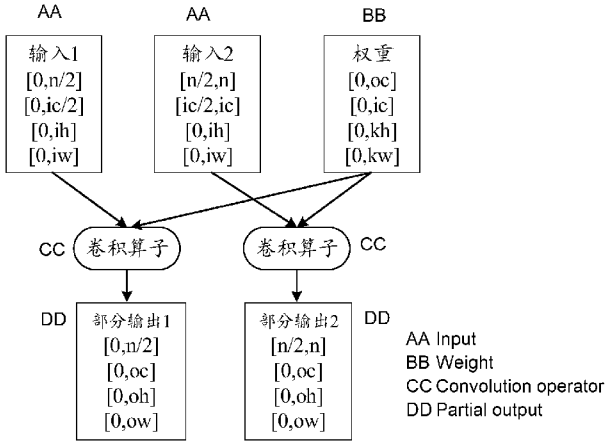

图 5

(57) **Abstract:** Embodiments of the present application disclose a method of performing splitting in a neural network model by means of a multi-core processor, and a related product. The method comprises: when a splittable operator is present in a neural network model, splitting the operator, and selecting the optimal split combination to obtain the optimal splitting result of the entire neural network model; and executing a sub-operator corresponding to the optimal splitting result by means of multi-core parallel processing. The invention thus reduces consumption of computer equipment resources.

(57) 摘要: 本申请实施例公开了一种用多核处理器实现神经网络模型拆分方法及相关产品，当神经网络模型中存在可以进行拆分的算子时，对算子进行拆分，并选择最优拆分组合获得整个神经网络模型的最优拆分结果，再由多核并行执行最优拆分结果对应的子算子，达成减少计算机设备的资源消耗的目的。

WO 2021/057722 A1

[见续页]

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))。

用多核处理器实现神经网络模型拆分方法及相关产品

技术领域

本发明涉及深度学习技术领域，尤其涉及一种用多核处理器实现神经网络模型拆分方法及相关产品。

背景技术

近年来，神经网络处理器被不断提出，并如同通用处理器一样，正在由单核向多核扩展。这种扩展后的多核结构可以在训练阶段支持数据并行的方式来提高数据吞吐量，加快训练速度。然而，在推理阶段，相比吞吐量深度神经网络对端到端的时延有着更高的要求，这往往决定了加速器在某个场景下的可用性。传统的数据并行方案不能满足推理场景下对加速器小数据、低延迟的要求。

发明内容

为实现上述目的，第一方面，本申请实施例提供了一种用多核处理器实现神经网络模型拆分方法，该方法包括：

根据所述神经网络模型中目标算子的算子，确定与所述目标算子的算子关联的张量数据的原始拆分状态集合；其中，所述目标算子为所述神经网络模型中的至少一层；

在所述目标算子的算子与所述原始拆分状态集合之间插入胶水算子，调整所述算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据；

遍历所述调整后的拆分状态集合，确定相邻拆分状态集合之间所述算子的张量数据的拆分路径；

根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径；

根据所述目标拆分方式对所述神经网络模型的目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

第二方面，本申请实施例提供了一种用多核处理器实现神经网络模型拆分装置，该装置可以包括：

第一确定单元，用于根据所述神经网络模型中目标算子的算子，确定与所述目标算子的算子关联的张量数据的原始拆分状态集合；其中，所述目标算子为所述神经网络模型中的至少一层；

调整单元，用于在所述目标算子的算子与所述原始拆分状态集合之间插入胶水算子，调整所述算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据；

遍历单元，用于遍历所述调整后的拆分状态集合，确定相邻拆分状态集合之间所述算子的张量数据的拆分路径；

第二确定单元，用于根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径；

拆分单元，用于根据所述目标拆分方式对所述神经网络模型的目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

第三方面，本申请实施例提供了一种芯片，所述芯片包括第二方面提供的神经网络模型处理装置。

第四方面，本申请实施例提供了一种计算机设备，所述计算机设备包括第三方面提供的芯片或第二方面提供的神经网络模型处理装置。

第五方面，本申请实施例提供了一种计算机设备，包括处理器和存储器，所述处理器和存储器相互连接，其中，所述处理器包括通用处理器和人工智能处理器，所述存储器用于存储支持计算机设备执行上述方法的计算机程序，所述计算机程序包括程序指令，所述处理器被配置用于调用所述程序指令，执行上述第一方面的方法。

第六方面，本申请实施例提供了一种计算机可读存储介质，所述计算机存储介质存储有计算机程序，所述计算机程序包括程序指令，所述程序指令当被处理器执行时使所述处理器执行上述第一方面的方法。

第七方面，本申请实施例提供了一种计算机程序产品，其中，上述计算机程序产品包括存储了计算机程序的非瞬时性计算机可读存储介质，上述计算机程序可操作来使计算机执行如本申请实施例第一方面所述的方法中所描述的部分或全部步骤。该计算机程序产品可以为一个软件安装包。

实施本申请实施例，计算机设备通过对神经网络模型中算子关联的张量数据进行拆分，获得张量数据对应的原始拆分状态集合，再通过胶水算子对算子关联的张量数据的拆分状态集合进行调整，获得调整后的拆分状态集合，然后目标算子关联的张量数据的拆分状态集合，确定相邻拆分状态集合之间目标算子的张量数据的目标拆分路径，最后根据目标拆分路径对目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。这样一方面通过拆分算子对应的张量数据实现拆分算子，可以使得在多核上并行执行拆分算子的情况下，避免了对每个算子原有的指令实现的修改和重构，在这个过程中，为了减少神经网络模型中互相连接的层之间，因为算子特性不同造成的张量数据拆分方式不同带来的相互制约，加入了胶水算子对张量数据的拆分方式进行调整，以便增加神经网络处理器调用神经网络模型时的可执行性。另一方面，通过对算子关联的张量数据进行拆分达到减小算子运算数据规模的目的，再根据张量数据拆分状态对应的拆分路径选择，进一步优化张量数据的拆分方式。最后拆分获得的张量数据分配至多核处理器上，使得多核处理器中的每个核的硬件资源都能有效利用，该方案能有效降低各种神经网络模型在多核处理器上的端到端时延。

附图说明

为了更清楚地说明本申请实施例中的技术方案，下面将对实施例描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图 1A 为人工智能处理器的软件栈示意图；

图 1B 为本申请实施例提供的一种基于计算图的算子拆分示意图；

图 1C~图 1H 为本申请实施例提供的卷积算子在并行度为 2 的情况下的对应拆分方式示意图；

图 2 为本申请实施例提供的一种计算机设备的结构示意图；

图 3 为本申请实施例提供的一种用多核处理器实现神经网络模型拆分方法的流程示意图；

图 4 为本申请实施例提供的一种神经网络模型中的算子连接关系示意图；

图 5 为本申请实施例提供的一种卷积算子拆分示意图；

图 6 为本申请实施例提供的一种拆分状态集合示意图；

图 7 为本申请实施例提供的一种 Transform 的调整过程示意图；

图 8A 为本申请实施例提供的 Concat 算子语义的示意图；

图 8B 为本申请实施例提供的 Split 算子语义的示意图；

图 9 为本申请实施例提供的一种插入 Transform 获得拆分状态的示意图；

图 10 为本申请实施例提供的一种拆分状态之间的拆分路径示意图；

图 11 为本申请实施例提供的一种深度残差网络中的残差块示意图；

图 12 为本申请实施例提供的一种用多核处理器实现神经网络模型拆分装置的结构示意图。

具体实施方式

下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行描述。

应当理解，本披露的权利要求、说明书及附图中的术语“第一”、“第二”和“第三”等是用于区别不同对象，而不是用于描述特定顺序。本披露的说明书和权利要求书中使用的术语“包括”和“包含”指示所描述特征、整体、步骤、操作、元素和/或组件的存在，但并不排除一个或多个其它特征、整体、步骤、操作、元素、组件和/或其集合的存在或添加。

还应当理解，在此本披露说明书中所使用的术语仅仅是出于描述特定实施例的目的，而并不意在限定本披露。如在本披露说明书和权利要求书中所使用的那样，除非上下文清楚地指明其它情况，否则单数形式的“一”、“一个”及“该”意在包括复数形式。还应当进一步理解 ,在本披露说明书和权利要求书中使用的术语“和/ 或”是指相关联列出的项中的一个或多个的任何组合以及所有可能组合，并且包括这些组合。

如在本说明书和权利要求书中所使用的那样，术语“如果”可以依据上下文被解释为“当... 时”或“一旦”或“响应于确定”或“响应于检测到”。类似地，短语“如果确定”或“如果检测到[所描述条件或事件]”可以依据上下文被解释为意指“一旦确定”或“响应于确定”或“一旦检测到[所描述条件或事件]”或“响应于检测到[所描述条件或事件]”。

为了便于更好的理解本申请所描述的技术方案，下面先解释本申请实施例所涉及的技术术语：

（1）张量（tensor）

在本技术方案中，张量仅仅是对存储的一块数据的特征描述，张量记录了数据的形状、类型等信息。

本申请实施例中，张量应该理解为张量数据，可以包括神经网络模型中输入张量数据、输出张量数据，也可以包括特征张量数据等。

以人工智能深度学习框架 TensorFlow 为例，一般使用阶（rank），形状（shape）和维数（dimension number）来描述张量的维度，其关系可以表示为如表 1 所示：

表 1

阶	形状	维数	例子
0	[]	0-D	4
1	[D1]	1-D	[2]
2	[D1,D2]	2-D	[6,2]
3	[D1,D2,D3]	3-D	[7,3,2]
n	[D1,D2,D3,...,Dn]	n-D	形为 [D1,D2,D3,...,Dn] 的张量

如表 1 所示，张量 A=4，其表示一个数。张量 A=[6,2]，其表示二维矩阵，具体地，该矩阵为 6 行 2 列的矩阵。

（2）数据并行

具体来说，所谓数据并行是指把数据划分成若干块分别映像到不同的处理器上，每一个处理器运行同样的处理程序对所分派的数据进行处理。现有中，大部分并行处理均采用

这种处理方式，尤其是对于计算复杂性很高的问题，如流体力学计算、图象处理等。

在本申请实施例中，数据并行可以应用于大规模的神经网络并行训练中。具体来说，数据并行的核心是使用多个处理器同时进行对于同一个神经网络模型的训练。在训练的每一轮迭代中，每个处理器从数据集中获取本轮迭代使用的数据，在每个处理器上完成一轮整个网络的推理及训练计算，并返回本轮计算得到的梯度数据来进行模型的更新。维护权值的服务器在收到所有处理器的梯度之后，使用这些梯度进行模型数据的更新。显然，由于多个处理器会并行地执行训练任务，其等价于在每轮迭代中一个更大批量的数据能够被处理，也就加快了系统完成这个训练任务所需要的时间。所以，数据并行的关键在于每一轮迭代中待处理数据的批量的大小，批量越大，尽可能划分到越多的处理器来并行处理。

（3）模型并行

在本申请实施例中，模型并行是数据并行之外的另一种神经网络并行计算方式。简单来说，模型并行是通过划分神经网络模型参数的方式把计算负载分配到不同的处理器上。

（4）多核处理器

当前多核处理器采用的最普遍的结构是基于存储共享的多核结构，处理器中包含了多个计算核，每个计算核上有独立的缓存，寄存器堆，计算单元以及指令控制单元，所有的计算核共享同一全局存储。

现有中，单个核已经足够完成任何复杂逻辑的计算任务，但其性能受限于摩尔定律和芯片工艺。为了进一步提升处理器的性能，多个计算核被引入处理器中，它们可以被用于处理那些有着较高并行度的计算任务。

在实际应用中，共享存储多核结构是一种经典的多核结构，并且非常适合数据并行的神经网络训练方法。每个核可以作为数据并行中的一个处理器，分别读取不同的数据，然后并行完成网络模型的正反向计算。每个核在计算阶段仍能够保持其在之前单核架构下良好的性能功耗比，与此同时，整个系统的吞吐量也可以随着核数的扩展而增加。

（5）算子拆分

在本申请实施例中，我们采用算子拆分的方式来实现计算任务的拆分，即把单个算子拆分成多个可以并行执行的子算子。需要说明的是，这里，拆分前的原始算子和拆分后的若干个子算子都是人工智能处理器所支持的算子，原始的张量数据随着算子的拆分也被拆分成若干个新的子张量数据。反映到计算图上，则是把原来的包含单个算子的计算图细化成了一张包含更多可并行执行的算子的计算图。通过这一实现方式，可以实现类似于模型并行的算子内任务拆分，同时又保证了拆分后的每个子算子都可以复用单核架构下算子的指令实现来进行计算，避免了对原有算子的指令实现的重构。

在本申请实施例中，算子拆分不完全局限于对模型参数的拆分，也会采用数据并行的方式对数据进行拆分，这种方法实际上模糊了模型并行和数据并行的界限。以卷积算子为例，如果把卷积算子的输入数据和权值作为计算图中等同低位的张量数据，那么，数据并行时基于对输入数据的划分来分割计算，而模型并行时基于权值的划分来分割计算，这二者都是通过划分卷积算子相关联的张量数据来实现对计算负载的划分。从这个角度来说，数据并行和模型并行是统一的。

（6）人工智能处理器

人工智能处理器，也称之为专用处理器，在本申请实施例中，人工智能处理器是指针对特定应用或者领域的处理器。例如：图形处理器（GPU，Graphics Processing Unit），又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备（如平板电脑、智能手机等）上进行图像运算工作的专用处理器。又例如：神经网络处理器（NPU，Neural Processing Unit），是一种在人工智能领域的应用中针对矩阵乘法运算的专用处理器，采用“数据驱动并行计算”的架构，特别擅长处理视频、图像类的海量

多媒体数据。

（7）人工智能处理器的软件栈

人工智能处理器的软件栈：参见图 1A，该软件栈结构 10 包括人工智能应用 100、人工智能框架 102、人工智能学习库 104、人工智能运行时库 106 以及驱动 108。接下来对其进行具体阐述。

人工智能应用 100 对应不同的应用场景，提供对应的人工智能算法模型。该算法模型可以直接被人工智能框架 102 的编程接口解析，在其中一个可能的实现方式中，通过人工智能学习库 104 将人工智能算法模型转换为二进制指令，调用人工智能运行时库 106 将二进制指令转换为人工智能学习任务，将该人工智能学习任务放在任务队列中，由驱动 108 调度任务队列中的人工智能学习任务让底层的人工智能处理器执行。在其中另一个可能的实现方式中，也可以直接调用人工智能运行时库 106，运行先前已固化生成的离线运行文件，减少软件架构的中间开销，提高运行效率。

人工智能框架是整个深度学习生态体系中的第一层。早期在 Caffe 中，Layer 被当做是构建神经网络的基本元素，而在之后的人工智能框架，例如 TensorFlow、MXNet 中，虽然采用了不同的称呼，例如 Operator，但与 Caffe 的 layer 在核心思想上依旧是相似的，都是将神经网络计算进一步拆分为各类常见的面向张量数据的算子，人工智能框架需要将神经网络映射的计算图结构所表达的深度学习任务具体化成可以在 CPU 或者人工智能处理器执行的指令和数据。在这个过程中，人工智能框架采用算子作为落实计算任务的具体元素，为每个算子都提供了在 CPU 或者人工智能处理器上执行的核函数（Kernel），根据计算图，人工智能框架调度执行计算图中每个算子对应的核函数，完成整个神经网络的计算。

为了便于更好的理解本申请，下面具体阐述本申请所描述的技术方案的研究思路：

现有技术中，数据并行的问题在于，其扩展性依赖于处理的数据批量的大小。尽管在训练阶段这通常不会是一个问题，但是对于推理阶段这个前提则难以保证。一般来说，用于实时服务领域（包括视频监控，自动驾驶等）的神经网络模型，处理的数据通常是以流的方式串行输入，导致了每次处理的数据规模很小甚至往往是单张图片。在这种情况下，数据并行不能提供任何并行度，所有的工作任务会集中在单个核上，这使得多核带来的计算资源不能转化成处理任务的速度。

当在线下使用数据集完成了神经网络模型的训练后，就会把模型部署到云端的服务器上来处理外界发来的数据，此时的应用场景就由离线训练变成了在线推理。在在线推理阶段，一个非常重要的指标是时延，也就是从服务器收到待处理数据到返回处理后的结果的时间，进一步来说，是使用神经网络模型处理数据的时间。低时延保证云端服务器能够对客户端发来的数据在最短的时间内做出响应，在一些更加敏感的场景下，直接决定了方案是否可用。因此，在线推理阶段对于人工智能处理器的要求就由处理大批量数据、高吞吐量转变为处理小批量数据、低时延。

