(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 113283613 B
(45)授权公告日 2021.11.09

(21)申请号 202110833986.0
(22)申请日 2021.07.23
(65)同一申请的已公布的文献号
申请公布号 CN 113283613 A
(43)申请公布日 2021.08.20
(73)专利权人 上海燧原科技有限公司
地址 201306 上海市浦东新区中国(上海)自由贸易试验区临港新片区业盛路188号A-522室
(72)发明人 石恒 姜天雨 刘育良
(74)专利代理机构 北京品源专利代理有限公司 11332
代理人 孔凡红
(51)Int.Cl.
G06N 20/00(2019.01)
G06F 40/253(2020.01)
(56)对比文件
CN 101873323 A,2010.10.27
CN 105183650 A,2015.12.23
CN 112925566 A,2021.06.08
审查员 李亚楠

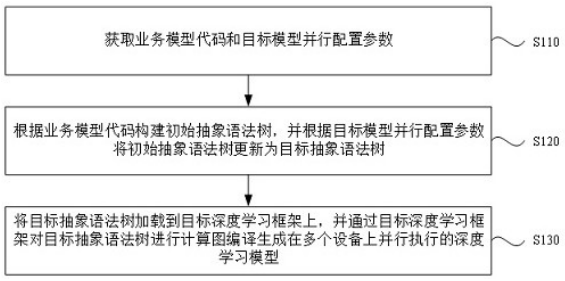
权利要求书3页 说明书16页 附图5页

(54)发明名称

深度学习模型的生成方法、优化方法、装置、设备及介质

(57)摘要

本发明实施例公开了一种深度学习模型的生成方法、优化方法、装置、设备及介质。该深度学习模型的生成方法，包括：获取业务模型代码和目标模型并行配置参数；根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。上述技术方案主要作用于模型代码编译阶段，能够避免对业务模型代码进行侵入式修改，实现了使深度学习模型支持模型并行的自动化。



CN 113283613 B

1.一种深度学习模型的生成方法,其特征在于,包括:

获取业务模型代码和目标模型并行配置参数;

根据所述业务模型代码构建初始抽象语法树,并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树;

将所述目标抽象语法树加载到目标深度学习框架上,并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型;

根据所述业务模型代码构建初始抽象语法树,并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树,包括:

构建所述业务模型代码的模块依赖图,以及与所述模块依赖图上每个模块对应的初始抽象语法树;

基于PASS机制根据所述目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树;

所述基于PASS机制根据所述目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树,包括:

根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性;

根据所述目标模型并行特性索引匹配的PASS操作,并基于所述匹配的PASS操作将所述模块依赖图上对应模块的当前抽象语法树进行更新,以使能所述目标模型并行特性;

返回执行根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性的操作,直至完成对所有需要使能的模型并行特性;

所述初始抽象语法树指的是根据业务模型代码直接构建的,不支持模型并行的抽象语法树;所述目标抽象语法树是对初始抽象语法树更新后得到的,支持模型并行的抽象语法树;

所述目标模型并行特性指的是按照目标模型并行配置参数中的任意一种配置参数运行;

所述业务模型代码,是指设定场景下设定业务所用的模型代码;

所述模块依赖图,是指由多个互相具有依赖关系的模块,每个模块对应一个完整的抽象语法树。

2.根据权利要求1所述的方法,其特征在于,构建所述业务模型代码的模块依赖图,以及与所述模块依赖图上每个模块对应的初始抽象语法树,包括:

根据所述业务模型代码确定主代码文件,根据所述主代码文件构建模块依赖图的初始模块以及与所述初始模块对应的待定抽象语法树;

将所述初始模块作为当前处理模块;

遍历所述当前处理模块的待定抽象语法树,确定所述当前处理模块所依赖的目标代码文件,将与所述目标代码文件对应的模块在所述待定抽象语法树中删除,得到与所述当前处理模块对应的初始抽象语法树;

根据所述目标代码文件构建所述当前处理模块的子模块,以及与所述子模块对应的待定抽象语法树;

将所述子模块作为所述当前处理模块,返回执行遍历所述当前处理模块的待定抽象语法树的操作,直至不存在需要构建初始抽象语法树的子模块;

所述待定抽象语法树，指的是根据代码文件直接生成的抽象语法树。

3.根据权利要求1所述的方法，其特征在于，构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树，包括：

根据与所述目标深度学习框架匹配的模块依赖图模块属性以及抽象语法树结构，构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树。

4.根据权利要求1所述的方法，其特征在于，所述目标模型并行配置参数包括：与分片模型并行模式对应的配置参数，和/或与流水线模型并行模式对应的配置参数。

5.根据权利要求1-4任一项所述的方法，其特征在于，通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型，包括：

通过所述目标深度学习框架将所述目标抽象语法树按照所述目标模型并行配置参数切割为多个抽象语法树子树，并根据每个所述抽象语法树子树编译生成相应的计算子图，并保存子图间的依赖关系；其中，每个所述计算子图用于单独加载到一个设备上去执行。

6.一种深度学习模型的优化方法，其特征在于，包括：

依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；

根据如权利要求1-5任一项所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；

根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

7.一种深度学习模型的生成装置，其特征在于，包括：

模型代码及模型并行配置获取模块，用于获取业务模型代码和目标模型并行配置参数；

抽象语法树构建更新模块，用于根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；

抽象语法树加载执行模块，用于将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型；

所述抽象语法树构建更新模块，包括：模块依赖图及抽象语法树构建单元和抽象语法树转换单元；

所述模块依赖图及抽象语法树构建单元，用于构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树；

所述抽象语法树转换单元，用于基于PASS机制根据所述目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树；

所述抽象语法树转换单元，具体用于根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性；根据所述目标模型并行特性索引匹配的PASS操作，并基于所述匹配的PASS操作将所述模块依赖图上对应模块的当前抽象语法树进行更新，以使能所述目标模型并行特性；返回执行根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性的操作，直至完成对所有需要使能的模型并行特性；

所述初始抽象语法树指的是根据业务模型代码直接构建的，不支持模型并行的抽象语法树；所述目标抽象语法树是对初始抽象语法树更新后得到的，支持模型并行的抽象语法树；

所述目标模型并行特性指的是按照目标模型并行配置参数中的任意一种配置参数运行；

所述业务模型代码，是指设定场景下设定业务所用的模型代码；

所述模块依赖图，是指由多个互相具有依赖关系的模块，每个模块对应一个完整的抽象语法树。

8.一种深度学习模型的优化装置，其特征在于，包括：

模型并行配置参数选取模块，用于依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；

待定深度学习模型生成模块，用于根据如权利要求1-5任一项所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；

目标深度学习模型确定模块，用于根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

9.一种电子设备，其特征在于，所述电子设备包括：

一个或多个处理器；

存储器，用于存储一个或多个程序，

当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现如权利要求1-5中任一项所述的方法，或者实现如权利要求6所述的方法。

10.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，该程序被处理器执行时实现如权利要求1-5中任一项所述的方法，或者实现如权利要求6所述的方法。

深度学习模型的生成方法、优化方法、装置、设备及介质

技术领域

[0001] 本发明实施例涉及计算机技术领域，尤其涉及一种深度学习模型的生成方法、优化方法、装置、设备及介质。

背景技术

[0002] 随着深度学习的发展，深度学习模型的规模呈现急剧增长的趋势，类似于Bert(Bidirectional Encoder Representations from Transformers，基于变换器的双向编码器表征)-Large、GPT3(General Pre-trained Transformer-3)等新的超大规模深度学习模型来说，主流的人工智能芯片或者GPGPU(General-Purpose computing on Graphics Processing Units，通用图形处理器)的存储容量已经远远不足以容纳单个模型进行训练的存储需求，计算资源的集群化部署已经成为一种必然趋势。

[0003] 目前，主流的模型并行框架，比如Fairscale、Deepspeed、Megatron等，均需要侵入式的修改模型代码来实现模型并行。对于人工智能企业来说，业务所用的模型代码需要进行大量的侵入式修改才能够从传统的数据并行模式转化为模型并行模式，而且需要大量的人力成本、学习成本以及试错成本，往往也会需要一定的项目周期来进行业务模型代码的迁移。

发明内容

[0004] 本发明实施例提供了一种深度学习模型的生成方法、优化方法、装置、设备及介质，以实现使深度学习模型支持模型并行的自动化，避免对业务模型代码进行侵入式修改。

[0005] 第一方面，本发明实施例提供了一种深度学习模型的生成方法，包括：

[0006] 获取业务模型代码和目标模型并行配置参数；

[0007] 根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；

[0008] 将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0009] 第二方面，本发明实施例还提供了一种深度学习模型的优化方法，包括：

[0010] 依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；

[0011] 根据如本发明任意实施例所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；

[0012] 根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

[0013] 第三方面，本发明实施例还提供了一种深度学习模型的生成装置，包括：

[0014] 模型代码及模型并行配置获取模块，用于获取业务模型代码和目标模型并行配置参数；

[0015] 抽象语法树构建更新模块，用于根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；

[0016] 抽象语法树加载执行模块，用于将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0017] 第四方面，本发明实施例还提供了一种深度学习模型的优化装置，包括：

[0018] 模型并行配置参数选取模块，用于依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；

[0019] 待定深度学习模型生成模块，用于根据如本发明任意实施例所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；

[0020] 目标深度学习模型确定模块，用于根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

[0021] 第五方面，本发明实施例还提供了一种电子设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序所述处理器执行所述程序时实现如本发明任意实施例中所述的深度学习模型的生成方法。

[0022] 第六方面，本发明实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现如本发明任意实施例中所述的深度学习模型的生成方法。

[0023] 第七方面，本发明实施例还提供了一种电子设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序所述处理器执行所述程序时实现如本发明任意实施例中所述的深度学习模型的优化方法。

[0024] 第八方面，本发明实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现如本发明任意实施例中所述的深度学习模型的优化方法。

[0025] 本发明实施例提供的技术方案，在获取到业务模型代码和目标模型并行配置参数之后，首先根据业务模型代码构建初始抽象语法树，然后根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树，再将目标抽象语法树加载到目标深度学习框架上，通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。上述技术方案主要作用于模型代码编译阶段，使生成的深度学习模型具备模型并行的特性，也即使生成的深度学习模型支持模型，能够避免对业务模型代码进行侵入式修改，实现了使深度学习模型支持模型并行的自动化。

附图说明

[0026] 图1是本发明实施例一中的一种深度学习模型的生成方法的流程图；

[0027] 图2是本发明实施例一中的一种流水线模式下流水聚合数的示例图；

[0028] 图3是本发明实施例二中的一种深度学习模型的生成方法的流程图；

[0029] 图4是本发明实施例二中的一种模块依赖图的示例图；

[0030] 图5是发发明实施例二中的一种模块依赖图的示例图；

[0031] 图6是本发明实施例二中的一种模块依赖图和初始抽象语法树的构建流程示意图；

[0032] 图7是本发明实施例三中的一种深度学习模型的生成方法的流程图；

[0033] 图8是本发明实施例四中的一种深度学习模型的优化方法的流程图；

[0034] 图9是本发明实施例五中的一种深度学习模型的生成装置的模块结构示意图；

[0035] 图10是本发明实施例六中的一种深度学习模型的优化装置的模块结构示意图；

[0036] 图11是本发明实施例七中的一种电子设备的结构示意图。

具体实施方式

[0037] 下面结合附图和实施例对本发明作进一步的详细说明。可以理解的是，此处所描述的具体实施例仅仅用于解释本发明，而非对本发明的限定。另外还需要说明的是，为了便于描述，附图中仅示出了与本发明相关的部分而非全部结构。

[0038] 在更加详细地讨论示例性实施例之前应当提到的是，一些示例性实施例被描述成作为流程图描绘的处理或方法。虽然流程图将各项操作(或步骤)描述成顺序的处理，但是其中的许多操作可以被并行地、并发地或者同时实施。此外，各项操作的顺序可以被重新安排。当其操作完成时所述处理可以被终止，但是还可以具有未包括在附图中的附加步骤。所述处理可以对应于方法、函数、规程、子例程、子程序等等。

[0039] 实施例一

[0040] 图1为本发明实施例一提供的一种深度学习模型的生成方法的流程图，本实施例可适用于使深度学习模型支持模型并行的情况，该方法可以由本发明实施例提供的深度学习模型的生成装置来执行，该装置可采用软件和/或硬件的方式实现，并一般可集成在电子设备中。

[0041] 如图1所示，本实施例提供的深度学习模型的生成方法，包括：

[0042] S110、获取业务模型代码和目标模型并行配置参数。

[0043] 业务模型代码，指的是设定场景下设定业务所用的模型代码，例如可以是人工智能场景下设定业务(如自然语言处理等)所用的模型代码。示例性的，业务模型代码可以是与机器翻译、语音识别、文本分析等业务对应的BERT模型代码，可以是与文本生成业务对应的GPT-2模型代码、GPT-3模型代码，也可以是与图片分类、物体识别等业务对应的Resnet(Residual Network，残差网络)模型代码等。

[0044] 目标模型并行配置参数，指的是任意一组使深度学习模型支持模型并行的配置参数。

[0045] 模型并行，指的是将一个深度学习模型切分成为不同的部分，并将其部署在多个设备上分别执行，以通过跨设备的通信和规约得到最终需要结果的技术过程。在模型并行中，会将一个完整的深度学习模型分成多个不同的部分，并分别部署到不同的设备上，只将一份数据输入至与模型入口对应的设备，设备间通过通信的方式进行数据传输，最终通过接力的方式计算得到最终结果。由于模型并行切分的是模型的参数，所以模型并行并不需要在每个设备上都存放全部的参数，以此大大节约了内存的使用，这也是超大规模的深度学习模型只能通过模型并行来进行部署的原因所在。