在这种情况下，传统的数据并行或者模型并行难以有效降低推理任务的时延。对于数据并行来说，大批量数据是前提，这本身与在线推理小批量数据的特点矛盾。对于模型并行来说，它通常是为了解决一个规模很大的神经网络模型超过了单个设备的内存限制而采用的方法，把算子分配到不同的核上并不能降低网络的时延。为了真正能够在多核人工智能处理器上降低推理任务的时延，必须寻找一种方法，能够把对小批量数据甚至单个数据的推理计算任务合理地分配到多核架构的各个核上，保证每一时刻都有尽可能多的核参与计算，才能充分利用多核架构的资源。一种方法是把神经网络中的每个算子的计算任务都拆分到多个核上计算，这种方法即使在处理单张图片的推理任务时也能保证每一时刻都有多个核参与计算，从而达到了利用多核资源降低时延的目的。

但是，对于多核人工智能处理器来说，还有很多要解决的问题。首先，深度学习人工

智能处理器通过定制化自身的硬件设计来适配深度学习算法本身的数据并行特征，提高计算吞吐量，人工智能处理器往往需要足够的数据规模才能达到较高的计算效率，而算子内的进一步拆分会减小每个核上的计算规模。当拆分达到一定粒度，每个核上计算效率的损失会超过拆分增加并行度所带来的收益。因此，必须在拆分并行和计算效率之间，在保证足够计算效率的同时提供足够的并行度。

另一方面，神经网络模型可以看作是一个由通常数以百计甚至千记的算子所构成的复杂计算图。不同种类的算子内的算法逻辑各不相同，这就导致对这些算子进行拆分的方法也不一样。每个算子的拆分，除了平衡自身的计算效率和并行度，还要考虑和前后算子的搭配，甚至于对全局的影响。深度学习的快速发展带来的是越来越多的大规模复杂网络，通过手动方式寻找一种好的并行方法是不现实的，因此需要一种自动化的方法来保证来对于不同的网络都能够给出一种较好的拆分并行策略。

此外，还需要考虑的是对于底层人工智能处理器的可移植性。对于没有足够良好的可编程性的人工智能处理器来说，由单核扩展到多核，并且实现算子内部的拆分并行所带来的修改软件栈的工作量是非常大的。传统的数据并行和模型并行的实现仍然是基于一个处理核完成一个算子的计算任务，所以并不会带来很多额外的工作，而单个算子的跨核并行需要对算子本身实现进行修改，这种修改的难易程度依赖于人工智能处理器的可编程性和原有算子实现逻辑的复杂程度。如何减小在多核架构上实现低时延推理过程中的额外开销，缓解实现过程中工作量对于人工智能处理器本身可编程性的依赖，使得方法能够在未来对于不同的多核人工智能处理器都有一定的通用性也是一个需要考虑的问题。

基于上述描述，我们采用算子拆分的方式来实现计算任务的拆分，即把单个算子拆分成多个可以并行执行的子算子。拆分前的原始算子和拆分后的若干子算子都是深度学习处理器所支持的元算子，原始张量数据随着算子的拆分也拆分成了若干个新的子张量数据。如图 1B 所示，图 1B 为本申请实施例提供的一种基于计算图的算子拆分示意图，如图 1B 所示，算子 Op0 由在单核 Core0 上执行，经过算子拆分，转换成在 Core0、Core1、Core2 和 Core3 多个核上并行执行。

算子的拆分就隐含了如何对该算子所关联的张量数据进行拆分的信息，算子关联的张量数据包括算子的输入张量数据和输出张量数据，例如在图 1B 中，算子 Op0 经过拆分成为 Opo0_0、Opo0_1、Opo0_2 和 Opo0_3 三个子算子 ,那么算子 Op0 的输入张量数据 Tensor1 也被对应拆分为 Tensor1_0、Tensor1_1、Tensor1_2 和 Tensor1_3。通过这种方法实现了算子内任务拆分，同时又保证了拆分后的每个子算子都可以复用单核架构下算子的指令实现来进行计算，避免了对原有算子的指令实现的重构，即子算子 Opo0_0、Opo0_1、Opo0_2 和 Opo0_3 的算子指令实现与 Op0 的算子指令实现相同。

图 1C~图 1H 给出了计算图上卷积算子在并行度为 2 的条件下的多种拆分方式。图 1C~图 1H 中每个张量数据给出了各个维度的起点和终点，用来明确拆分后的子张量数据与原始张量数据之间的关系。图中 n、ic、ih、iw、oc、oh、ow、kh、kw 依次表示输入张量数据批量大小、输入张量数据特征图像数量、输入张量数据特征图像的长度、输入张量数据特征图像的宽度、输出张量数据特征图像数量、输出张量数据特征图像的长度、输出张量数据特征图像的宽度、卷积核窗口长度、卷积核窗口宽度。其中，图 1C 是原始计算图；图 1D 按照输入张量数据的 N 维度拆分；图 1E 按照输出张量数据的 C 维度拆分；图 1F 按照输入张量数据 C 维度拆分；图 1G 按照输入张量数据的 H 维度拆分；图 1H 按照输入张量数据的 W 维度拆分。这些拆分方式执行在不同的维度上，同时彼此之间可以通过互相组合形成更多新的拆分方式，从而提供最够的并行度来利用多核资源，同时在一定程度上可以避免单个维度上的过度拆分影响计算效率。

参见图 2，为本申请实施例提供的一种计算机设备的结构示意图。如图 2 所示，计算

机设备 20 可以包括通用处理器 201、存储器 202、通信总线 203、通信接口 204 和至少一个人工智能处理器 205，通用处理器 201、人工智能处理器 205 通过所述通信总线连接所述存储器 202 和所述通信接口 203。

通用处理器 201 可以是中央处理单元(Central Processing Unit ,CPU) ,该通用处理器 201 还可以是其他通用处理器、数字信号处理器 (Digital Signal Processor，DSP)、专用集成电路 (Application Specific Integrated Circuit，ASIC)、现成可编程门阵列 (Field-Programmable Gate Array，FPGA) 或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。通用处理器 201 可以是微处理器或者该通用处理器 201 也可以是任何常规的处理器等。

通用处理器 201 还可以是一种集成电路芯片，具有信号的处理能力。在实现过程中，本申请的算子拆分方法的各个步骤可以通过通用处理器 201 中的硬件的集成逻辑电路或者软件形式的指令完成。

存储器 202 可以是只读存储器(Read-Only Memory ,ROM)、随机存取存储器(Random Access Memory，RAM) 或其他存储器。本申请实施例中，存储器 202 用于存储数据以及各种软件程序，例如本申请实施例中根据胶水算子的位置关系对神经网络模型进行优化的程序等。

可选的，在本申请实施例中，所述存储器可以包括用于存储信息的物理装置，通常是将信息数字化后再以利用电、磁或者光学等方法的媒体加以存储。本实施方式所述的存储器又可以包括：利用电能方式存储信息的装置，如 RAM、ROM 等；利用磁能方式存储信息的装置，如硬盘、软盘、磁带、磁芯存储器、磁泡存储器、U 盘；利用光学方式存储信息的装置，如 CD 或 DVD。当然，还有其他方式的存储器，例如量子存储器、石墨烯存储器等等。

通信接口 204 使用例如但不限于收发器一类的收发装置 ,来实现计算机设备 20 与其他设备或通信网络之间的通信。例如，可以通过通信接口 204 接收其他设备发送的模型文件。

人工智能处理器 205 可以作为协处理器挂载到主 CPU (Host CPU) 上，由主 CPU 为其分配任务。在实际应用中，人工智能处理器 205 可以实现一种或多种运算。例如，以神经网络处理器 (Network Processing Unit，NPU) NPU 为例，NPU 的核心部分为运算电路，通过控制器控制运算电路提取存储器 202 中的矩阵数据并进行乘加运算。

可选的，人工智能处理器 205 可以包括 8 个集群 (cluster) ，每个 cluster 中包括 4 个人工智能处理器核。

可选的，人工智能处理器 205 可以是可重构体系结构的人工智能处理器。这里，可重构体系结构是指，如果某一人工智能处理器能够利用可重用的硬件资源，根据不同的应用需求，灵活的改变自身的体系结构，以便为每个特定的应用需求提供与之相匹配的体系结构，那么这一人工智能处理器就称为可重构的计算系统，其体系结构称为可重构的体系结构。

应当理解，计算机设备 20 仅为本申请实施例提供的一个例子，并且，计算机设备 20 可具有比示出的部件更多或更少的部件，可以组合两个或更多个部件，或者可具有部件的不同配置实现。

下面结合图 3 所示的本申请实施例提供的一种用多核处理器实现神经网络模型拆分方法的流程示意图，具体说明在本申请实施例中是如何通过对目标算子的输入张量数据的拆分来表征对目标算子的拆分方式，进而达到优化处理器核运算过程的目的，下面以 caffe 为例参考附图详细描述。可以包括但不限于如下步骤。

步骤 301、根据所述神经网络模型对应计算图中的目标算子，确定与所述目标算子相关联的张量数据的原始拆分状态集合。

在 caffe 框架下，所述目标算子可以是神经网络模型中的对应目标层（layer），该目标层为所述神经网络模型中的至少一层，所述张量数据包括输入张量数据和输出张量数据。

在本申请实施例中，“神经网络模型”也称模型，如“第一神经网络模型”、“第二神经网络模型”或“第三神经网络模型”，可以接收输入数据，并根据接收的输入数据和当前的模型参数生成预测输出。在实际应用中，预测输出可以包括图像检测输出结果、语义分析输出结果、图像分类输出结果等等。该神经网络模型可以包括深度学习神经网络模型(deep neural network，DNN）、卷积神经网络模型（Convolutional Neural Network, CNN）、极限学习机模型（extreme learning machine，ELM）或其他的神经网络模型等。

在 caffe 框架下，神经网络模型具有层级结构，如图 4 所示，图 4 为本申请实施例提供的一种神经网络模型中的算子连接关系示意图，神经网络模型中可以包括卷积层 Conv、激活函数层 Relu、池化层 Pool、分类器 Softmax 和全连接层，每一层对应至少一个算子。算子本身的拆分方式具有唯一的与之关联的张量数据的拆分方式与之对应。请参阅图 5，图 5 为本申请实施例提供的一种卷积算子拆分示意图，如图 5 所示，卷积算子 Conv 的输入张量数据包括输入规模 Input 和权值 Weight，对于输入规模 Input{[0,n), [0,ic) [0,ih), [0,iw)}这一输入张量数据，其中 n 表示输入数据批量大小，ic 表示输入数据特征图像数据量，ih 表示输入数据特征图像的长度，iw 表示输入数据特征图像的宽度，可以进行 n 维度上的拆分，获得输入子张量数据 Input1{[0,n/2), [0,ic/2) [0,ih), [0,iw)}和 Input2{[n/2,n), [ic/2,ic) [0,ih), [0,iw)}，我们把张量数据按照任意一种方式拆分得到的所有子张量数据称为该张量数据的一种拆分状态 s，张量数据所有可能的状态构成了该张量数据的状态空间 S。假设神经网络模型中有算子 Op 按照某一拆分方式进行拆分，则其输入数据 Tensor0 和输出数据 Tensor1 分别有状态 s 和 t，二者分属于 Tensor0 的拆分状态空间 S 和 Tensor1 的拆分状态空间 T。在此基础上，Op 自身的拆分方式可以视作是一条由 s 指向 t 的有向边。例如(Input1, Input2)即为一种拆分状态 s。假设图 5 中的卷积算子 Conv 的输出子张量数据为 Partial1{[0,n/2), [0,oc) [0,oh), [0,ow)}和 Partial2{[n/2,n), [0,oc) [0,oh), [0,ow)}，(Partial1, Partial2)为一种拆分状态 t，那么 Conv 算子自身的拆分方式可以视作是一条由 s 指向 t 的有向边。

理论上与算子关联的张量数据可以按照任意一种算子能够执行的方式进行拆分，但是在实际的神经网络模型中，张量数据往往与多个算子存在关联关系，请参阅图 6，图 6 为本申请实施例提供的一种拆分状态集合示意图，如图 6 所示，张量数据 Tensor1 既是算子 Op0 的输出张量数据，也是 Op1 的输入张量数据。当 Op0 确定按照某种方式进行拆分后，Tensor1 作为 Op0 的输出，同样也确定了按照某种方式拆分成了一系列子张量数据，那么 Op1 在选择拆分方式时 ,必须保证选择的方式与其输入张量数据 Tensor1 已经确定的拆分方式兼容，这使得 Op1 的选择范围受到了约束。继续推而广之，Op1 在这种约束下所选择的拆分方式又会通过与其关联的张量数据约束其他相邻的算子的拆分选择。

在一个可选的实施例中，神经网络模型对应计算图中的目标算子的输入张量数据的原始拆分状态集合中的拆分状态根据目标算子的运算逻辑和对应输出张量数据的拆分状态集合中的拆分状态确定。

在本申请实施例中，考虑到不同的算子具有不同的特性，为了避免不合理的拆分方式带来的负面影响，在对算子进行拆分时，计算机设备可以根据算子的类型确定算子的拆分方式，从而可以得到原始拆分状态集合中的状态。

算子的拆分方式主要包括以下几种：（1）算子支持在任意维度上进行拆分；（2）算子支持在有限维度上拆分；（3）算子不支持拆分。例如 ReLU 算子，Conv 算子，所支持的拆分方式允许其输入数据在 NCHW (输入数据批量大小、特征图像的个数、特征图像的长和特征图像的宽)中的任意维度上进行拆分；有些算子，譬如 Softmax 算子，所支持的拆分方式只允许其输入数据在某几个特定的维度上进行拆分；而最后一些算子，往往是实现

上非常复杂的算子，譬如非极大值抑制（Non-Maximum Suppression，NMS）算子，难以通过算子拆分的方式把计算负载分配到多个核上并行，因此这类算子最终只能在单个核上执行，相应的其输入数据应该保持完整不拆分的状态。那么多层算子之间的拆分方式互相影响结果包括：（1）完全支持；（2）部分支持；（3）不支持。如果相互连接的两个算子都支持在任意维度上进行拆分，那么两个算子之前的拆分方式互相之间完全支持，可以按照任意维度拆分获得两个算子对应的张量数据的拆分状态集合。如果互相连接的两个算子，其中一个支持任意维度上的拆分，另一个不支持拆分，或者只支持有限维度上的拆分，那么两个算子的拆分方式互相之间部分支持，需要对两个算子的张量数据可能的状态拆分集合求交集，获得最终的算子对应的状态拆分集合。或者，如果互相连接的两个算子其中一个支持有限维度的拆分，另一个不支持拆分，或者两个都不支持拆分，那么两个算子的拆分方式互相之间不支持，两个算子的张量数据不能进行拆分，对应的拆分状态集合中的拆分状态只有原张量数据对应的拆分状态。

对于算子来说，在已经确定其对应的输出张量数据的拆分状态集合中的拆分状态的情况下，可以根据算子的运算逻辑和输出张量数据的拆分状态集合中的拆分状态确定输入张量数据的拆分状态集合中的拆分状态。例如图 6 中的算子 Op0，其输入张量数据的拆分状态集合 T0 中的拆分状态可以根据 Op0 的运算逻辑和其对应的输出张量数据的拆分状态集合 T1 中的拆分状态确定。假设 Op1 是一个只能在有限维度上拆分的算子，且已经确定了 T1 对应的有限个拆分状态，而 Op0 是一个可以在任意维度上拆分的算子，T0 中的拆分状态根据 T1 和 Op0 的所有拆分状态求交集获得。

在一个可选的实施例中，神经网络模型对应计算图的目标算子的输出张量数据的原始拆分状态集合中的拆分状态根据算子的运算逻辑和对应输入张量数据的拆分状态集合中的拆分状态确定。

同样的，在已经确定算子的输入张量数据的拆分状态集合中的拆分状态的情况下，可以根据算子的运算逻辑和输入张量数据的拆分状态集合中的拆分状态确定输出张量数据的拆分状态集合中的拆分状态。例如图 6 中的算子 Op1，在已经确定其输入张量数据的拆分状态集合 T1 中的拆分状态时，可以根据 Op1 的运算逻辑和 T1 中的拆分状态确定 T2 中的拆分状态。假设 Op1 是一个只能在有限维度上拆分的算子，且已经确定了 T1 对应的有限个拆分状态，那么 T2 中为与 T1 中的拆分状态对应的有限维度上拆分获得的拆分状态。

与目标算子关联的张量数据包括输入张量数据和输出张量数据，这两种张量数据拆分获得的子张量数据形成的拆分状态集合为原始拆分状态集合。

步骤 302、在所述目标算子与所述原始拆分状态集合之间插入胶水算子，调整所述目标算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据。

在得到算子关联的张量数据对应的原始拆分状态集合方面，因为原始拆分状态集合可以根据算子自身的运算逻辑获得，或者根据算子的运算逻辑和输入张量数据的原始拆分状态集合中的拆分状态获得，又或者根据算子的运算逻辑和输出张量数据的原始拆分状态集合中的拆分状态获得，无论是哪一种获得方式，都与算子自身的运算逻辑密切相关。但是，神经网络模型中层级连接的算子需要共用同一个原始拆分状态集合，例如图 6 中的原始拆分状态集合 T1 既是算子 Op0 的输出张量数据对应的原始拆分状态集合，也是算子 Op1 的输入张量数据对应的原始拆分状态集合，但是算子 Op0 和算子 Op1 的运算逻辑不同，支持的拆分方式也不同，那么在根据 Op1 的运算逻辑确定 Tensor1 的拆分方式后，Op1 再选择拆分方式时，必须保证选择的方式与其输入数据 Tensor1 已经确定的拆分方式相兼容，这使得 Op1 的选择范围受到了约束。那么，可以理解的是，Op1 在这种约束下所选择的拆分方

式又会通过与其关联的张量数据约束其他相邻算子的拆分选择。

这种算子间关于拆分方式选择的相互影响会带来很多问题，首先，会带来性能方面的问题。在实际应用中，当计算机设备在多核处理器上调用不同的拆分方式下对应子计算任务时，性能间有差异。那么，可以理解的是，如果相邻的两算子最佳的算子拆分方案在其共同关联的张量数据上的拆分方式上不一致的情况下，为了避免冲突，必然有一方要屈就于另一方的选择。

其次，算子之间的拆分方式的相互影响会影响整个网络的可执行性。如前述内容提到的，有些算子是实现上非常复杂的算子，譬如 NMS（Non-maximum suppression）算子，难以通过算子拆分的方式把计算负载分配到多个核上并行，这类算子最终只能在单个核上执行，相应的输入数据应该保持完整不拆分的状态。那么，可以理解的是，如果一个神经网络模型中存在这一类算子，则必须保证该算子的输入数据处于完整不拆分的状态，否则网络在该算子处无法继续执行。如果这种约束随着网络结构扩散，就会使得难以通过算子拆分的方式在神经网络计算中挖掘出足够数量的并行度。

为了解决算子拆分彼此之间相互影响的问题，在目标算子的算子与关联的原始拆分状态集合之间插入胶水算子，该胶水算子可以实现让神经网络模型对应的计算图中的每个算子可以灵活不受限地选择作用于自身的拆分方式。

胶水算子插入位置为根据输入张量数据获得的原始拆分状态集合和算子之间，原始拆分状态集合是根据算子自身的运算逻辑、数据规模，以及现有的相关联的拆分状态集合获得的，在原始拆分状态集合之后插入胶水算子，获得输入张量数据对应的第三拆分状态集合，第三拆分状态集合中可能包括原始拆分状态集合中的拆分状态，以及一些算子支持的新的拆分状态，将第三拆分状态集合和原始拆分状态集合进行合并，保证新的拆分状态集合中包括所有原始拆分状态集合和第三拆分状态集合中的拆分状态，获得输入张量数据对应的调整后的拆分状态集合，再使用调整后的拆分状态集合进行算子运算，可以使算子直接调用新的拆分状态集合，而不用耗费多余时间执行原始拆分状态集合。

具体来说，本申请实施例引入胶水算子 Transform，用于将张量数据按照一种方式拆分得到的若干个子张量数据调整成按照另一种方式拆分得到的若干个子张量数据。若当前张量数据的拆分方式不被其后的算子的任何一种拆分方式所允许，又或者其后的算子在兼容当前张量数据的拆分方式的前提下可选择的方案所带来的性能提升很差，可以通过在计算图中插入 Transform 把当前数据调整至另一种更优的拆分方式，如图 7 所示，图 7 为 Transform 的调整过程示意图，如图 7 所示，原本按照第一种方式拆分得到的子张量数据 Tensor_0 和 Tensor_1 经过 Transform 调整后，获得按照第二种方式拆分得到的新的子张量数据，新的子张量数据能够被算子 Op_0 和 Op_1 兼容。

Transform 的语义可以包括神经网络中常见的算子 Concat 和 Split 两方面的含义。下面对其进行具体阐述。

Concat 算子，也即，拼接算子，用于将多个张量数据沿着指定的维度拼接成一个张量。除了在指定维度外，输入张量的其他维度应该保持一致。通过 Concat 算子，神经网络将代表来自上游不同位置的特征的多个张量拼接成一个，从而可以在下游计算中对这些特征共同进行处理。具体地，可以参见图 8A 所示的 Concat 算子语义的示意图。

Split 算子，也即拆分算子，用于将一个张量在指定维度上拆分成多个张量。拆分后的多个张量除了指定维度之外，在其他维度上保持一致。通过 Split 算子，可以把属于同一张量数据的特征拆成多份，从而在后续计算中分别进行针对性处理。具体地，可以参见图 8B 所示的 Split 算子语义的示意图。

Transform 可以使用拼接-拆分两阶段的方式来实现。在拼接阶段，可以使用 Concat 算子把在任意维度上相邻的子张量数据拼接成一个新的子张量数据，在拆分阶段，可以把

Split 算子把任意一个子张量数据拆分成几个更小的子张量数据。这样，张量数据按照任意一种方式拆分得到的子张量数据可以转换成按照任意另一种方式拆分得到的子张量数据。

在一个可选的实施例中，调整算子的输入张量数据的拆分状态集合中的拆分状态，包括：对原始拆分状态集合中的拆分状态进行拼接。

请参阅图 9，图 9 为本申请实施例提供的一种插入 Transform 获得拆分状态的示意图，如图 9 所示，假设张量数据 Tensor1 对应的原始拆分状态集合中的原拆分状态 state1 包括两个子张量数据 Input1{[0,n/2), [0,ic/2) [0,ih), [0,iw)}和 Input2{[n/2,n), [ic/2,ic) [0,ih), [0,iw)}，对其进行拼接后，获得的张量数据为 Input{[0,n), [0,ic) [0,ih), [0,iw)}，对应拆分状态 state2，即是一次通过 Transform 完成对拆分状态进行调整的过程。

在一个可选的实施例中，调整目标算子的张量数据的拆分状态集合中的拆分状态，包括：对原始拆分状态集合中的的拆分状态进行拆分。

具体地，假设张量数据 Tensor1 对应的原始拆分状态集合中的拆分状态 state1 包括两个子张量数据 Input1{[0,n/2), [0,ic/2) [0,ih), [0,iw)}和 Input2{[n/2,n), [ic/2,ic) [0,ih), [0,iw)}。对其进行拆分后，获得的子张量数据为 Input3{[0,n/4), [0,ic/4) [0,ih), [0,iw)}，Input4{[n/4,n/2), [ic/4,ic/2) [0,ih), [0,iw)}，Input5{[n/2,3n/4), [ic/2,3ic/4) [0,ih), [0,iw)}，Input6{[3n/4,n), [3ic/4,ic) [0,ih), [0,iw)}，对应拆分状态 state3，即是一次通过 Transform 完成对拆分状态进行调整的过程。

在一个可选的实施例中，调整算子的张量数据的拆分状态集合中的拆分状态，包括：对原始拆分状态集合中张量数据的拆分状态进行拼接，再对经过拼接处理后的拆分状态集合中的拆分状态进行拆分。

对于目标算子来说，需要获得的调整后的拆分状态集合中的新的拆分状态可能与原始拆分状态集合中的原始拆分状态在不同维度上的拆分粒度不同，例如在输入数据批量大小 n 维度上，新的拆分状态需要更大的粒度，在输入数据特征图像的长度 ih 维度上，新的拆分状态需要更小的粒度，那么在通过 Transform 进行调整时，需要对原始拆分状态进行子张量数据的先拼接，再拆分，获得新的拆分状态，完成 Transform 对拆分状态的调整过程。

在一个可选的实施例中，调整目标算子的拆分状态集合中的拆分状态，包括：对原始拆分状态集合中的拆分状态进行拆分，再对经过拆分处理后的拆分状态集合中的拆分状态进行拼接。

同样的，对于上述调整后的拆分状态集合中的新的拆分状态与原始拆分状态集合中的原始拆分状态在不同维度上的拆分粒度不同的问题，可以对原始拆状态对应的子张量数据的先拆分和再拼接。例如在 n 维度上，新的拆分状态需要更小的粒度，在 ih 维度上，新的拆分状态需要更大的粒度，对原始拆分状态进行子张量数据先拆分，再拼接，获得新的拆分状态，完成 Transform 对拆分状态的调整过程。

可见，在本申请实施例中，通过胶水算子 Transform 对原始拆分状态集合中的原始拆分状态对应的子张量数据进行拆分、合并，或者拆分后再合并，或者合并后再拆分，以完成将原始拆分状态集合中的原始拆分状态转换为调整后的拆分状态集合中的新的拆分状态，使得调整后的拆分状态集合能够更好地被算子兼容，降低提升人工智能处理器调用神经网络模型时的不可执行概率。

步骤 303、遍历所述目标算子关联的张量数据的所述拆分状态集合，确定相邻拆分状态集合之间所述目标算子的张量数据的拆分路径。

在一个神经网络模型对应的计算图中，包括多个目标层对应的目标算子，可以为所有目标算子与原始拆分状态集合之间都插入 Transform 算子，也可以为部分目标算子与所述原始拆分状态集合之间插入 Transform 算子。如果都插入了 Transform 算子，目标算子的张量数据的拆分状态全部由调整后的拆分状态集合确定；如果部分插入了 Transform 算子，那么

目标算子的张量数据的拆分状态由部分原始拆分状态集合和部分调整后的拆分状态集合确定。

如果部分目标算子与原始拆分状态集合之间插入了 Transform 算子，另一部分目标算子与原始拆分状态集合之间没插入 Transform 算子，遍历插入 Transform 算子后获得的调整后的拆分状态集合和未插入 Transform 算子的原始拆分状态集合，确定相邻拆分状态集合之间的拆分路径，例如图 9 所示，原始拆分状态集合和目标算子 Op1 之间插入了 Transform 算子，得到调整后的拆分状态集合 T1'，那么需要遍历确定相邻拆分状态集合之间的拆分路径的拆分状态集合包括 T0，T1'，T2 和 T3。

如果所有目标算子与原始拆分状态集合之间都插入了 Transform 算子，遍历所有调整后的拆分状态集合，确定相邻拆分状态集合之间的拆分路径。例如图 9 中的原始拆分状态集合 T0，T1，T2，T3 分别与算子 Op0，Op1，Op2 和 Op3 之间插入了 Transform 算子，得到调整后的拆分状态集合 T0'， T1'，T2'，和 T3'，那么需要遍历确定相邻拆分状态集合之间的拆分路径的拆分状态集合包括 T0'，T1'，T2'和 T3'。

另外，除了在目标算子与对应的输入张量数据之间插入胶水算子，也可以在，目标算子与对应的输出张量数据之间插入胶水算子，更可以在目标算子与对应的输入张量数据、输出张量数据之间均插入胶水算子，此次仅仅是例举的部分情况，而不是穷举，本领域技术人员在理解本申请技术方案的精髓的情况下，可能会在本申请技术方案的基础上产生其它的变形或者变换，但只要其实现的功能以及达到的技术效果与本申请类似，那么均应当属于本申请的保护范围。

路径表示输入张量数据到输出张量数据的中间过程，拆分路径表示相邻拆分状态集合之间拆分状态到拆分状态的中间过程。请参阅图 10，图 10 为本申请实施例提供的一种拆分状态之间的拆分路径示意图，如图 10 所示，以神经网络模型对应计算图中的所有目标算子和原始拆分状态集合之间都插入了 Transform 算子为例，目标算子关联的张量数据的调整后的拆分状态集合中的拆分状态之间存在有向边，例如算子 Op1 对应的调整后的拆分状态集合 T1'和 T2'之间，包括 T1'：State1 指向 T2'：State2 的有向边，其中，有向边可以有两层含义：一表示算子与算子之间的连接关系；二表示人工智能处理器执行神经网络模型的执行顺序。对于第一种含义，是指如果拆分状态之间存在有向边，那么拆分状态对应的算子之间存在连接关系，存在连接关系的算子之间相邻且相互关联。对于第二种含义，是指如果拆分状态之间存在有向边，那么有向边指向的方向对应神经网络模型中的算子在处理器上的执行顺序。

在图 10 中，虚线框代表每个张量数据调整后的拆分状态集合，集合中包含了若干个拆分状态，这些状态来自于该张量数据的拆分状态空间。算子的输入张量数据的拆分状态集合中的拆分状态和输出张量数据的拆分状态集合中的状态之间的每条有向边表示该算子本身的一种拆分方式，使用该拆分方式下的并行执行时间作为有向边的权重。Tensor0'是整个网络的输入张量，Tensor3'是整个网络的输出张量数据，任意一条由 Tensor0'的状态集合中的任一状态出发，到 Tensor3'的状态集合中的任一状态结束的路径，都对应了一种该神经网络的有效拆分方案，记为 P。对于给定的神经网络模型，要搜索一个好的拆分方案，就是在图 10 中寻找一条由 Tensor0'的状态到 Tensor3'的状态的目标路径。

在本技术方案中，拆分状态与拆分状态之间的有向边具有权重，即拆分路径的权重。每条拆分路径的权重是按照该算子运算操作方式和对应的拆分后的子张量数据在神经网络多核处理器上并行执行的时间。在确定时间时，一方面要考虑算子本身的规模，另一方面要考虑包括访存带宽、运算器频率在内的多个硬件参数。神经网络模型的算子中基本上没有条件跳转，其计算量在给定算子的规模的前提下是确定的。此外，因为执行在各个核上的拆分得到的子算子的对称性，采用均分的方式来评估多核并行执行下每个核在访问全局

存储的过程中得到的访存带宽。因此，所述拆分路径的权重根据所述拆分路径对应的所述算子的运算操作类型、所述算子的张量数据经所述拆分路径获取的对应子数据的数据规模、每个处理器核的吞吐率和访存带宽确定。

在实际中，为了确保拆分路径的权重的精准性，也可以采用实际测试的方式来获得算子在各种拆分并行下的执行时间，做到这一点同样是因为算子本身的执行具有确定性。一旦我们策略并存储了某种算子在某一数据规模下按照某种方式拆分并行的实际时间，就可以将该数值用于表示所有代表该种数据规模的该种算子的该种拆分方式对应的拆分路径的权重。

人工智能处理器调用算子进行运算是会有对应的资源消耗，资源消耗的多少与算子的运算操作类型，算子的张量数据经过拆分路径获取的子数据的数据规模以及每个处理器核的吞吐率和访存带宽都有关，因此为了优化人工智能处理器的运算效率，会偏向于选择表示资源消耗更小的权重对应的有向边。

步骤 303、根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径。

在确定相邻拆分状态集合之间的算子的张量数据的拆分路径之后，只是针对单个算子的张量数据的拆分路径，对于整个神经网络模型的多层结构来说，还需要进一步获得张量数据对应的拆分路径。

在实际中，可以使用类似于 Viterbi 算法的方式从图 10 中找出最短路径。Viterbi 算法是一种动态规划算法，用于寻找最有可能产生观测时间序列的隐含状态序列。Viterbi 算法被广泛用于语音识别、关键字识别和通信解码等领域中。我们可以将张量数据拆分状态集合中的状态看作 Viterbi 算法中的隐含状态，将拆分状态之间的有向边看作是隐含状态之间的转移关系，而有向边的权重对应着隐含状态之间转移概率的对数值。

具体实现中，首先由前往后遍历网络计算图中的所有算子，当访问第 i 个算子时，已知神经网络的输入张量数据的拆分状态集合中的状态到当前算子的输入张量的拆分状态集合$\left\{s_i^0, s_i^1, ..., s_i^{p-1}\right\}$中的每个状态的最短路径$\left\{l_{s_i^0}, l_{s_i^1}, ..., l_{s_i^{p-1}}\right\}$，结合当前算子对应的所有有向边及其权重$w_{s_i^u \to s_{i+1}^v}$，可以得到由神经网络的输入张量数据的拆分状态集合中的状态到当前算子的输出张量的拆分状态集合$\left\{s_{i+1}^0, s_{i+1}^1, ..., s_{i+1}^{q-1}\right\}$中的每个状态的最短路径$\left\{l_{s_{i+1}^0}, l_{s_{i+1}^1}, ..., l_{s_{i+1}^{q-1}}\right\}$，式(1)是计算公式。当完成所有算子的遍历后，我们会得到由神经网络模型的输入张量数据的拆分状态集合中的状态到输出张量数据的拆分状态集合中的每个状态的最短路径，从这些最短路径中再次选出最短路径，即是目标的全局最短路径。最后通过回溯的方式由输出张量到输入张量确定最短路径在每个算子处选择的有向边以及每个张量数据处的拆分状态，即是我们要寻找的该计算图上的最优拆分方案。