[0046] 可选的，目标模型并行配置参数包括：与分片(Sharding)模型并行模式(下述简称

分片模式)对应的配置参数,和/或与流水线(Pipeline)模型并行模式(下述简称流水线模式)对应的配置参数。

[0047] 模型并行的基本运行模式包括:分片模式和流水线模式。分片模式,指的是对于一个计算需求,将输入数据复制到多个设备,同时将权重参数切片分片到多个设备,由每个设备分别计算得到部分结果,并通过一个集成通信计算进行规约得到完整的计算结果。流水线模式,指的是将模型按照执行顺序切成连续的几个阶段,各阶段部分模型单独放在一个设备上,将数据输入第一个阶段部分模型后,等待第一个阶段部分模型运行完毕,在第一个阶段部分模型运行完毕后,第二个阶段部分模型接力运行第一个阶段部分模型的输出数据,此时第一段部分模型继续接收下一批次的数据,与其他阶段部分模型并行地运行。

[0048] 在本实施例中,模型并行的模式可以是分片模式,可以是流水线模式,也可以是分片模式和流水线模式的混合。可选的,目标模型并行配置参数,可以包括:分片尺寸、分片维度、流水段数、流水聚合数、设备映射信息等。

[0049] 其中,分片尺寸,也可以称之为分片数量,指的是分片模式下分片切分的数量。分片尺寸的值可以为2的幂,如1、2、4、8等整数。

[0050] 分片维度,指的是分片模式下对模型参数进行分片时的切分维度。分片维度可以是最高维,分片维度也可以是最低维等。假设,以第0维表示最高维、以第-1维表示最低维。例如,对于矩阵乘,参数为[K,N]两个维度,若分片维度的配置参数为0,则表示切分在K维度上,若分片维度的配置参数为-1,则表示切分在N维度上;再例如,对于卷积核,参数为[Ci,h,w,Co]四个维度,若分片维度的配置参数为0,则表示切分在Ci维度上,若分片维度的配置参数为-1,则表示切分在Co维度上。

[0051] 流水段数,指的是流水线模式下整个模型分段放置的阶段数。流水段数可以是1,也可以是大于1的整数。可选的,在实施过程中,由于单服务器节点最大的设备数量为8,流水段数可以最大设置为8。

[0052] 流水聚合数,指的是流水线模式下允许整个流水线中一次性处理的数据批次总数,也即在数据批次传输至阶段部分设备后,对应的梯度尚未更新到模型参数之前,阶段部分设备一次性处理的数据批次总数。如图2所示的示例,流水聚合数为4,以第一阶段部分设备(layer1)为例,其一次性处理的数据批次总数为4,分别为数据批次1、数据批次2、数据批次3和数据批次4。

[0053] 设备映射信息,指的是设备与流水段以及分片之间的映射关系。其中,设备映射信息为二维映射信息,分别为流水段维度和分片维度,设备映射信息也即是每个流水段下每个分片与设备之间的映射关系,流水段维度为外层维度,分片维度为内层维度。可选的,设备映射信息可以以设备映射表的形式存在,每个表项对应于一个流水段数、一个分片数以及一个设备名,如第n流水段下第m个分片的设备名。

[0054] 示例性的,目标模型并行配置参数可以如下:

[0055] “mode”: “sharding”, //分片模式

[0056] “sharding_size”:4, //分片尺寸为4

[0057] “sharding_dim”:-1, //分片维度为最低维

[0058] “stage_cnt”:2, //流水段数为2

[0059] “accum_degree”:1, //流水聚合度为1

[0060] “device_mapping”:(//设备映射表

[0061] “stage_0:(

[0062] “shard_0”: “/device”:0”

[0063] “shard_1”: “/device”:1”

[0064] “shard_2”: “/device”:2”

[0065] “shard_3”: “/device”:3”

[0066])

[0067] “stage_1:(

[0068] “shard_0”: “/device”:5”

[0069] “shard_1”: “/device”:6”

[0070] “shard_2”: “/device”:7”

[0071] “shard_3”: “/device”:8”

[0072])

[0073])

[0074] S120、根据业务模型代码构建初始抽象语法树，并根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树。

[0075] 抽象语法树(Abstract Syntax Tree，AST)，是源代码语法结构的一种抽象表示。它以树状的形式表现编程语言的语法结构，树上的每个节点都表示源代码中的一种结构。

[0076] 在本实施例中，初始抽象语法树指的是根据业务模型代码直接构建的，不支持模型并行的抽象语法树，而目标抽象语法树是对初始抽象语法树更新后得到的，支持模型并行的抽象语法树。

[0077] 可选的，在根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树时，可以基于PASS处理来实现。

[0078] PASS，在编译器中表示对程序的某种IR(Intermediate Representation，中间表达)进行优化变形的功能类。IR是LLVM(Low Level Virtual Machine，底层虚拟机)编译器架构以及其他主流的现代编译器共用的术语，表示程序从源代码到机器码、汇编码之间的所有表示方式的统称。其中，抽象语法树就是最上层的一种典型的IR。

[0079] PASS可以简单的理解成一个函数，输入是一种IR，输出是同一种IR经过优化变形后的结果。在本实施例中，遵循标准的PASS规范，将抽象语法树结构作为输入，使能后的抽象语法树结构作为输出。具体的，PASS处理输入的是不支持模型并行的初始语法树，输出的是支持模型并行的目标语法树。通过PASS处理对目标模型并行配置参数进行使能，以此得到与业务模型代码对应的支持模型并行的目标抽象语法树。

[0080] 示例性的，以切片方式为例，更新得到的目标抽象语法树，计算量大的二元操作会切分，同时这个二元操作对应的权重参数也会被切分，放置到不同的设备上。因此，在抽象语法树的变形上，首先去识别对应的抽象语法树二元计算节点，将其切分成几个分片节点；然后找到其对应的权重参数，同时也将该参数节点进行切分；最后将修改过的所有节点的“设备”属性都会得到基于目标模型并行配置参数的更新。由此，使变形得到的目标抽象语法树支持模型并行。

[0081] 比如，当进行矩阵乘操作的时候，抽象语法树中会存在一个二元的matmul计算操

作，如果二元操作的左值和右值符合PASS的要求，则对该节点进行变形，具体来说，就是将矩阵权重按照配置要求分配到指定的设备上，并且依据具体情况确定是否插入AllReduce规约计算，最后计算结果也会根据PASS分散或者聚合地出现在不同设备上，进而，一个二元的matmul计算操作也就变换成了一系列符合模型并行需求的操作序列。