在访问每个算子时，当前算子的输出状态集合中的状态是根据输入状态集合中的状态结合算子本身的计算语义枚举得到的。具体来说，对输入张量数据的拆分状态集合中的每个拆分状态，枚举当前算子存在哪些可能的拆分方式能够兼容当前的输入状态，这些可能的算子拆分方式所对应的输出张量数据的拆分状态将被加入输出张量数据的拆分状态集合中。有些算子也并非仅有一个输入张量数据。例如：Convolution、InnerProduction 可以有包含输入数据、权值和偏执在内至多三个输入张量，BatchNorm 和 Scale 同样可以有输入数据、均值/α 和方差/β 在内的最多三个输入张量，而图 7 中每个算子只有一个输入和一个输出。为了弥合二者之间的差异，在具体实现中我们将输入数据之外的其他输入张量的拆分状态包含在了算子的有向边之内。换句话说，虽然图 7 中每个算子只有一个输入和输出，其他的一些辅助张量被隐式地放在了有向边中。这种基于 Viterbi 算法的方式降低了搜索最

优拆分方案的复杂度。假设神经网络模型有 M 层，其中每个张量数据的拆分状态集合中至多有 N 个状态，那么每个算子最多有 N2 种不同的拆分方式。把拆分路径的比较操作作为基本操作，全遍历的情况下的时间复杂度是 O(NM)，而 Viterbi 算法下的时间复杂度是 O(MN2)；把要维护的待选拆分路径数量作为空间复杂度的评价标准，全遍历的空间复杂度是 O(NM)，Viterbi 算法是 O(N)。

$$l_{s_{i+1}^{v}} = \min_{u=0,\ldots,p-1}(l_{s_i^{u}} + w_{s_i^{u} \to s_{i+1}^{v}}) \quad (1)$$

在一个可选的实施例中，确定目标算子的张量数据的目标拆分路径，包括：遍历目标算子的算子关联的张量数据的所有拆分状态集合和所述胶水算子的张量数据拆分状态集合，对当前拆分状态集合，遍历其中的每一拆分状态，获得所有指向当前拆分状态的有向边以及有向边的起点对应的拆分状态到目标算子或胶水算子的输入张量数据的拆分状态之间的拆分路径；根据有向边的权重和有向边对应的起始拆分状态到目标算子或胶水算子的输入张量数据的拆分状态之间的拆分路径的权重确定当前拆分状态到目标算子或胶水算子的输入张量数据的拆分状态之间的拆分路径；其中，拆分路径的权重根据拆分路径对应的所有有向边的权重确定；遍历完目标算子的张量数据的所有拆分集合和胶水算子的张量数据的所有拆分状态集合后，获得目标算子的张量数据的目标拆分路径。

如果所有的目标算子与原始拆分状态集合之间都插入了胶水算子，那么得到的调整后的拆分状态集合实际是胶水算子的张量数据的拆分状态集合。因此，在整个计算图中，实际上是需要遍历胶水算子的张量数据的拆分状态集合来得到指向当前拆分状态的有向边，例如图 10 所示，目标算子为 Op2 时，Op2 关联的张量数据的拆分状态集合包括 T2'和 T3'，假设当前拆分状态集合为 T3'，遍历 T3'中的每一个拆分状态，获得指向当前拆分状态的有向边，假设当前拆分状态为 T3'：State1，那么指向当前拆分状态的有向边有两条，分别为：T2'：State1→T3'：State1，以及 T2'：State2→T3'：State1，然后获得有向边的起点对应的拆分状态到目标算子的输入张量数据的拆分状态之间的拆分路径，对于有向边 T2'：State1→T3'：State1，起点为 T0'：state1，从有向边起点到目标算子的输入张量数据的拆分状态之间的路径为：T0'：State1→T1'：State2→T2'：State1。对于另一条有向边 T2'：State2→T3'：State1，起点为 T0'：State2，从有向边起点到目标算子的输入张量数据的拆分状态之间的路径为：T0'：State2→T1'：State1→T2'：State2，这是一个正向遍历的过程。

根据拆分路径中包括的所有有向边的权重可以获得拆分路径的权重，包括对所有有向边的权重求和，求乘积，加权求和，或者求积分等。以权重求和为例，对于拆分路径 T0'：State1→T1'：State2→T2'：State2，有向边 T0'：State1→T1'：State2 的权重为ω 1，有向边 T1'：State2→T2'：State2 的权重为ω 2，拆分路径的权重可以为拆分路径中所有有向边的权重之和，即ω 11=ω 1+ω 2。

对于当前拆分状态 T3'：State1，假设有向边 T2'：State1→T3'：State1 和 T2'：State2→T3'：State1 对应的权重分别为ω 01 和ω 02，而起始拆分状态到目标算子的输入张量数据的拆分状态之间的拆分路径也有两条，其中：

T0'：State1→T1'：State2→T2'：State2，权重为ω 11；

T0'：State2→T1'：State1→T2'：State2，权重为ω 12；

那么当前拆分状态 T3'：State1 到目标算子的输入张量数据的拆分状态之间的拆分路径也包括 2 条，即为：

T0':State1→T1':State2→T2':State2→T3':State1，权重为ω 21=ω 01+ω 11；

T0':State2→T1':State1→T2':State2→T3':State1，权重为ω 22=ω 02+ω 12；

遍历完目标算子的所有调整后的拆分状态集合后，可以获得目标算子的输入张量数据

的调整后的拆分状态集合与目标算子的输出张量数据的调整后的拆分状态集合之间的目标拆分路径。目标拆分路径根据当前拆分状态到到目标算子的输入张量数据的拆分状态之间的拆分路径的权重来确定。目标拆分路径是从多条拆分路径中选取的一条，可以是总消耗时长最短的一条，或者是总占用内存最少的一条，或者是吞吐量最大的一条。对应到拆分路径中，即可以选择拆分路径的权重最大的一条，或者权重最小的一条。

例如，对目标算子 Op2 来说，已经确定了其对应的调整后的拆分状态集合中的拆分状态 T3'：State1 到目标算子的输入张量数据的拆分状态之间的拆分路径包括 2 条，且两条拆分路径对应的权重分别为ω 21 和ω 22，若权重表示算子根据输入张量数据运算获得输出张量数据的时间消耗，且ω 21 > ω 22，那么当需要选择时间消耗更少的拆分路径时，选择ω 22 对应的拆分路径。同样的，对于算子 Op2 对应的调整后的拆分状态集合中的其他拆分状态，获得其他拆分状态到目标算子的输入张量数据的拆分状态之间的拆分路径，并选择其中耗时最少的拆分路径，再从每个拆分状态对应的耗时最少的拆分路径中决策出唯一一条耗时最少的拆分路径。

假设算子 Op2 对应的唯一一条耗时最少的拆分路径为ω 22 对应的拆分路径，在这条拆分路径中，算子 Op2 的输入张量数据的调整后的拆分状态集合与输出张量数据的调整后的拆分状态集合之间的目标拆分路径可以确定为 T2':State2→T3':State1，即对算子对应的目标拆分路径的选择是根据神经网络模型全局拆分路径的权重来选取的，而不是根据单个算子的相邻拆分状态集合的拆分状态之间的有向边权重来确定的。

在一个可选的实施例中，确定目标算子的张量数据的目标拆分路径，包括：遍历目标算子的所有调整后的拆分状态集合，对当前拆分状态集合，遍历每一拆分状态，获得所有以当前拆分状态为起点的有向边以及有向边的终点对应的拆分状态到目标算子的输出张量数据的拆分状态之间的拆分路径；根据有向边的权重和有向边的终点对应的拆分状态到目标算子的输出张量数据的拆分状态之间的拆分路径的权重确定当前拆分状态到目标算子的输出张量数据的拆分状态之间的拆分路径；其中，拆分路径的权重根据拆分路径对应的所有有向边的权重确定；遍历完目标算子的所有调整后的拆分状态集合后，获得目标算子的输入张量数据的调整后的拆分状态集合与目标算子的输出张量数据的调整后的拆分状态集合之间的目标拆分路径。

对于目标算子的调整后的拆分状态集合，还可以遍历获得所有以当前拆分状态为起点的有向边，请参阅图 10，例如目标算子为 Op1 时，Op1 关联的张量数据的调整后的拆分状态集合包括 T1'和 T2'，假设当前调整后的拆分状态集合为 T1'，遍历 T1'中的每一个拆分状态，获得以当前拆分状态为起点的有向边，假设当前拆分状态为 T1'：State1，那么以当前拆分状态为起点的有向边有一条：T1': State1→T2'：State2，然后获得有向边的终点对应的拆分状态到目标算子的输出张量数据的拆分状态之间的拆分路径，对于有向边 T1': State1→T2'：State2，终点为 T3'：state1，从有向边终点到目标算子的输出张量数据的拆分状态之间的路径为：T2'：State2→T3'：State1。这是一个反向遍历的过程。

根据拆分路径中包括的所有有向边的权重可以获得拆分路径的权重，同样包括对所有有向边的权重求和，求乘积，加权求和，或者求积分等。以权重求和为例，对于拆分路径 T2'：State2→T3'：State1，其中仅包括一条有向边，那么拆分路径的权重=有向边的权重。

对于当前拆分状态 T1'：State1，假设有向边 T1': State1→T2'：State2 对应的权重为ω 31，而有向边的终点对应的拆分状态到目标算子的输出张量数据的拆分状态之间的拆分路径有一条，即为：T2'：State2→T3'：State1，权重为ω 41；那么当前拆分状态 T1'：State1 到目标算子的输出张量数据的拆分状态之间的拆分路径为：T1': State1→T2'：State2→T3'：State1，权重为：ω 51=ω 31+ω 41。

遍历完目标算子的所有调整后的拆分状态集合后，可以获得目标算子的输入张量数据

的调整后的拆分状态集合与目标算子的输出张量数据的调整后的拆分状态集合之间的目标拆分路径。

对于目标算子 Op1 来说，遍历完 T1'和 T2'中的所有拆分状态之后，获得以 T1'中的拆分状态为起点的有向边，到有向边的终点，为一条目标算子 Op1 对应的全局拆分路径，根据全局拆分路径的权重选择其中一条作为最优拆分路径。同样的，权重对应的含义包括总消耗时长，总占用内存，或吞吐量，对应到拆分路径中，可以选择权重最大的一条，或者权重最小的一条作为最优拆分路径。而从最优拆分路径中，截取出目标算子 Op1 的相邻调整后的拆分状态集合对应的有向边，即为目标算子的输入张量数据的调整后的拆分状态集合与目标算子的输出张量数据的调整后的拆分状态集合之间的目标拆分路径。

同样的，对于神经网络模型对应的计算图中只有部分算子插入了 Transform 算子的情况，遍历未插入 Transform 算子的目标算子的张量数据的拆分状态集合（原始拆分状态集合），以及插入的 Transform 算子的张量数据的拆分状态集合（调整后的拆分状态集合），得到目标算子的张量数据的目标拆分路径。

可见，在本申请实施例中，通过神经网络模型中全局张量数据的有向边组成的拆分路径的权重确定目标算子张量数据的目标拆分路径，可以在全局最优拆分方式的前提下获得目标算子张量数据的最佳拆分方式，提升了张量数据拆分准确性和适应性，进而提升了人工智能处理器调用神经网络模型的效率，整体上有效降低资源消耗。

在一个可选的实施例中，在目标算子与原始拆分状态集合之间插入胶水算子，还包括：利用包含所述胶水算子在内的计算图中的目标算子的目标拆分路径对插入的每个胶水算子进行选择，在满足目标拆分路径中包含的胶水算子的输入张量数据的状态和输出张量数据的状态相同的情况下，删除胶水算子。

在一个可选的实施例中，还包括：在满足目标拆分路径中插入了胶水算子的输入张量数据的状态和输出张量数据的状态不相同的情况下，保留胶水算子。

上述根据图 10 获得的目标拆分路径都是根据调整后的拆分状态集合确定的目标算子的张量数据的目标拆分路径，也就是说都是在插入 Transform 算子之后获得的目标拆分路径。事实上，在一些情况下，目标算子的目标拆分路径不一定是通过 Transform 调整后获得的，而是在原始拆分状态集合的情况下获得的。假设遍历目标算子 Op0，Op1，Op2，Op3 对应的原始拆分状态集合 T0-T1-T2-T3，确定目标算子 Op2 的目标拆分路径为：T2:State2→T3:State1，而图 10 中算子 Op2 根据调整后的拆分状态集合确定的目标拆分路径为 T2':State2→T3':State1，其中 T2':State2 与 T2:State2 对应的张量数据拆分方式相同，T3':State1 与 T3:State1 对应的张量数据拆分方式相同，那么 Transform 没有起到优化目标算子 Op2 对应的张量数据拆分方式的作用。但是人工智能处理器在执行插入 Transform 的神经网络模型时，会带来额外的开销，为了减少这种不必要的额外开销，当获得的目标拆分路径中，Transform 对应的输入张量数据的拆分状态和输出张量数据的拆分状态相同，即删除该目标算子插入的胶水算子 Transform，得到优化后的目标拆分路径，即根据目标算子的原始拆分状态集合得到的目标拆分路径。

对应的，如果在获得的目标拆分路径中，Transform 对应的输入张量数据的拆分状态和输出张量数据的拆分状态不相同，说明胶水算子 Transform 起到了优化目标拆分路径的作用，保留该目标算子的 Transform。

可见，在本申请实施例中，在获得目标算子的张量数据对应的目标拆分路径后，进一步确定胶水算子是否起到了优化目标拆分路径的作用，如果有，则保留该目标算子胶水算子，如果没有，则删除该目标算子的胶水算子，可以较少引入胶水算子带来的人工智能处理器额外的开销，提升神经网络模型的执行效率。

在一个可选的实施例中，该方法还包括：当前目标算子的输出张量数据被至少两个算

子作为输入张量数据，或当前目标算子具有至少两个输出张量数据时，当前目标算子的输出张量数据的拆分状态集合中保留一个拆分状态，且保留的拆分状态经由当前算子的同一有向边确定。

在一些情况下，目标算子的输出张量数据被至少两个算子作为输入张量数据，或当前目标算子具有至少两个输出张量数据，如图 11 所示，图 11 为本申请实施例提供的一种深度残差网络中的残差块示意图，如图 11 所示，前一算子的输出张量数据 X_l 被最下方的 Add 算子作为输入张量数据，同时被批量标准化算子 BN 作为输入张量数据，对于算子 Add 和算子 BN 来说，支持的拆分状态是不同的，在正向遍历时，可以获得 Add 算子和 BN 算子的输入张量数据各自的拆分状态集合，正向遍历时可以根据有向边的权重对拆分路径进行决策，选择支持 Add 算子或 BN 算子的输入张量数据对应的拆分状态集合，但是在进行回溯时，会同时得到 Add 算子对应的输入张量数据的拆分状态集合，和输出张量数据 X_l^b 对应的输入张量数据的拆分状态集合，为了保证回溯时对 X_l 的选择不发生冲突，对于 X_l 的拆分状态集合，只保留其中的一个拆分状态，且保留的拆分状态根据算子的同一有向边确定，例如可以根据前一算子的输入张量数据对应的拆分状态集合中的拆分状态指向 X_l 的拆分状态集合中的拆分状态确定。

在一个可选的实施例中，方法还包括：当前目标算子具有至少两个输入张量数据时，当前目标算子的输入张量数据的拆分状态集合中保留一个拆分状态，且拆分状态经由目标算子的同一有向边确定。

同样的，如果当前目标算子具有多个输入张量数据，那么每个张量数据都有其对应的拆分状态集合，而在进行回溯时，也可能得到目标算子的多个可选择拆分状态，为了保证算子的输入张量数据的拆分状态之间不发生冲突，保留算子输入张量数据的拆分状态集合中的一个拆分状态即可，且保留的拆分状态经由算子的同一有向边确定。

步骤 304、根据所述目标拆分路径对所述神经网络模型的目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

目标拆分路径是全局最优拆分路径中目标算子对应的拆分路径。因此神经网络模型中所有目标拆分路径组合也可以形成全局最优拆分路径，按照最优拆分路径对算子的张量数据进行拆分，进而获得算子的最优拆分方式。

在对算子的张量数据进行拆分后，在多核上调用拆分后的子张量数据，即可实现并行执行拆分后的子算子，可以提升神经网络模型的执行效率。另外，多核架构中的核数通常为 2 的整数次幂，例如 1,2,4,8,16 等，一个并行度不是 2 的整数次幂的任务往往会导致核的调度上产生 “碎片”，因此拆分后的子算子数量应该为 2 的整数次幂。算子的拆分个数可以由拆分状态中包括的子张量数据个数确定，例如图 5 中的(Input1, Input2)即为一种拆分状态，包括 2 个输入子张量数据，即将算子拆分成 2 个子算子。

可见，在本申请实施例中，根据神经网络模型对应计算图中目标算子的算子，确定与目标算子关联的张量数据的原始拆分状态集合；在目标算子与原始拆分状态集合之间插入胶水算子，调整目标算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；然后遍历目标算子关联的张量数据的拆分状态集合，确定相邻拆分状态集合之间目标算子的张量数据的拆分路径以及拆分路径的权重；再根据拆分路径的权重，确定目标算子的张量数据的目标拆分路径；最后根据目标拆分路径对神经网络模型的目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。这样一方面通过拆分算子对应

的张量数据实现拆分算子，可以使得在多核上并行执行拆分算子的情况下，避免了对每个算子原有的指令实现的修改和重构，在这个过程中，为了减少神经网络模型中互相连接的层之间，因为算子特性不同造成的张量数据拆分方式不同带来的相互制约，加入了胶水算子对张量数据的拆分方式进行调整，以便增加神经网络处理器调用神经网络模型时的可执行性。另一方面，通过对算子关联的张量数据进行拆分达到减小算子运算数据规模的目的，再根据张量数据拆分状态对应的拆分路径选择，进一步优化张量数据的拆分方式。最后拆分获得的张量数据分配至多核处理器上，使得多核处理器中的每个核的硬件资源都能有效利用，该方案能有效降低各种神经网络模型在多核处理器上的端到端时延。

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，本披露并不受所描述的动作顺序的限制，因为依据本披露，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披露所必须的。

进一步需要说明的是，虽然图 3 的流程图中的各个步骤按照箭头的指示依次显示，但是这些步骤并不是必然按照箭头指示的顺序依次执行。除非本文中有明确的说明，这些步骤的执行并没有严格的顺序限制，这些步骤可以以其它的顺序执行。而且，图 3 中的至少一部分步骤可以包括多个子步骤或者多个阶段，这些子步骤或者阶段并不必然是在同一时刻执行完成，而是可以在不同的时刻执行，这些子步骤或者阶段的执行顺序也不必然是依次进行，而是可以与其它步骤或者其它步骤的子步骤或者阶段的至少一部分轮流或者交替地执行。

上述详细阐述了本申请实施例的方法，为了便于更好地实施本申请实施例的上述方案，相应地，下面还提供用于配合实施上述方案的相关装置。

参见图 12，图 12 为本申请实施例提供的一种用多核处理器实现神经网络模型拆分装置的结构示意图，该装置 40 至少可以包括：

第一确定单元 401，用于根据所述神经网络模型对应计算图中目标算子，确定与所述目标算子相关联的张量数据的原始拆分状态集合；

调整单元 402，用于在所述目标算子的算子与所述原始拆分状态集合之间插入胶水算子，调整所述目标算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据；

遍历单元 403，用于遍历所述目标算子关联的张量数据的所述拆分状态集合，确定相邻拆分状态集合之间所述目标算子的张量数据的拆分路径；

第二确定单元 404，用于根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径；

拆分单元 405，用于根据所述目标拆分方式对所述目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

在一种可能的实现方式中，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元 402 具体用于：

对所述原始拆分状态集合中的拆分状态进行拼接。

在一种可能的实现方式中，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元 402 具体用于：

对所述原始拆分状态集合中的拆分状态进行拆分。

在一种可能的实现方式中，在调整所述算子的张量数据的拆分状态集合中的拆分状态，所述调整单元 402 具体用于：

对所述原始拆分状态集合中的拆分状态进行拼接，再对经过拼接处理后的拆分状态集合中的状态进行拆分。

在一种可能的实现方式中，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元 402 具体用于：

对所述原始拆分状态集合中的拆分状态进行拆分，再对经过拆分处理后的拆分状态集合中的状态进行拼接。

在一种可能的实现方式中，所述调整单元 402 还用于：

利用包含所述胶水算子在内的计算图中的目标算子的目标拆分路径对插入的每个胶水算子进行选择，在满足所述目标拆分路径中包含的胶水算子的输入张量数据的拆分状态和输出张量数据的拆分状态相同的情况下，删除所述胶水算子。

在一种可能的实现方式中，所述调整单元 402 还用于：

在满足所述目标拆分路径中插入了胶水算子的输入张量数据的拆分状态和输出张量数据的拆分状态不相同的情况下，保留所述胶水算子。

在一种可能的实现方式中，所述第二确定单元 404 具体用于：

遍历所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合，对当前拆分状态集合，遍历每一拆分状态，获得所有指向当前拆分状态的有向边以及所述有向边的起点对应的拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径；

根据所述有向边的权重和所述有向边对应的起始拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径的权重确定所述当前拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径；其中，所述拆分路径的权重根据所述拆分路径对应的所有有向边的权重确定；

遍历完所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合后，获得所述目标算子的张量数据的目标拆分路径。

在一种可能的实现方式中，所述第二确定单元 404 具体用于：

遍历所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合，对当前拆分状态集合，遍历每一拆分状态，获得所有以当前拆分状态为起点的有向边以及所述有向边的终点对应的拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径；

根据所述有向边的权重和所述有向边的终点对应的拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径的权重确定所述当前拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径；其中，所述拆分路径的权重根据所述拆分路径对应的所有有向边的权重确定；

遍历完所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合后，获得所述目标算子的张量数据的目标拆分路径。

在一种可能的实现方式中，所述第二确定单元 404 还用于：

当前目标算子的输出张量数据被至少两个算子作为输入张量数据，或当前目标算子具有至少两个输出张量数据时，当前目标算子的输出张量数据的拆分状态集合中保留一个拆分状态，且保留的拆分状态经由当前目标算子的同一有向边确定。

在一种可能的实现方式中，所述第二确定单元 404 还用于：

当前目标算子具有至少两个输入张量数据时，当前目标算子的输入张量数据的拆分状态集合中保留一个拆分状态，且所述拆分状态经由当前目标算子的同一有向边确定。

应该理解，上述的装置实施例仅是示意性的，本披露的装置还可通过其它的方式实现。例如，上述实施例中所述单元/模块的划分，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、模块或组件可以结合，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。

所述作为分离部件说明的单元或模块可以是物理上分开的，也可以不是物理上分开的。作为单元或模块说明的部件可以是物理单元，也可以不是物理单元，即可以位于一个装置中，或者也可以分布到多个装置上。本披露中实施例的方案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。

本申请实施例还提供一种芯片，该神经网络芯片可以为多核芯片，其中，包括中央处理单元(Central Processing Unit，CPU) 和 N 个单核神经网络处理器(Neural Network Processor，NNP)，N 为大于 1 的整数。所述 CPU 用于对所述芯片进行整体的控制和调度，为本申请实施例中的神经网络模型处理方法的执行主体。

本申请实施例还提供另一种计算机设备，该计算机设备包含上述芯片或上述神经网络模型处理装置 40。

本申请实施例还提供了一种计算机存储介质，用于存储为上述图 2 所示的计算机设备所用的计算机软件指令，其包含用于执行上述方法实施例所涉及的程序。通过执行存储的程序，可以根据神经网络模型中胶水算子的位置关系对神经网络模型进行优化，以提高神经网络模型的整体性能。当计算机设备调用优化后的神经网络模型时，由于无需执行多余的冗余操作，可以减少计算机设备的资源消耗。

由上可见，本申请实施例提供的算子拆分方法、装置、计算机设备和存储介质，该方法可以根据神经网络模型中胶水算子的位置关系对神经网络模型进行优化，以提高神经网络模型的整体性能。当计算机设备调用优化后的神经网络模型时，由于无需执行多余的冗余操作，可以减少计算机设备的资源消耗。

本领域内的技术人员应明白，本申请的实施例可提供为方法、系统、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器和光学存储器等）上实施的计算机程序产品的形式。

本申请是参照根据本申请实施例的方法、设备（系统）、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

权 利 要 求 书

1、一种用多核处理器实现神经网络模型拆分方法，其特征在于，所述方法包括：

根据所述神经网络模型对应计算图中的目标算子，确定与所述目标算子相关联的张量数据的原始拆分状态集合；

在所述目标算子与所述原始拆分状态集合之间插入胶水算子，调整所述目标算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据；

遍历所述目标算子关联的张量数据的所述拆分状态集合，确定相邻拆分状态集合之间所述目标算子的张量数据的拆分路径；

根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径；

根据所述目标拆分路径对所述目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

2、根据权利要求 1 所述的方法，其特征在于，所述调整所述目标算子的张量数据的拆分状态集合中的拆分状态，包括：

对所述原始拆分状态集合中的拆分状态进行拼接。

3、根据权利要求 1 所述的方法，其特征在于，所述调整所述目标算子的张量数据的拆分状态集合中的拆分状态，包括：对所述原始拆分状态集合中的拆分状态进行拆分。

4、根据权利要求 1 所述的方法，其特征在于，所述调整所述目标算子的拆分状态集合中的拆分状态，包括：

对所述原始拆分状态集合中的拆分状态进行拼接，再对经过拼接处理后的拆分状态集合中的拆分状态进行拆分。

5、根据权利要求 1 所述的方法，其特征在于，所述调整所述目标算子的拆分状态集合中的拆分状态，包括：

对所述原始拆分状态集合中的拆分状态进行拆分，再对经过拆分处理后的拆分状态集合中的拆分状态进行拼接。

6、根据权利要求 2-5 任一项所述的方法，其特征在于，在所述目标算子与所述原始拆分状态集合之间插入胶水算子，还包括：

利用包含所述胶水算子在内的计算图中的目标算子的目标拆分路径对插入的每个胶水算子进行选择，在满足所述目标拆分路径中包含的胶水算子的输入张量数据的拆分状态和输出张量数据的拆分状态相同的情况下，删除所述胶水算子。

7、根据权利要求 6 所述的方法，其特征在于，在所述目标算子与所述原始拆分状态集合之间插入胶水算子，所述方法还包括：

在满足所述目标拆分路径中插入了胶水算子的输入张量数据的拆分状态和输出张量数据的拆分状态不相同的情况下，保留所述胶水算子。

8、根据权利要求 1 所述的方法，其特征在于，确定所述目标算子的张量数据的目标拆分路径的步骤包括：

遍历所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合，对当前拆分状态集合，遍历每一拆分状态，获得所有指向当前拆分状态的有向边以及所述有向边的起点对应的拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径；

根据所述有向边的权重和所述有向边对应的起始拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径的权重确定所述当前拆分状态到所述目标算子或所述胶水算子的输入张量数据的拆分状态之间的拆分路径；其中，所述拆分路径的

权重根据所述拆分路径对应的所有有向边的权重确定；

遍历完所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合后，获得所述目标算子的张量数据的目标拆分路径。

9. 根据权利要求 1 所述的方法，其特征在于，确定所述目标算子的张量数据的目标拆分路径的步骤包括：

遍历所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合，对当前拆分状态集合，遍历每一拆分状态，获得所有以当前拆分状态为起点的有向边以及所述有向边的终点对应的拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径；

根据所述有向边的权重和所述有向边的终点对应的拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径的权重确定所述当前拆分状态到所述目标算子或所述胶水算子的输出张量数据的拆分状态之间的拆分路径；其中，所述拆分路径的权重根据所述拆分路径对应的所有有向边的权重确定；

遍历完所述目标算子的张量数据的所有拆分状态集合和所述胶水算子的张量数据的所有拆分状态集合后，获得所述目标算子的张量数据的目标拆分路径。

10、根据权利要求 9 所述的方法，其特征在于，所述方法还包括：

当前目标算子的输出张量数据被至少两个算子作为输入张量数据，或当前目标算子具有至少两个输出张量数据时，当前目标算子的输出张量数据的拆分状态集合中保留一个拆分状态，且保留的拆分状态经由当前目标算子的同一有向边确定。

11、根据权利要求 10 所述的方法，其特征在于，所述方法还包括：

当前目标算子具有至少两个输入张量数据时，当前目标算子的输入张量数据的拆分状态集合中保留一个拆分状态，且所述拆分状态经由当前目标算子的同一有向边确定。

12、一种用多核处理器实现神经网络模型拆分装置，其特征在于，所述装置包括：

第一确定单元，用于根据所述神经网络模型对应计算图中的目标算子，确定与所述目标算子相关联的张量数据的原始拆分状态集合；

调整单元，用于在所述目标算子与所述原始拆分状态集合之间插入胶水算子，调整所述目标算子的张量数据的拆分状态集合中的拆分状态，得到调整后的拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据；

遍历单元，用于遍历所述目标算子关联的张量数据的所述拆分状态集合，确定相邻拆分状态集合之间所述目标算子的张量数据的拆分路径；

第二确定单元，用于根据所述拆分路径的权重，确定所述目标算子的张量数据的目标拆分路径；

拆分单元，用于根据所述目标拆分方式对所述目标算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理。

13、根据权利要求 12 所述的装置，其特征在于，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元具体用于：对所述原始拆分状态集合中的拆分状态进行拼接。

14、根据权利要求 12 所述的装置，其特征在于，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元具体用于：对所述原始拆分状态集合中的拆分状态进行拆分。

15、根据权利要求 12 所述的装置，其特征在于，在调整所述算子的张量数据的拆分状态集合中的拆分状态方面，所述调整单元具体用于：

对所述原始拆分状态集合中的拆分状态进行拼接，再对经过拼接处理后的拆分状态集

合中的拆分状态进行拆分。

16、一种芯片，其特征在于，所述芯片集成如权利要求 12-15 任一项所述的用多核处理器实现神经网络模型拆分装置。

17、一种计算机设备，其特征在于，所述计算机设备包括如权利要求 16 所述的芯片或如权利要求 12-15 任一项所述的用多核处理器实现神经网络模型拆分装置。

18、一种计算机设备，其特征在于，包括处理器和存储器，所述处理器和存储器相互连接，其中，所述处理器包括通用处理器和人工智能处理器，所述存储器用于存储计算机程序，所述计算机程序包括程序指令，所述处理器被配置用于调用所述程序指令，执行如权利要求 1-11 任一项所述的方法。

19、一种计算机可读存储介质，其特征在于，所述计算机可读存储介质存储有计算机程序，所述计算机程序包括程序指令，所述程序指令当被处理器执行时使所述处理器执行如权利要求 1-11 任一项所述的方法。

20、一种计算机程序产品，其特征在于，所述计算机程序产品包括存储了计算机程序的非瞬时性计算机可读存储介质，所述计算机程序可操作来使计算机执行如权利要求 1-11 任一项所述的方法。

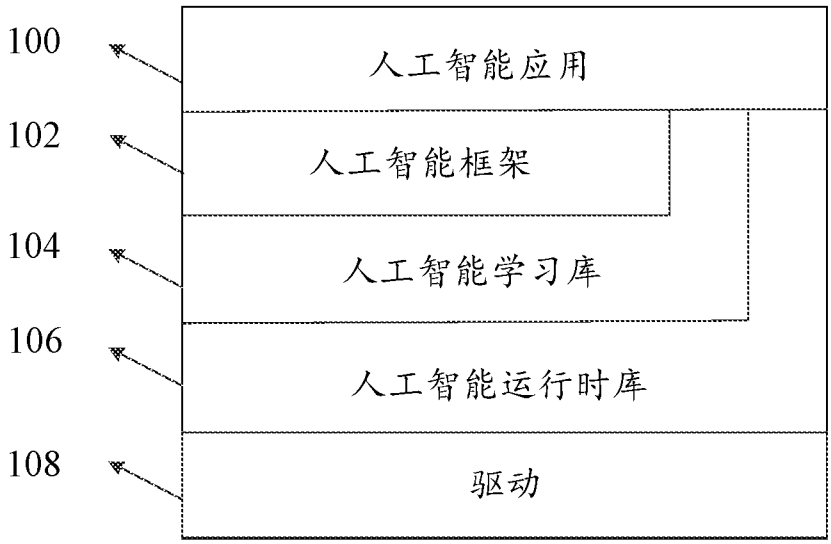
100
人工智能应用
102
人工智能框架
104
人工智能学习库
106
人工智能运行时库
108
驱动