[0082] S130、将目标抽象语法树加载到目标深度学习框架上，并通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0083] 目标深度学习框架，指的是任意一种深度学习框架，例如可以是Tensorflow、Pytorch、Mxnet、Mindspore等。

[0084] 将目标抽象语法树加载到目标深度学习框架上，目标深度学习框架将其转换为计算图。由于目标抽象语法树是支持模型并行的，故目标深度学习框架可以将目标抽象语法树转换为多个并行的计算子图结构，并对多个计算子图结构分别进行编译，将编译完成得到的多个计算子图发送到多个设备上去执行，以此得到了能够在多个设备上并行执行的深度学习模型，也即生成的深度学习模型是支持模型并行的。

[0085] 作为一种可选的实施方式，通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型，可以具体为：

[0086] 通过目标深度学习框架将目标抽象语法树按照目标模型并行配置参数切割为多个抽象语法树子树，并根据每个抽象语法树子树编译生成相应的计算子图，并保存子图间的依赖关系；其中，每个所述计算子图用于单独加载到一个设备上去执行。

[0087] 具体的，目标深度学习框架按照目标模型并行配置参数中的设备数量将目标抽象语法树切割为多个抽象语法树子树，并将每个抽象语法树子树转换成相应的计算子图结构，对计算子图结构进行编译生成相应的计算子图，并保存子图间的依赖关系。进而可以将每个计算子图单独加载到一个设备上去执行，也即将编译完成得到的多个计算子图发送到多个设备上去执行，以此使深度学习模型支持模型并行。

[0088] 示例性的，若深度学习模型的模型并行模式为分片模式，假设为N个分片，目标深度学习框架则会将目标抽象语法树切份为N个结构相同的抽象语法树子树，并针对每个抽象语法树子树分别进行计算图编译，得到结构相同的N个计算子图，其中，不同计算子图对应的设备编号不同(设备编号可详见设备映射信息)。进而，可以将编译完成得到的N个计算子图分别加载到对应编号的设备上去执行，以此实现深度学习模型的模型并行运行。

[0089] 本发明实施例提供的技术方案，在获取到业务模型代码和目标模型并行配置参数之后，首先根据业务模型代码构建初始抽象语法树，然后根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树，再将目标抽象语法树加载到目标深度学习框架上，通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。上述技术方案主要作用于模型代码编译阶段，使生成的深度学习模型具备模型并行的特性，也即使生成的深度学习模型支持模型，能够避免对业务模型代码进行侵入式修改，实现了使深度学习模型支持模型并行的自动化。

[0090] 实施例二

[0091] 图3为本发明实施例二提供的一种深度学习模型的生成方法的流程图。本实施例在前述实施例的基础上进行具体化，其中，根据业务模型代码构建初始抽象语法树，并根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树，可以具体为：

[0092] 构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树；基于PASS机制根据目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树。

[0093] 如图3所示，本实施例提供的深度学习模型的生成方法，包括：

[0094] S210、获取业务模型代码和目标模型并行配置参数。

[0095] S220、构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树。

[0096] 模块依赖图(module dependency graph)，是由多个互相具有依赖关系的模块，每个模块对应一个完整的抽象语法树。

[0097] 在获取到业务模型代码之后，可以根据业务模型代码构建与业务模型代码对应的模块依赖图，并针对模型依赖图中的每个模块构建与之对应的初始抽象语法树。

[0098] 示例性的，假设业务模型代码是在python编程语言下编写的，并假设业务模型代码的主代码文件为run_classifier.py，主代码文件即可模块依赖图中的初始模块或根模块，若该文件(也即初始模块)引用了其他的模块，那么会通过Import name1，或者From name2 import name3的语法来引用其他模块。在本实施例中，会获取主代码文件run_classifier.py所有的Import和FromImport的语句，找到对应的代码文件，并构建如图4所示的模块依赖图。如果模块name1然继续引用了其他文件(模块)，则迭代之前的过程，构建的模块依赖图可以参照图5。

[0099] 需要注意的是，编程语言的系统库引用，以及深度学习框架对应的模块等的引用会在构建模块依赖图时被过滤掉，目的在于使构建的模块依赖图中只包括与业务模块代码相关的文件模块。

[0100] 在构建得到的模块依赖图中，每个模块都对应于一份模型代码文件，对其进行编译即可得到对应的抽象语法树，作为与模块对应的初始抽象语法树。例如，针对各个模块对应的模型代码文件，将其通过编译器中的Parser、Lexer等操作即可构造出对应的抽象语法树。

[0101] 作为一种可选的实施方式，构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树，可以具体为：

[0102] S221、根据业务模型代码确定主代码文件，根据主代码文件构建模块依赖图的初始模块以及与初始模块对应的待定抽象语法树。

[0103] S222、将初始模块作为当前处理模块。

[0104] S223、遍历当前处理模块的待定抽象语法树，确定当前处理模块所依赖的目标代码文件，将与目标代码文件对应的模块在待定抽象语法树中删除，得到与当前处理模块对应的初始抽象语法树。

[0105] S224、根据目标代码文件构建所述当前处理模块的子模块，以及与子模块对应的待定抽象语法树。

[0106] S225、将所述子模块作为当前处理模块，返回执行S223，直至不存在需要构建初始抽象语法树的子模块。

[0107] 其中，待定抽象语法树，指的是根据代码文件直接生成的抽象语法树。根据该代码文件的依赖关系对待定抽象语法树进行调整，即可得到与该代码文件对应的初始抽象语法

树。

[0108] 读取业务模型代码的主代码文件，生成与主代码文件对应的待定抽象语法树，以及生成模块依赖图中与主代码文件对应的初始模块。将所述初始模块作为当前处理模块进行分析，遍历与当前处理模块对应的待定抽象语法树，判断是否存在当前处理模块(也即主文件代码或初始模块)所依赖的目标代码文件，若是，则将与目标代码文件对应的模块在与当前处理模块对应的待定抽象语法树中删除，以此得到与当前处理模块(也即主文件代码或初始模块)对应的初始抽象语法树。其中，在将与目标代码文件对应的模块在待定抽象语法树中删除之前，可以确定与目标代码文件对应的模块是否为真实的模型代码模块，若是，则将其在待定抽象语法树中删除，若否，则继续判断是否还存在当前处理模块(也即主文件代码或初始模块)所依赖的目标代码文件。

[0109] 针对确定出的当前处理模块(也即主文件代码或初始模块)所依赖的各个目标代码文件，依次根据各个目标代码文件构建模块依赖图中当前处理模块(也即主文件代码或初始模块)的子模块，并以此构造与各个子模块对应的待定抽象语法树。

[0110] 进而，依次根据各个子模块的依赖关系分别对与各个子模块对应的待定抽象语法树进行调整，得到与各个子模块对应的初始抽象语法树。其中，在针对任意一个模块生成初始抽象语法树时，均可以将其作为当前处理模块，执行如S223和S224所述的操作，以生成相应的初始抽象语法树。