图 1A

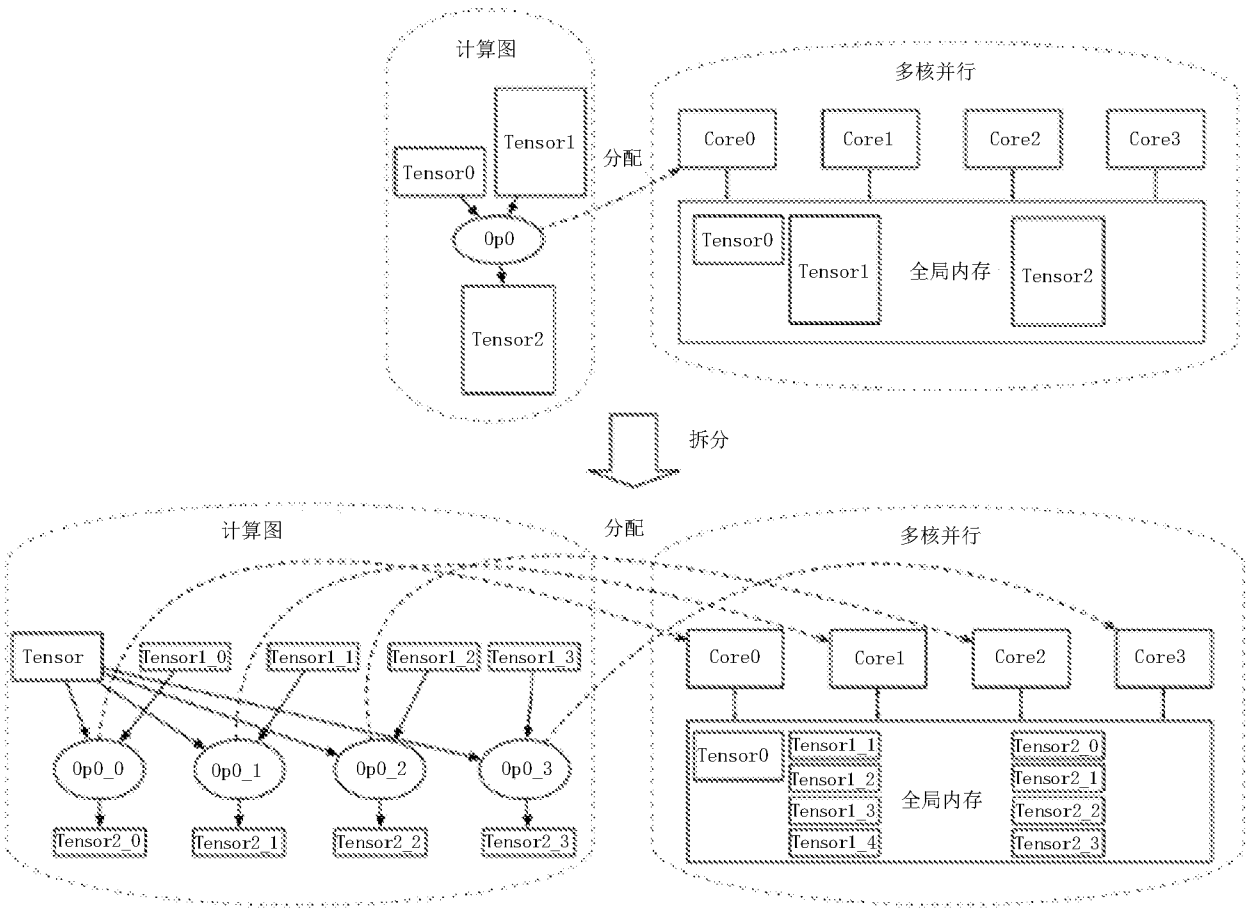
计算图
Tensor1
Tensor0
0p0
Tensor2
分配
多核并行
Core0
Core1
Core2
Core3
Tensor0
Tensor1
全局内存
Tensor2
拆分
计算图
分配
多核并行
Tensor
Tensor1_0
Tensor1_1
Tensor1_2
Tensor1_3
0p0_0
0p0_1
0p0_2
0p0_3
Tensor2_0
Tensor2_1
Tensor2_2
Tensor2_3
Core0
Core1
Core2
Core3
Tensor0
Tensor1_1
Tensor1_2
Tensor1_3
Tensor1_4
全局内存
Tensor2_0
Tensor2_1
Tensor2_2
Tensor2_3

图 1B

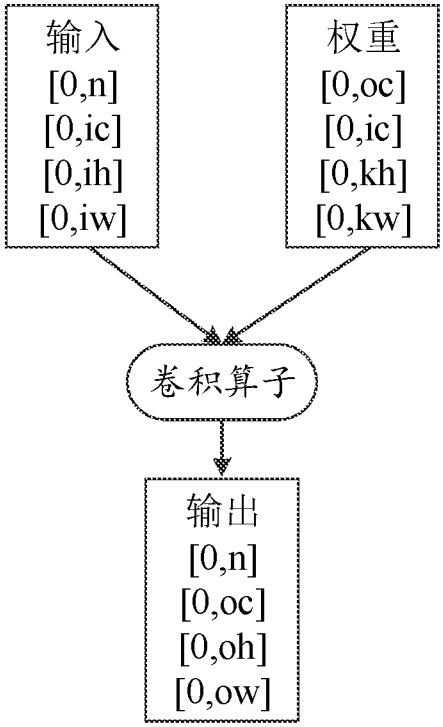
输入
[0,n]
[0,ic]
[0,ih]
[0,iw]
权重
[0,oc]
[0,ic]
[0,kh]
[0,kw]
卷积算子
输出
[0,n]
[0,oc]
[0,oh]
[0,ow]

图 1C

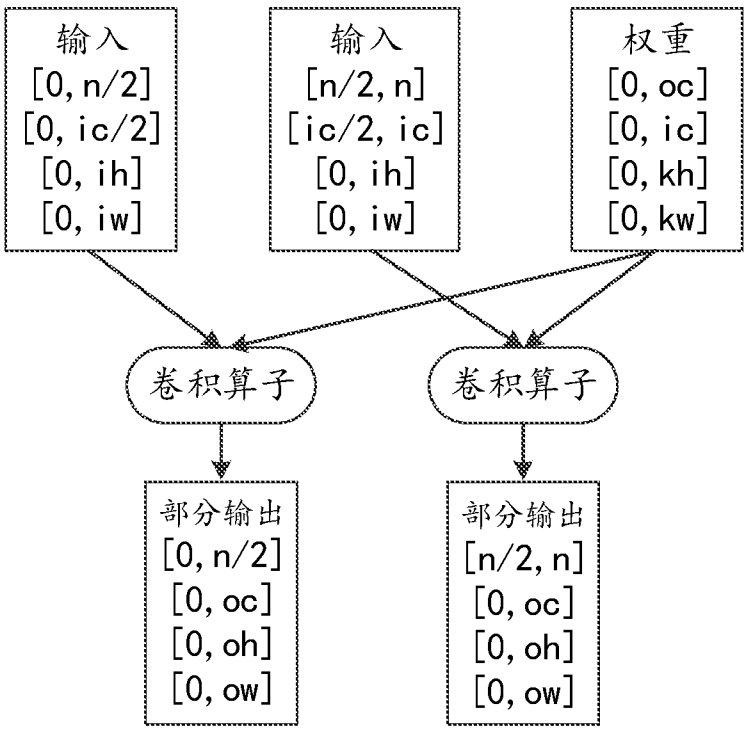
输入
[0, n/2]
[0, ic/2]
[0, ih]
[0, iw]
输入
[n/2, n]
[ic/2, ic]
[0, ih]
[0, iw]
权重
[0, oc]
[0, ic]
[0, kh]
[0, kw]
卷积算子
卷积算子
部分输出
[0, n/2]
[0, oc]
[0, oh]
[0, ow]
部分输出
[n/2, n]
[0, oc]
[0, oh]
[0, ow]

图 1D

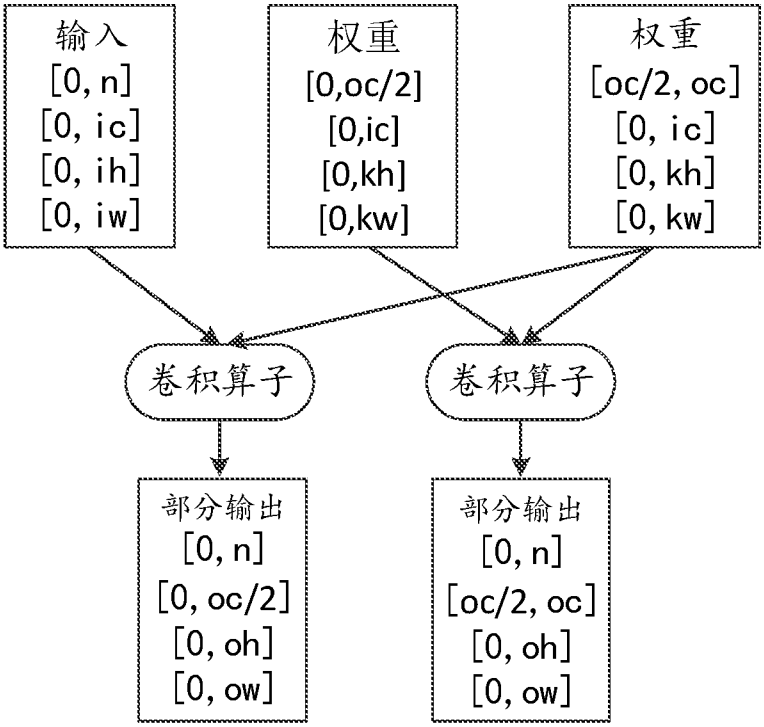
输入
[0, n]
[0, ic]
[0, ih]
[0, iw]
权重
[0,oc/2]
[0,ic]
[0,kh]
[0,kw]
权重
[oc/2, oc]
[0, ic]
[0, kh]
[0, kw]
卷积算子
卷积算子
部分输出
[0, n]
[0, oc/2]
[0, oh]
[0, ow]
部分输出
[0, n]
[oc/2, oc]
[0, oh]
[0, ow]

图 1E

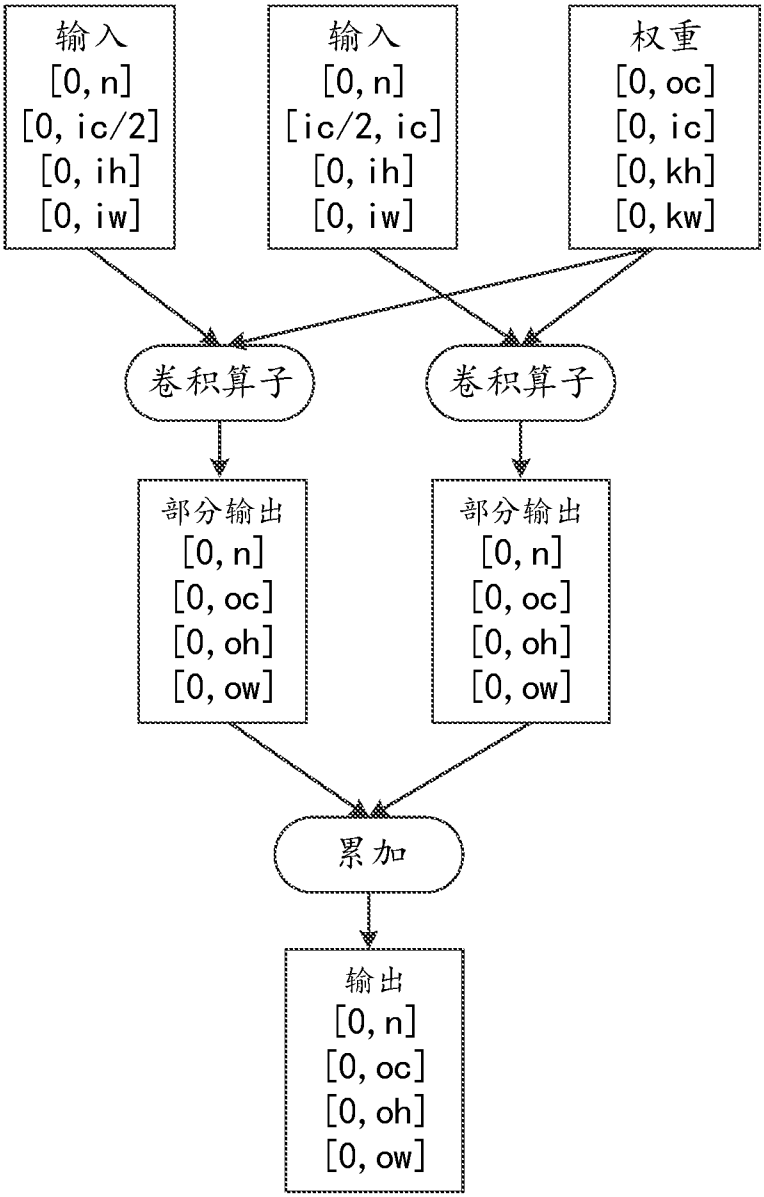
输入
[0, n]
[0, ic/2]
[0, ih]
[0, iw]
输入
[0, n]
[ic/2, ic]
[0, ih]
[0, iw]
权重
[0, oc]
[0, ic]
[0, kh]
[0, kw]
卷积算子
卷积算子
部分输出
[0, n]
[0, oc]
[0, oh]
[0, ow]
部分输出
[0, n]
[0, oc]
[0, oh]
[0, ow]
累加
输出
[0, n]
[0, oc]
[0, oh]
[0, ow]

图 1F

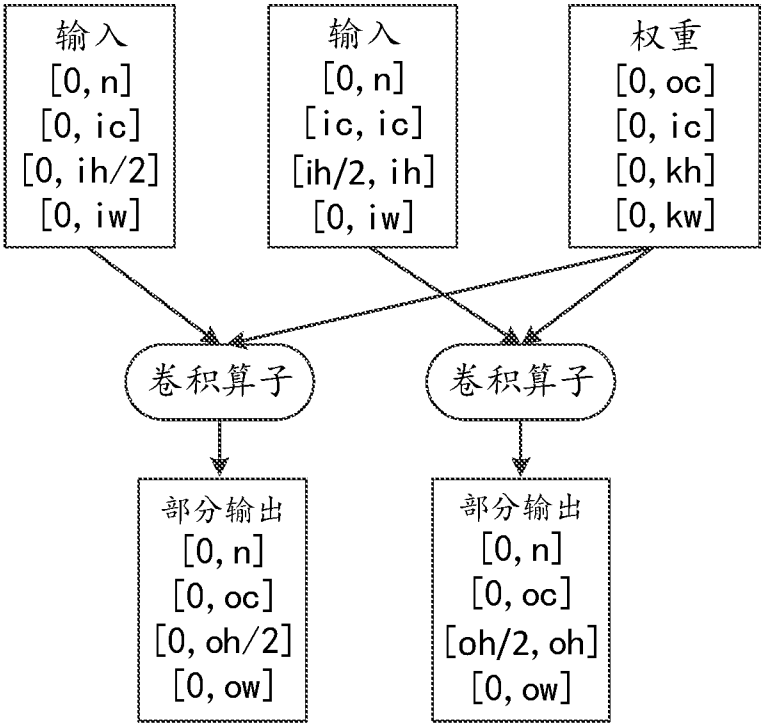
输入
[0, n]
[0, ic]
[0, ih/2]
[0, iw]
输入
[0, n]
[ic, ic]
[ih/2, ih]
[0, iw]
权重
[0, oc]
[0, ic]
[0, kh]
[0, kw]
卷积算子
卷积算子
部分输出
[0, n]
[0, oc]
[0, oh/2]
[0, ow]
部分输出
[0, n]
[0, oc]
[oh/2, oh]
[0, ow]

图 1G

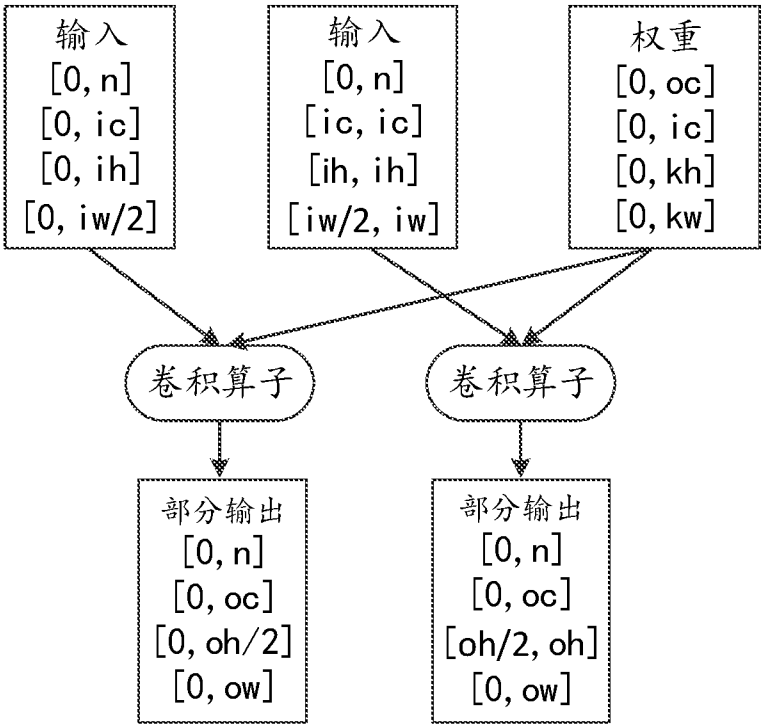
输入
[0, n]
[0, ic]
[0, ih]
[0, iw/2]
输入
[0, n]
[ic, ic]
[ih, ih]
[iw/2, iw]
权重
[0, oc]
[0, ic]
[0, kh]
[0, kw]
卷积算子
卷积算子
部分输出
[0, n]
[0, oc]
[0, oh/2]
[0, ow]
部分输出
[0, n]
[0, oc]
[oh/2, oh]
[0, ow]

图 1H

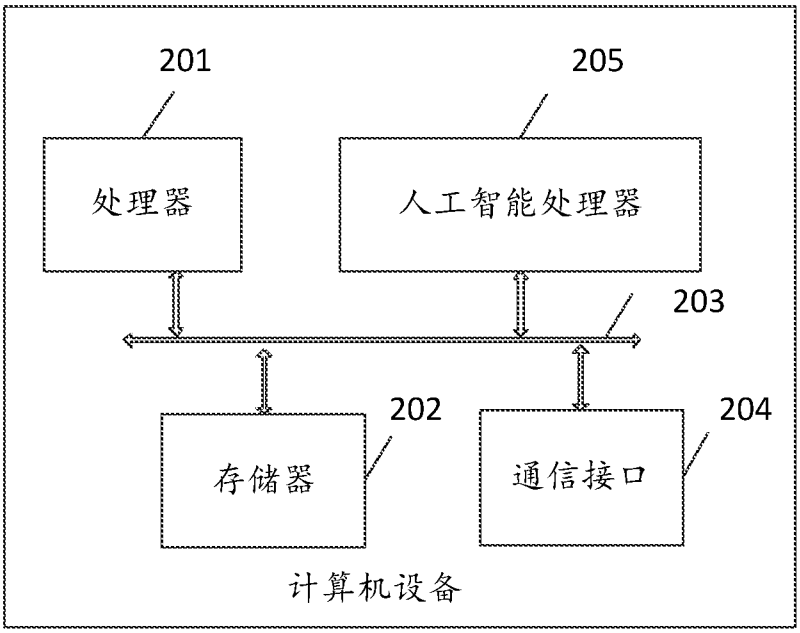
201
205
处理器
人工智能处理器
203
202
204
存储器
通信接口
计算机设备

图 2

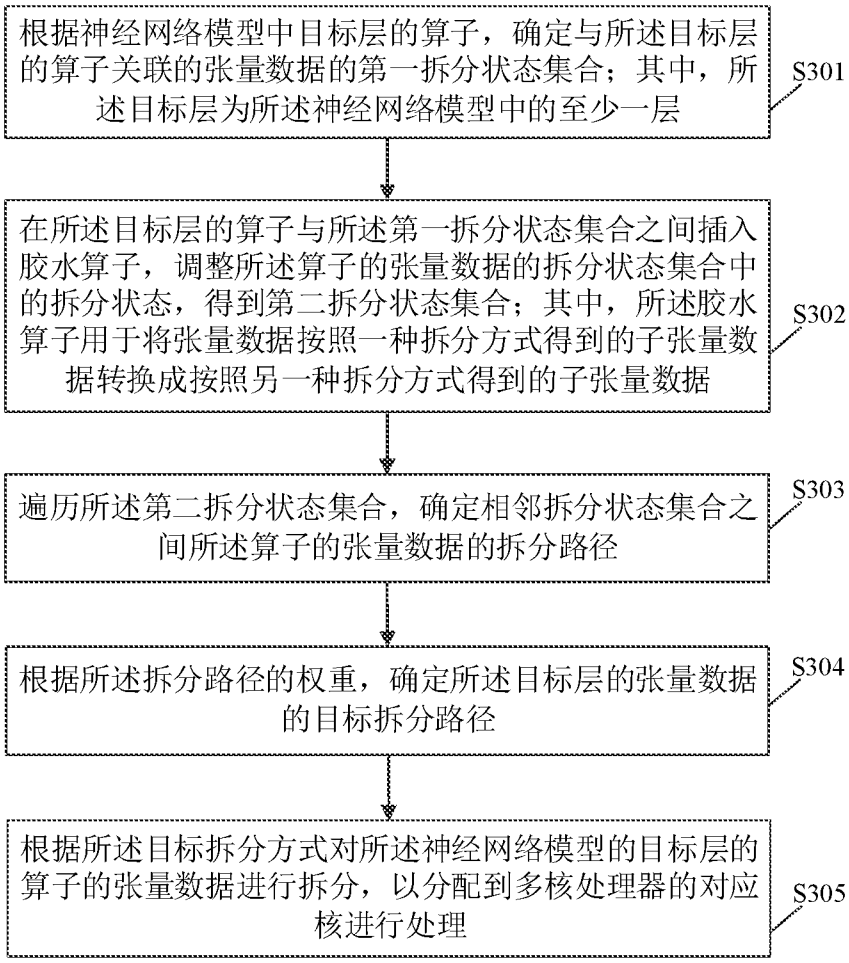
根据神经网络模型中目标层的算子，确定与所述目标层的算子关联的张量数据的第一拆分状态集合；其中，所述目标层为所述神经网络模型中的至少一层
S301
在所述目标层的算子与所述第一拆分状态集合之间插入胶水算子，调整所述算子的张量数据的拆分状态集合中的拆分状态，得到第二拆分状态集合；其中，所述胶水算子用于将张量数据按照一种拆分方式得到的子张量数据转换成按照另一种拆分方式得到的子张量数据
S302
遍历所述第二拆分状态集合，确定相邻拆分状态集合之间所述算子的张量数据的拆分路径
S303
根据所述拆分路径的权重，确定所述目标层的张量数据的目标拆分路径
S304
根据所述目标拆分方式对所述神经网络模型的目标层的算子的张量数据进行拆分，以分配到多核处理器的对应核进行处理
S305

图 3

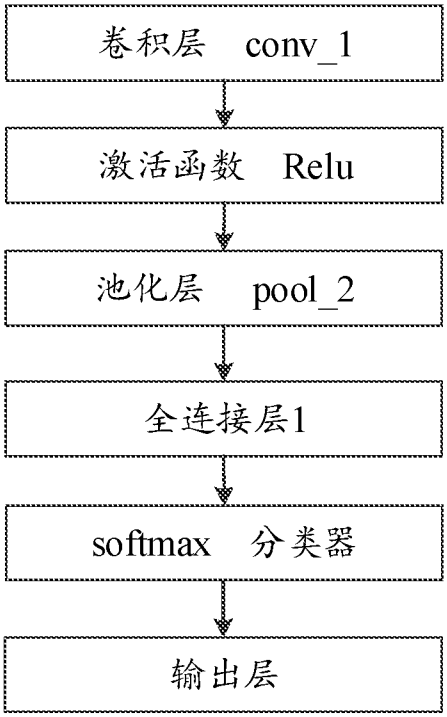

图 4

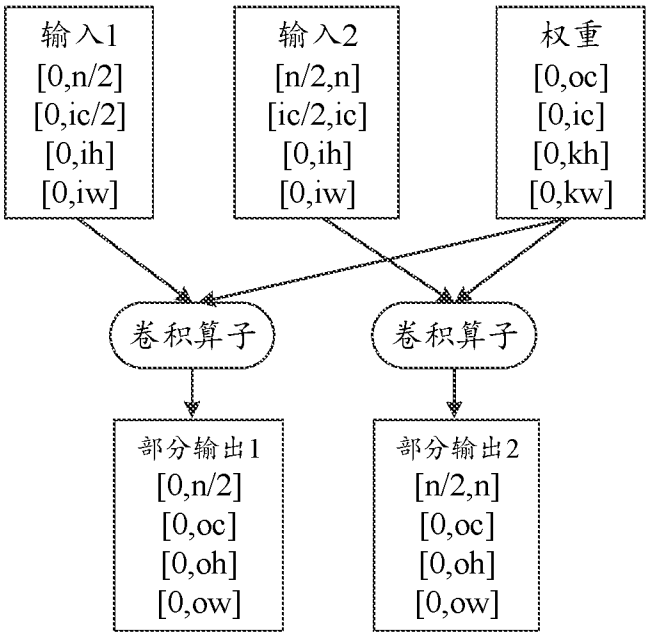
输入1
[0,n/2]
[0,ic/2]
[0,ih]
[0,iw]
输入2
[n/2,n]
[ic/2,ic]
[0,ih]
[0,iw]
权重
[0,oc]
[0,ic]
[0,kh]
[0,kw]
卷积算子
卷积算子
部分输出1
[0,n/2]
[0,oc]
[0,oh]
[0,ow]
部分输出2
[n/2,n]
[0,oc]
[0,oh]
[0,ow]

图 5

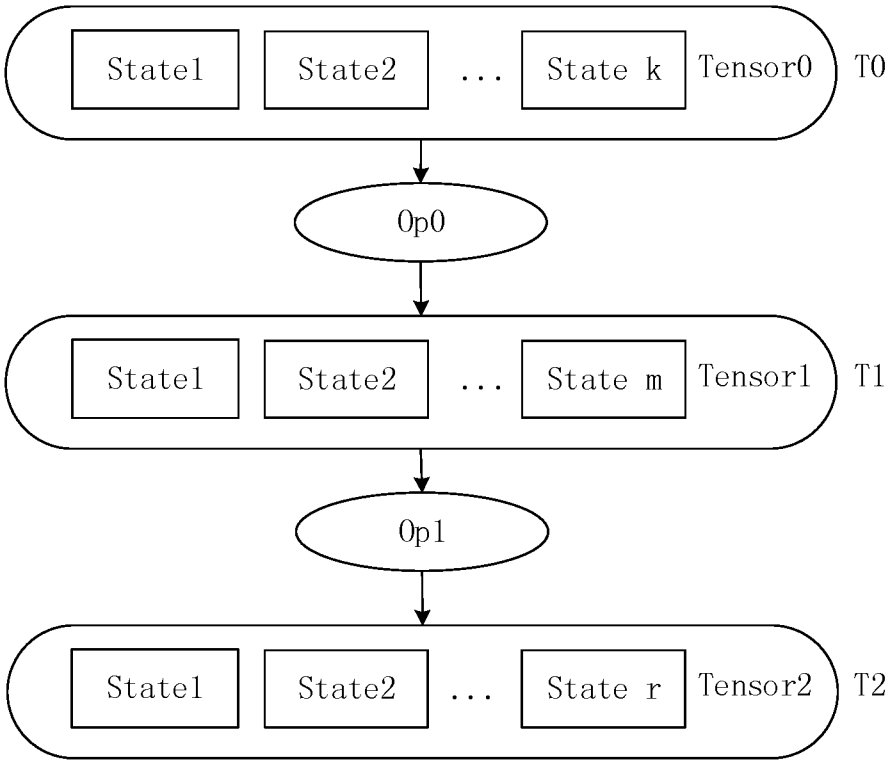
State1
State2
...
State k
Tensor0
T0
0p0
State1
State2
...
State m
Tensor1
T1
0p1
State1
State2
...
State r
Tensor2
T2

图 6

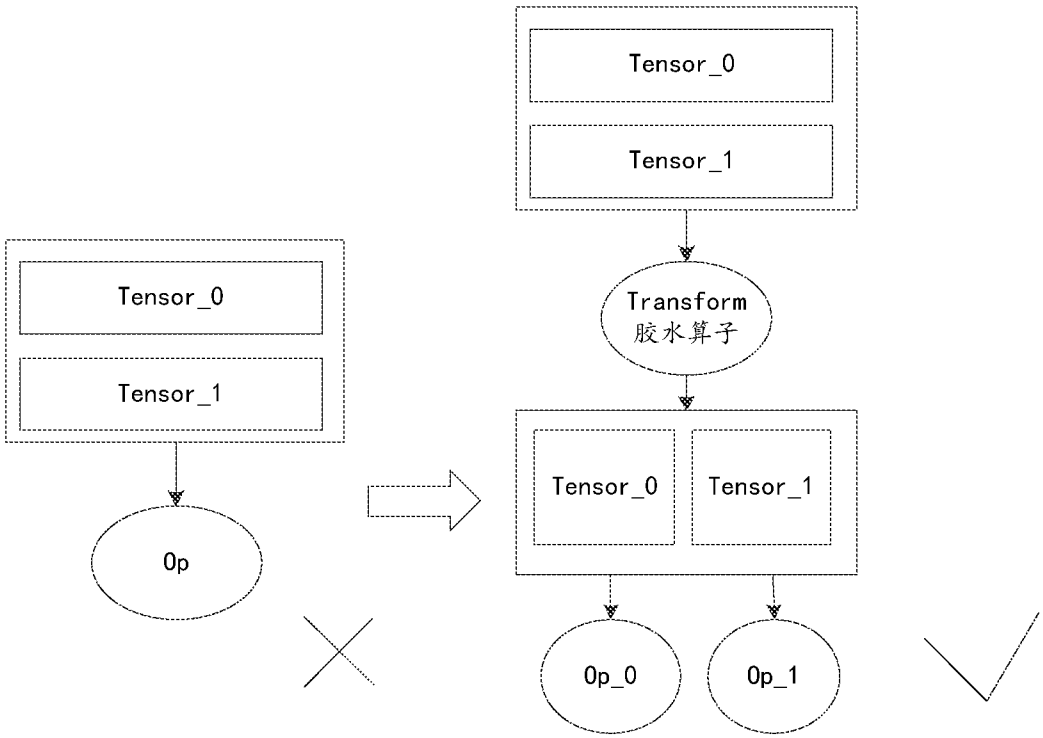
Tensor_0
Tensor_1
Transform
胶水算子
Tensor_0
Tensor_1
Tensor_0
Tensor_1
0p
0p_0
0p_1