[0111] 在对与各个子模块对应的待定抽象语法树进行调整时，子模块还可能存在其依赖的目标代码文件，进而在模块依赖图中继续构建子模块的子模块，以及相应模块的待定抽象语法树，以此迭代，即可得到与业务模型代码对应的完整的模块依赖图，以及模块依赖图中每个模块对应的初始抽象语法树。

[0112] 参照图6，作为一种具体的实施方式，针对构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树，可以具体为：

[0113] S21、读取业务模型代码的主代码文件。

[0114] S22、生成与主代码文件对应的抽象语法树，并生成模块依赖图中与主代码文件对应的初始模块。

[0115] 其中，此步骤生成的抽象语法树为与主代码文件对应的待定抽象语法树。

[0116] S23、创建待处理队列，并将初始模块添加至待处理队列中。

[0117] 其中，待处理队列为先进先出队列。

[0118] S24、判断待处理队列是否为空，若是，则执行S25，若否，则执行S26。

[0119] S25、结束流程。

[0120] S26、取出待处理队列中的首个模块N。

[0121] S27、遍历模块N的抽象语法树，确定其中的第一个Import和FromImport的模块P。

[0122] S28、判断模块P是否为真是的模型代码模块，若否，则执行S20，若是，则执行S29。

[0123] S29、删除模块N的抽象语法树中的模块P，在模块依赖图中创建模块N的子模块P，生成与模块P对应的抽象语法树，并将模块P添加至待处理队列中。

[0124] 其中，此步骤生成的抽象语法树为与模块P对应的待定抽象语法树。

[0125] S20、判断模块N的抽象语法树中是否存在Import和FromImport的模块，若是，则执行S27，若否，则执行S24。

[0126] 其中，当模块N的抽象语法树中是否存在Import和FromImport的模块时，此时模块N的抽象语法树即为模块N的初始抽象语法树。

[0127] 作为一种可选的实施方式，构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树，可以具体为：

[0128] 根据与目标深度学习框架匹配的模块依赖图模块属性以及抽象语法树结构，构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树。

[0129] 在本实施例中，在构建业务模型代码的模块依赖图时，依据与目标深度学习框架匹配的模块依赖图模块属性，在构建与模块依赖图上每个模块对应的初始抽象语法树时，依据与目标深度学习框架匹配的抽象语法树结构，以此实现对任意深度学习框架的兼容。

[0130] 由于本实施例技术方案的生效阶段是在模型代码编译加载过程中，未到深度学习框架生效的阶段，因此，只需根据与目标深度学习框架匹配的模块依赖图模块属性以及抽象语法树结构，构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树，即可实现完全兼容任意的深度学习框架。

[0131] 示例性的，针对tensorflow的功能实现中，抽象语法树的结构与模块依赖图的模块属性等，与针对pytorch、mxnet等其他框架的所生成的抽象语法树的结构以及模块依赖图的模块属性会有细微的差别，只需在构建模块依赖图以及抽象语法树时，参照对应框架的结构属性，即可在具体实施中得以区分并协调一致。

[0132] S230、基于PASS机制根据目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树。

[0133] 在本实施例中，基于编译器领域的PASS机制根据目标模型并行配置参数对初始抽象语法树进行处理，使其更新为支持模型并行的目标抽象语法树。其中，PASS处理是通用式的，以此实现本实施例提供的深度学习模型的生成方法的通用性。

[0134] 示例性的，用户在编写业务模型代码时无论是采用什么模型，如BERT模型、GPT-2模型、GPT-3模型、Resnet模型等，tensorflow下卷积的接口函数都只有类似于‘tf.nn.conv2d’或者‘tf.keras.layers.Conv2D’等使用方法，在抽象语法树中会存在一个对应的子树。对抽象语法树进行操作使其支持模型并行，即可达到方法通用化的目的。

[0135] 在本实施例中，通用的抽象语法树变形，支持不同的模型，实现了在各种模型上自动的使能与模型并行相关的特性，进而使模型支持模型并行，无需针对每个模型进行定制化的修改，提高了方案的通用性。

[0136] 作为一种可选的实施方式，基于PASS机制根据目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树，可以具体为：

[0137] 根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性；

[0138] 根据所述目标模型并行特性索引匹配的PASS操作，并基于所述匹配的PASS操作将所述模块依赖图上对应模块的当前抽象语法树进行更新，以使能所述目标模型并行特性；

[0139] 返回执行根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性的操作，直至完成对所有需要使能的模型并行特性。

[0140] 其中，目标模型并行特性指的是按照目标模型并行配置参数中的任意一种配置参数运行。

[0141] 可选的，可以按照目标模型并行配置参数中包括的具体配置参数和/或深度学习

模型中涉及的计算类型，确定各个目标模型并行特性使能的顺序，本实施例对此不作具体限定。

[0142] 在确定当前需要使能的目标模型并行特性之后，还需要根据深度学习模型中涉及多个计算类型依次对不同的计算类型进行目标模型并行特性的使能。例如，首先针对所有的矩阵乘计算使能目标模型并行特性，然后针对所有的卷积计算使能目标模型并行特性。

[0143] 当针对任意一种计算类型使能目标模型并行特性时，首先索引与目标模型并行特性使能对应的PASS操作，其中，PASS操作的类型可以是一个或多个，在索引到的PASS操作的类型为多个时，依次执行索引到的各个PASS操作。也即，按照索引到的PASS操作的顺序，依次选取一个PASS操作对模块依赖图上符合该PASS操作条件的模块的当前抽象语法树(即当前时刻的抽象语法树)进行更新，直至处理完索引到的各个PASS操作。

[0144] 示例性的，当索引到多个PASS操作时，多个PASS操作是顺序且首尾相连的，前一个PASS操作的IR输出作为下一个PASS操作的IR输入。当完成当前索引到的多个PASS操作时，就完成了针对当前循环迭代中目标模型并行特性的使能，相应的抽象语法树就获取了与这多个PASS操作对应的优化了。

[0145] 需要注意的是，基于PASS机制根据目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树，涉及多层迭代循环，第一层迭代循环指的是依据目标模型并行配置参数涉及的多种配置参数(也即多个需要使能的目标模型并行特性)确定的迭代循环，第二层迭代循环指的是在针对某个目标模型并行特性进行使能时针对不同的计算类型(如矩阵乘计算、卷积计算等)确定的迭代循环，第三层迭代循环指的是针对某种计算类型进行某个目标模型并行特性的使能时依据索引到的多个PASS操作确定的迭代循环。

[0146] 需要指出的是，模块依赖图中任意一个模块对应的初始抽象语法树进行更新得到目标抽象语法树的过程中，与模块对应的抽象语法树在初始抽象语法树和目标抽象语法树之间还会存在一个或多个中间状态的抽象语法树，也即模块对应的抽象语法树是经过多次更新才得到相应的目标抽象语法树。在前述的迭代循环过程中，涉及的当前抽象语法树，可以是模块的初始抽象语法树，也可以是模块的任意一个中间状态的抽象语法树，具体与其抽象语法树的更新次数相关。另外，各个模块的抽象语法树，指的在符合当前迭代循环中相应PASS操作的条件时，才会进行此轮更新，否则此轮迭代循环中抽象语法树不变。

[0147] S240、通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0148] 本实施例中，目标深度学习框架加载的目标抽象语法树，是与模块依赖图上各个模型分别对应的目标抽象语法树，也即与模块依赖图对应的目标抽象语法树，或者称之为与业务模型代码对应的目标抽象语法树。

[0149] 目标深度学习框架将目标抽象语法树按照目标模型并行配置参数中的设备数量切割为多个抽象语法树子树，并将每个抽象语法树子树转换成相应的计算子图结构，对计算子图结构进行编译生成相应的计算子图，并保存子图间的依赖关系。进而可以将每个计算子图单独加载到一个设备上去执行，也即将编译完成得到的多个计算子图发送到多个设备上去执行，以此使深度学习模型支持模型并行。

[0150] 在上述技术方案中，原封不动地将用户的业务模型代码作为输入，在模型代码编

译阶段，将其组织成为初始抽象语法树的结构之后，对其进行变形、修改、注入等操作，以使得到的目标抽象语法树支持模型并行，通过目标深度学习框架对目标抽象语法树进行计算图编译，以生成在多个设备上并行执行的深度学习模型。上述流程是在模型代码编译过程中自动完成的，对于用户来讲，完全不需要侵入式地进行模型代码的修改工作。而且，上述技术方案可完全兼容各种深度学习框架，弥补了在工业界基于Tensorflow的模型并行框架鲜有使用的缺陷。

[0151] 实施例三

[0152] 图7为本发明实施例三提供的一种深度学习模型的生成方法的流程图。本实施例在前述实施例的基础上提供了一种具体的实施方式。

[0153] 如图7所示，本实施例提供的深度学习模型的生成方法，包括：

[0154] S310、获取业务模型代码。

[0155] S320、判断业务模型代码是否使用模型并行，若是，则执行S330，若否，则执行S3100。

[0156] S330、初始化并获取目标模型并行配置参数。

[0157] 可选的，目标模型并行配置参数包括：与分片模式对应的配置参数，和/或与流水线模式对应的配置参数，具体可以包括分片尺寸、分片维度、流水段数、流水聚合数、设备映射信息等配置信息。

[0158] S340、构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的抽象语法树。

[0159] S350、判断是否完成与目标模型并行配置参数对应的抽象语法树变换，若否，则执行S360，若是，则执行S390。

[0160] S360、根据目标模型并行配置参数确定当前需要使能的目标模型并行特性。

[0161] S370、根据目标模型并行特性索引匹配的PASS操作。

[0162] 其中，PASS操作可以存储于全局注册表中。

[0163] S380、基于匹配的PASS操作将模块依赖图上对应模块的当前抽象语法树进行变换，执行S350。

[0164] 抽象语法树通过Pass机制，注入、修改、变形于该抽象语法树，得到具备模型并行功效的抽象语法树。

[0165] 上述业务模型代码使用模型并行的流程(S340-S380)，大致可以分为三个阶段，分别为：构建业务模型代码的模块依赖图，以及与模块依赖图中每个模块对应的抽象语法树；迭代地对目标模型并行配置参数进行使能；在每个迭代循环中，索引当前需要使能的目标模型并行特性，并从全局的PASS注册表中索引赌赢的PASS实现，对每个模块的抽象语法树进行更新。

[0166] S390、将目标抽象语法树加载到目标深度学习框架上，并通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0167] 目标深度学习框架，指的是任意一种深度学习框架，例如可以是Tensorflow、Pytorch、Mxnet、Mindspore等。

[0168] S3100、编译业务模型代码得到可执行代码，并通过目标深度学习框架加载可执行代码，进行计算图编译，并发到指定设备上执行。

[0169] 若业务模型代码不使用模型并行，则深度学习模型的生成流程分为：模型代码编译；模型代码在深度学习框架上加载执行；深度学习框架生成计算图结构；深度学习框架进行计算图编译；深度学习框架加载计算图并执行。若业务模型代码使用模型并行，则深度学习模型的生成流程分为：模型代码构建抽象语法树；抽象语法树通过Pass机制，注入、修改、变形于该抽象语法树，得到具备模型并行功效的抽象语法树；将具备模型并行功效的抽象语法树在深度学习框架上加载执行；深度学习框架生成计算图结构；深度学习框架进行计算图编译，得到深度学习模型；深度学习框架加载计算图并执行。

[0170] 本实施例未尽详细解释之处请参见前述实施例，在此不再赘述。

[0171] 若业务模型代码使用模型并行，上述技术方案主要作用与模型代码的编译阶段，以使其支持模型并行。现有主流的模型并行框架，针对用户提供了模型并行对应的程序接口集(API)，要求用户使用该程序接口集来编写具备模型并行的代码，因此用户需要大量的工作将本地/单机单设备执行的深度学习模型代码，迁移到支持多机多卡的模型并行。而本申请技术方案原封不动地将用户的模型代码作为输入，在方案内部将其组织成为抽象语法树的结构之后，其进行变形、修改、注入，此流程是在方案中自动完成的，因此对于用户来说，完全不需要侵入式地进行模型代码的修改工作。

[0172] 实施例四

[0173] 图8为本发明实施例四提供的一种深度学习模型的优化方法的流程图，本实施例可适用于使深度学习模型支持模型并行并自动调优的情况，该方法可以由本发明实施例提供的深度学习模型的优化装置来执行，该装置可采用软件和/或硬件的方式实现，并一般可集成在电子设备中。

[0174] 如图8所示，本实施例提供的深度学习模型的优化方法，包括：

[0175] S410、依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数。

[0176] 可选的，目标模型并行配置参数包括：与分片模式对应的配置参数，和/或与流水线模式对应的配置参数，具体可以包括分片尺寸、分片维度、流水段数、流水聚合数、设备映射信息等配置信息。

[0177] 候选的多组模型并行配置参数，指的是预先确定的用于使业务模型代码支持模型并行的多组模型并行配置参数。

[0178] 可选的，根据由模型并行配置信息构造的配置参数搜索空间确定候选的多组模型并行配置参数。其中，配置参数搜索空间的维数与目标模型并行配置参数涉及的参数类型数量是匹配的，每个维度上参数取值与相应的参数类型相关，本实施例对此不作具体限定。