图 7

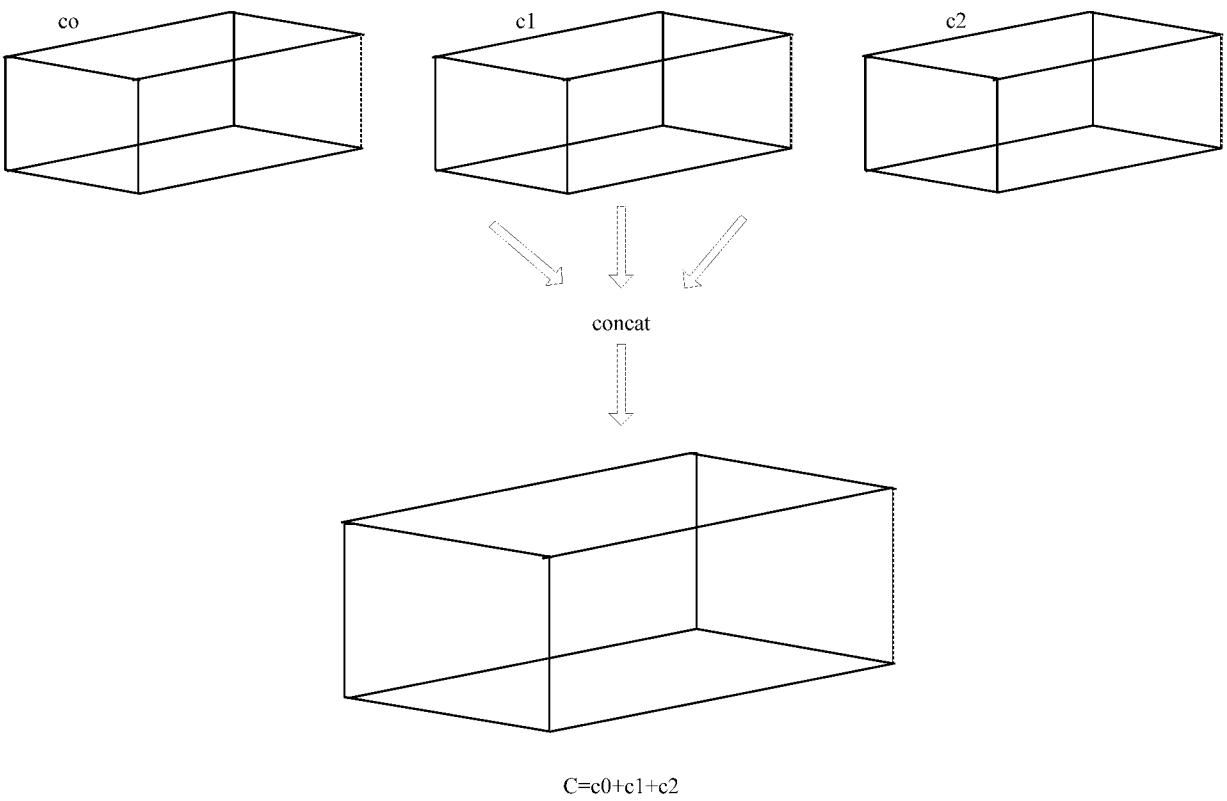

图 8A

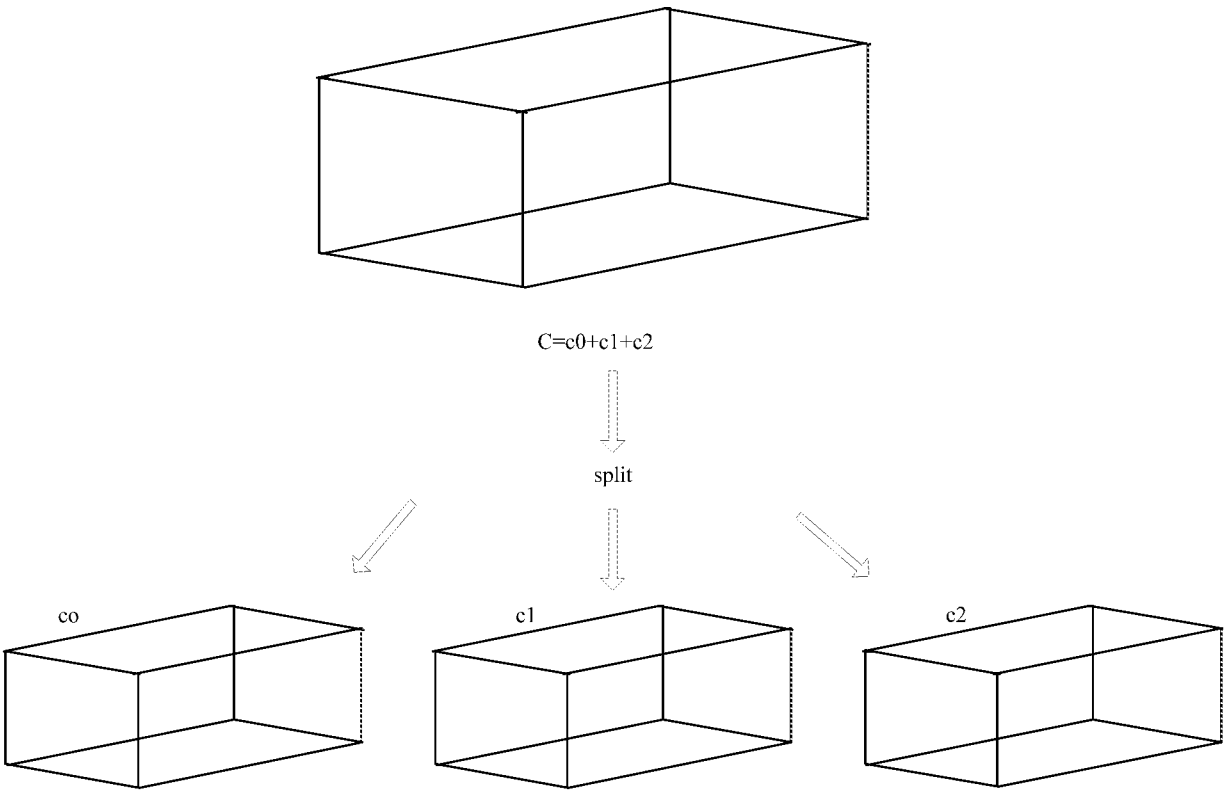

图 8B

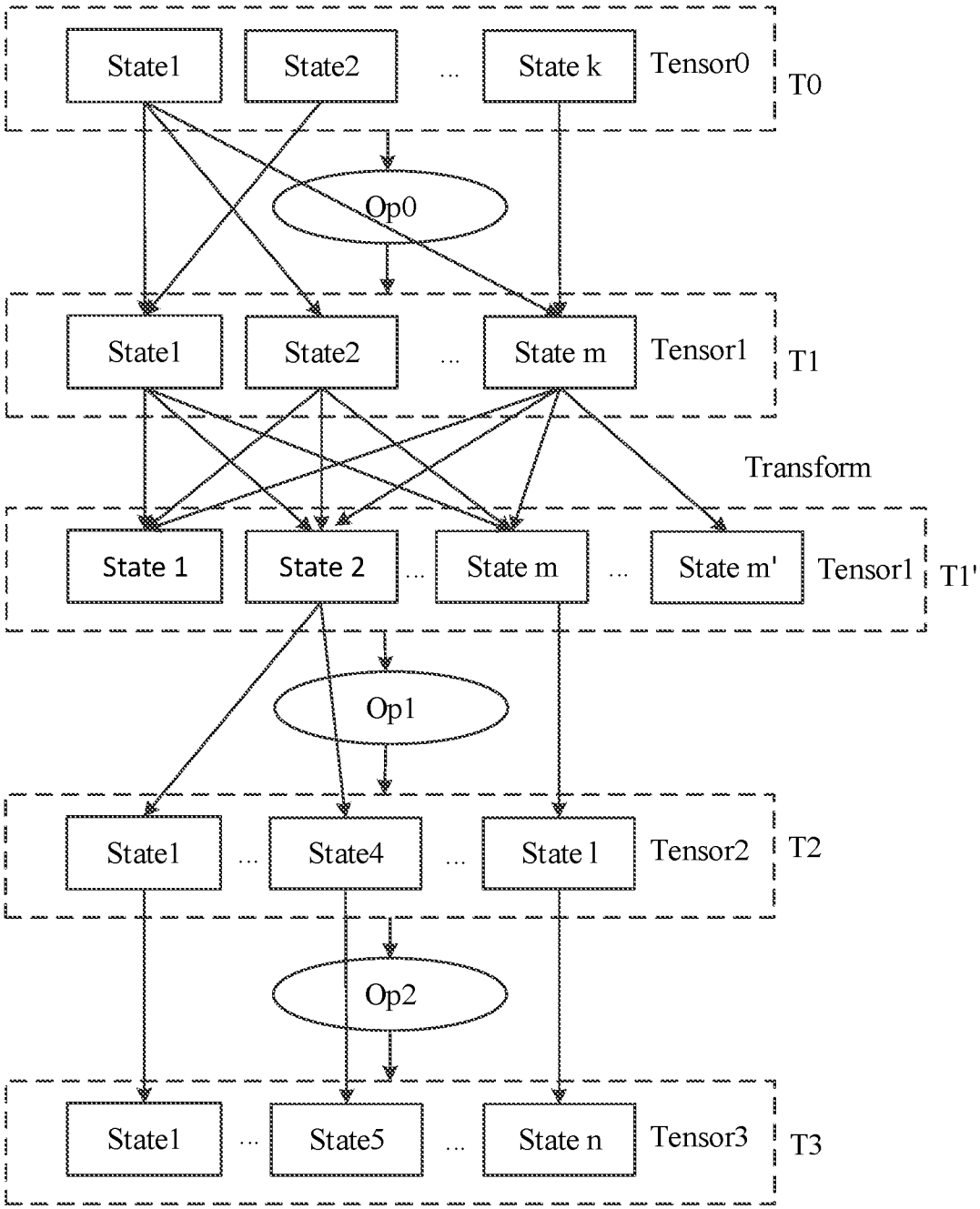
State1
State2
...
State k
Tensor0
T0
Op0
State1
State2
...
State m
Tensor1
T1
Transform
State 1
State 2
...
State m
...
State m'
Tensor1
T1'
Op1
State1
...
State4
...
State l
Tensor2
T2
Op2
State1
...
State5
...
State n
Tensor3
T3

图 9

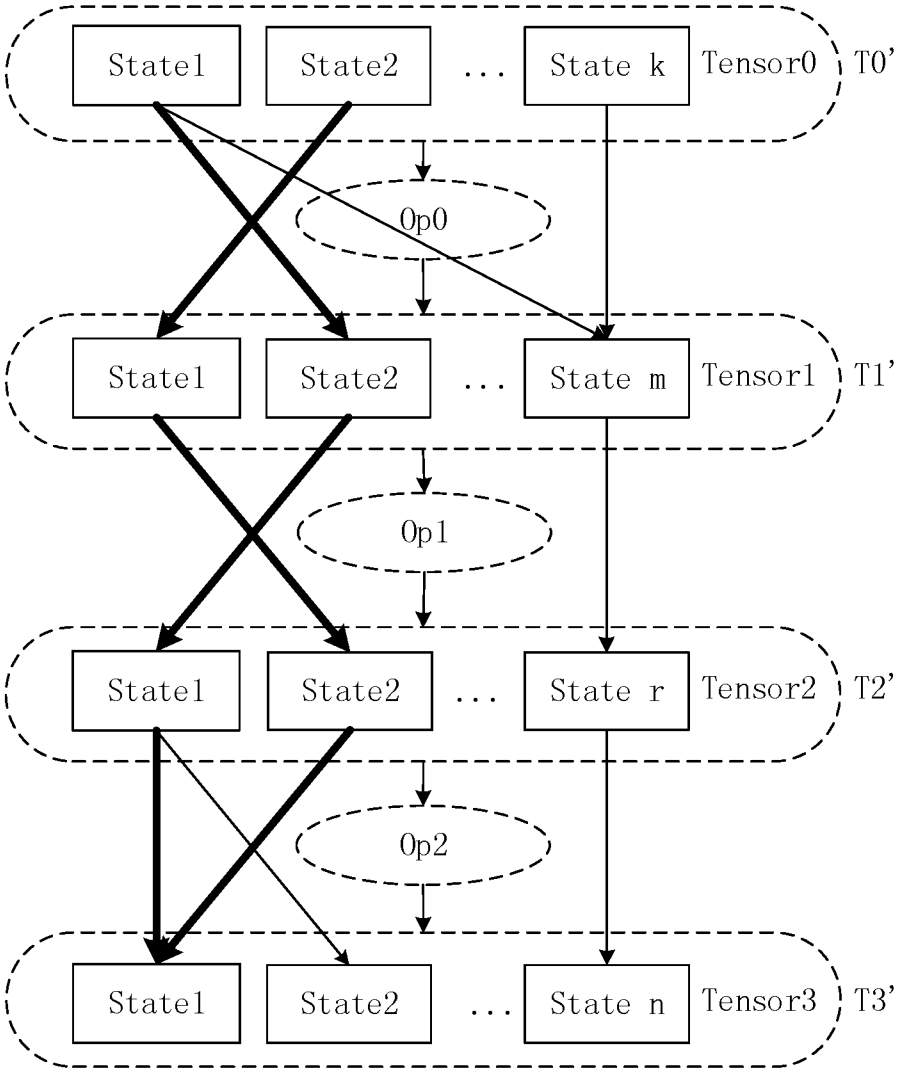
State1
State2
...
State k
Tensor0
T0'
0p0
State1
State2
...
State m
Tensor1
T1'
0p1
State1
State2
...
State r
Tensor2
T2'
0p2
State1
State2
...
State n
Tensor3
T3'

图 10

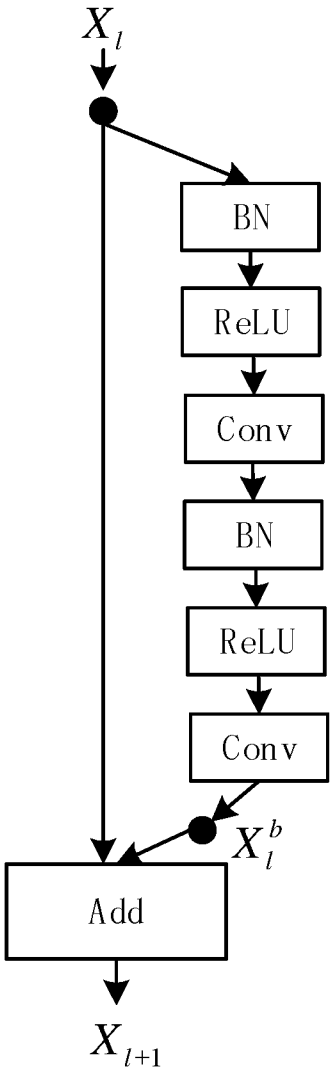
X_l
BN
ReLU
Conv
BN
ReLU
Conv
X_l^b
Add
X_{l+1}

图 11

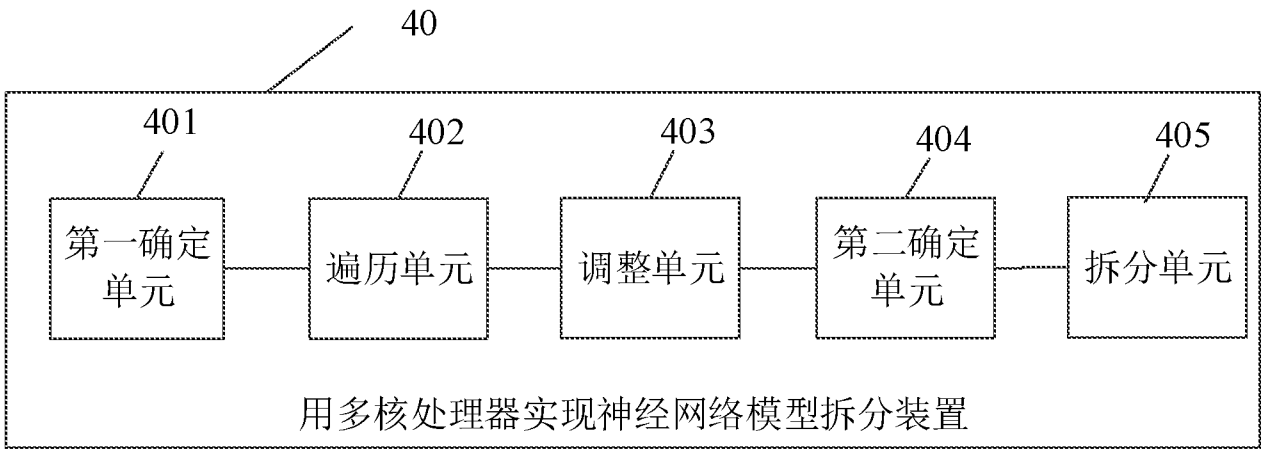
40
401
402
403
404
405
第一确定单元
遍历单元
调整单元
第二确定单元
拆分单元
用多核处理器实现神经网络模型拆分装置

图 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/116820

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/063(2006.01)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)

G06N

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)

CNABS, CNTXT, CNKI, DWPI, SIPOABS, USTXT: 算子, 胶水, 连接, 拼接, 拆分, 路径, 神经网络, 权重, operator , gluewater, connect, join, split, path, NN, neural w network, weight

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 110689121 A (SHANGHAI CAMBRICON INFORMATION TECHNOLOGY CO., LTD.) 14 January 2020 (2020-01-14) claims 1-14, description, page 6 line 44 - page 7 line 22	1-20
Y	CN 109657782 A (CAMBRICON TECHNOLOGIES CO., LTD.) 19 April 2019 (2019-04-19) description, paragraphs [0081]-[0099], [0183], [0223]-[0255], figures 1-9	1-5, 12-20
Y	CN 110008950 A (NANJING UNIVERSITY) 12 July 2019 (2019-07-12) description, paragraphs [0038]-[0041]	1-5, 12-20
A	CN 109299142 A (SUN YAT-SEN UNIVERSITY) 01 February 2019 (2019-02-01) entire document	1-20
A	US 2019239095 A1 (VERIZON PATENT AND LICENSING INC.) 01 August 2019 (2019-08-01) entire document	1-20

☐ 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
17 December 2020	**28 December 2020**
Name and mailing address of the ISA/CN **China National Intellectual Property Administration (ISA/CN)** **No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/116820

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 110689121 A	14 January 2020	None	
CN 109657782 A	19 April 2019	None	
CN 110008950 A	12 July 2019	None	
CN 109299142 A	01 February 2019	None	
US 2019239095 A1	01 August 2019	US 10728773 B2	28 July 2020

国际检索报告

国际申请号
PCT/CN2020/116820

A. 主题的分类

G06N 3/063(2006.01)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)

G06N

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））

CNABS,CNTXT,CNKI,DWPI,SIPOABS,USTXT:算子，胶水，连接，拼接，拆分，路径，神经网络，权重，operator ，gluewater，connect，join，split，path，NN，neural w network，weight

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
PX	CN 110689121 A (上海寒武纪信息科技有限公司) 2020年 1月 14日 (2020-01-14) 权利要求1-14，说明书第6页第44行-第7页第22行	1-20
Y	CN 109657782 A (北京中科寒武纪科技有限公司) 2019年 4月 19日 (2019-04-19) 说明书第[0081]-[0099]段、[0183]段、[0223]-[0255]段、图1-9	1-5，12-20
Y	CN 110008950 A (南京大学) 2019年 7月 12日 (2019-07-12) 说明书第[0038]-[0041]段	1-5，12-20
A	CN 109299142 A (中山大学) 2019年 2月 1日 (2019-02-01) 全文	1-20
A	US 2019239095 A1 (VERIZON PATENT & LICENSING INC) 2019年 8月 1日 (2019-08-01) 全文	1-20

☐ 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 12月 17日	2020年 12月 28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 冯慧萍 电话号码 62411838

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2020/116820

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)
CN 110689121 A	2020年 1月 14日	无	
CN 109657782 A	2019年 4月 19日	无	
CN 110008950 A	2019年 7月 12日	无	
CN 109299142 A	2019年 2月 1日	无	
US 2019239095 A1	2019年 8月 1日	US 10728773 B2	2020年 7月 28日