[0179] 示例性的，若目标模型并行配置参数包括分片尺寸、分片维度、流水段数以及流水聚合数，则配置参数搜索空间的维数为四维。例如，模型并行配置信息中分片尺寸配置为1、2、4、8中一个，分片维度配置为0或-1，流水段数配置为5、6、7、8中一个，流水聚合数配置为4或5，则根据由模型并行配置信息构造的配置参数搜索空间能够确定4*2*4*2=64组模型并行配置参数作为候选的多组模型并行配置参数。

[0180] 在候选的多组模型并行配置参数中依次选择一组模型并行配置参数作为目标模型并行配置参数，实现业务模型代码的模型并行。

[0181] S420、根据如本申请任意实施例所述的深度学习模型的生成方法，生成待定深度

学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据。

[0182] 根据如本申请任意实施例所述的深度学习模型的生成方法，也即执行如下操作：获取业务模型代码和目标模型并行配置参数；根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0183] 关于本申请其他实施例所述的深度学习模型的生成方法，在此不再赘述。

[0184] 此时生成的深度学习模型，本实施例中称之为待定深度学习模型。根据候选的一组模型并行配置参数，便可以生成一个深度学习模型作为一个待定深度学习模型。

[0185] 每当生成一个待定深度学习模型之后，对该待定深度学习模型进行训练，例如执行预设数量个(如100个)训练循环，收集待定深度学习模型的性能数据，例如按照性能记录表中性能表项记录性能指标，本实施例对性能指标不作具体限定。

[0186] S430、根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

[0187] 若针对候选的多组模型并行配置参数均生成相应的待定深度学习模型，并完成对各待定深度学习模块的训练，收集得到各待定深度学习模块的性能数据，则对各待定深度学习模型的性能数据进行分析，根据性能数据在这多个待定深度学习模型中确定出一个深度学习模型作为目标深度学习模型，例如，将性能数据最优的一个深度学习模型作为目标深度学习模型。

[0188] 在确定目标深度学习模型之后，可以将对目标深度学习模型对应的一组模型并行配置参数进行持久化，以实现对深度学习模型的自动调优。可选的，性能记录表中除了记录性能指标，还记录相应的模型并行配置参数，进而可以将与最优性能指标对应的一组模型并行配置参数进行持久化。

[0189] 本实施例未尽详细解释之处请参见前述实施例，在此不再赘述。

[0190] 上述技术方案不仅使业务模型代码支持并行，还实现了深度学习模型支持模型并行的自动调优，其能够以一种通用的方式对不同模型均能实现动态调优，进而得到合适的模型部署方案。

[0191] 实施例五

[0192] 图9为本发明实施例五提供的一种深度学习模型的生成装置的结构示意图，该装置可采用软件和/或硬件的方式实现，一般可集成在电子设备中。如图9所示，该深度学习模型的生成装置具体包括：模型代码及模型并行配置获取模块510、抽象语法树构建更新模块520和抽象语法树加载执行模块530。其中，

[0193] 模型代码及模型并行配置获取模块510，用于获取业务模型代码和目标模型并行配置参数；

[0194] 抽象语法树构建更新模块520，用于根据所述业务模型代码构建初始抽象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；

[0195] 抽象语法树加载执行模块530，用于将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0196] 本发明实施例提供的技术方案，在获取到业务模型代码和目标模型并行配置参数之后，首先根据业务模型代码构建初始抽象语法树，然后根据目标模型并行配置参数将初始抽象语法树更新为目标抽象语法树，再将目标抽象语法树加载到目标深度学习框架上，通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。上述技术方案主要作用于模型代码编译阶段，使生成的深度学习模型具备模型并行的特性，也即使生成的深度学习模型支持模型，能够避免对业务模型代码进行侵入式修改，实现了使深度学习模型支持模型并行的自动化。

[0197] 可选的，抽象语法树构建更新模块520，包括：模块依赖图及抽象语法树构建单元和抽象语法树转换单元，其中，

[0198] 模块依赖图及抽象语法树构建单元，用于构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树；

[0199] 抽象语法树转换单元，用于基于PASS机制根据所述目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树。

[0200] 进一步的，抽象语法树转换单元，具体用于根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性；根据所述目标模型并行特性索引匹配的PASS操作，并基于所述匹配的PASS操作将所述模块依赖图上对应模块的当前抽象语法树进行更新，以使能所述目标模型并行特性；返回执行根据所述目标模型并行配置参数确定当前需要使能的目标模型并行特性的操作，直至完成对所有需要使能的模型并行特性。

[0201] 进一步的，模块依赖图及抽象语法树构建单元，具体用于根据所述业务模型代码确定主代码文件，根据所述主代码文件构建模块依赖图的初始模块以及与所述初始模块对应的待定抽象语法树；将所述初始模块作为当前处理模块；遍历所述当前处理模块的待定抽象语法树，确定所述当前处理模块所依赖的目标代码文件，将与所述目标代码文件对应的模块在所述待定抽象语法树中删除，得到与所述当前处理模块对应的初始抽象语法树；根据所述目标代码文件构建所述当前处理模块的子模块，以及与所述子模块对应的待定抽象语法树；将所述子模块作为所述当前处理模块，返回执行遍历所述当前处理模块的待定抽象语法树的操作，直至不存在需要构建初始抽象语法树的子模块。

[0202] 可选的，模块依赖图及抽象语法树构建单元，具体用于根据与所述目标深度学习框架匹配的模块依赖图模块属性以及抽象语法树结构，构建所述业务模型代码的模块依赖图，以及与所述模块依赖图上每个模块对应的初始抽象语法树。

[0203] 可选的，所述目标模型并行配置参数包括：与分片模型并行模式对应的配置参数，和/或与流水线模型并行模式对应的配置参数。

[0204] 可选的，抽象语法树加载执行模块530，具体用于通过所述目标深度学习框架将所述目标抽象语法树按照所述目标模型并行配置参数切割为多个抽象语法树子树，并根据每个所述抽象语法树子树编译生成相应的计算子图，并保存子图间的依赖关系；其中，每个所述计算子图用于单独加载到一个设备上去执行。

[0205] 上述深度学习模型的生成装置可执行本发明任意实施例所提供的深度学习模型的生成方法，具备执行方法相应的功能模块和有益效果。

[0206] 实施例六

[0207] 图10为本发明实施例六提供的一种深度学习模型的优化装置的结构示意图，该装

置可采用软件和/或硬件的方式实现，一般可集成在电子设备中。如图10所示，该深度学习模型的优化装置具体包括：模型并行配置参数选取模块610、待定深度学习模型生成模块620和目标深度学习模型确定模块630。其中，

[0208] 模型并行配置参数选取模块610，用于依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；

[0209] 待定深度学习模型生成模块620，用于根据如本申请任意实施例所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；

[0210] 目标深度学习模型确定模块630，用于根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

[0211] 上述技术方案不仅使业务模型代码支持并行，还实现了深度学习模型支持模型并行的自动调优，其能够以一种通用的方式对不同模型均能实现动态调优，进而得到合适的模型部署方案。

[0212] 上述深度学习模型的优化装置可执行本发明任意实施例所提供的深度学习模型的优化方法，具备执行方法相应的功能模块和有益效果。

[0213] 实施例七

[0214] 图11是本发明实施例七提供的一种电子设备的结构示意图，如图11所示，该电子设备包括处理器710和存储器720；电子设备中处理器710的数量可以是一个或多个，图11中以一个处理器710为例；电子设备中的处理器710和存储器720可以通过总线或其他方式连接，图11中以通过总线连接为例。

[0215] 存储器720作为一种计算机可读存储介质，可用于存储软件程序、计算机可执行程序以及模块，如本发明实施例中的一种深度学习模型的生成方法对应的程序指令/模块(如图9所示的深度学习模型的生成装置中包括的模型代码及模型并行配置获取模块510、抽象语法树构建更新模块520和抽象语法树加载执行模块530)，又如本发明实施例中的一种深度学习模型的优化方法对应的程序指令/模块(如图10所示的深度学习模型的优化装置中包括的模型并行配置参数选取模块610、待定深度学习模型生成模块620和目标深度学习模型确定模块630)。处理器710通过运行存储在存储器720中的软件程序、指令以及模块，从而执行电子设备的各种功能应用以及数据处理，即实现上述的深度学习模型的生成方法或深度学习模型的优化方法。

[0216] 存储器720可主要包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需的应用程序；存储数据区可存储根据电子设备的使用所创建的数据等。此外，存储器720可以包括高速随机存取存储器，还可以包括非易失性存储器，例如至少一个磁盘存储器件、闪存器件、或其他非易失性固态存储器件。在一些实例中，存储器720可进一步包括相对于处理器710远程设置的存储器，这些远程存储器可以通过网络连接至电子设备。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0217] 实施例八

[0218] 本发明实施例八还提供一种存储有计算机程序的计算机可读存储介质，计算机程序在由电子设备执行时用于执行一种深度学习模型的生成方法，包括：

[0219] 获取业务模型代码和目标模型并行配置参数；根据所述业务模型代码构建初始抽

象语法树，并根据所述目标模型并行配置参数将所述初始抽象语法树更新为目标抽象语法树；将所述目标抽象语法树加载到目标深度学习框架上，并通过所述目标深度学习框架对所述目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型。

[0220] 或者，计算机程序在由电子设备执行时用于执行一种深度学习模型的优化方法，包括：

[0221] 依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数；根据如本发明任意实施例所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据；根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型。

[0222] 当然，本发明实施例所提供的存储有计算机程序的计算机可读存储介质，其计算机程序不限于如上的方法操作，还可以执行本发明任意实施例所提供的深度学习模型的生成方法中的相关操作，或者执行本发明任意实施例所提供的深度学习模型的优化方法中的相关操作。

[0223] 通过以上关于实施方式的描述，所属领域的技术人员可以清楚地了解到，本发明可借助软件及必需的通用硬件来实现，当然也可以通过硬件实现，但很多情况下前者是更佳的实施方式。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在计算机可读存储介质中，如计算机的软盘、只读存储器(Read-Only Memory，ROM)、随机存取存储器(Random Access Memory，RAM)、闪存(FLASH)、硬盘或光盘等，包括若干指令用以使得一台主板控制器执行本发明各个实施例的方法。

[0224] 值得注意的是，上述深度学习模型的生成装置或者深度学习模型的优化装置的实施例中，所包括的各个单元和模块只是按照功能逻辑进行划分的，但并不局限于上述的划分，只要能够实现相应的功能即可；另外，各功能单元的具体名称也只是为了便于相互区分，并不用于限制本发明的保护范围。

[0225] 注意，上述仅为本发明的较佳实施例及所运用技术原理。本领域技术人员会理解，本发明不限于这里所述的特定实施例，对本领域技术人员来说能够进行各种明显的变化、重新调整和替代而不会脱离本发明的保护范围。因此，虽然通过以上实施例对本发明进行了较为详细的说明，但是本发明不仅仅限于以上实施例，在不脱离本发明构思的情况下，还可以包括更多其他等效实施例，而本发明的范围由所附的权利要求范围决定。

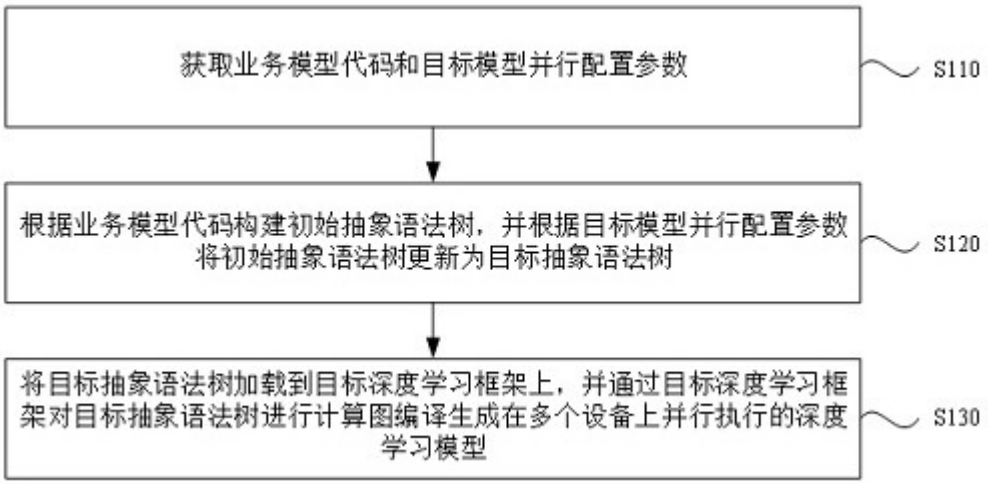


图1

Layer\Tick	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Layer4				1	2	3	4	4	3	2	1				W				1	2	3	4	4	3	2	1				W
Layer3			1	2	3	4			4	3	2	1			W			1	2	3	4			4	3	2	1			W
Layer2		1	2	3	4					4	3	2	1		W		1	2	3	4					4	3	2	1		W
Layer1	1	2	3	4							4	3	2	1	W	1	2	3	4							4	3	2	1	W
Layer\Tick	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
BackwordLayer1											4	3	2	1	W											4	3	2	1	W
BackwordLayer2										4	3	2	1		W										4	3	2	1		W
BackwordLayer3									4	3	2	1			W									4	3	2	1			W
BackwordLayer4								4	3	2	1				W								4	3	2	1				W
ForwardLayer4				1	2	3	4												1	2	3	4								
ForwardLayer3			1	2	3	4												1	2	3	4									
ForwardLayer2		1	2	3	4												1	2	3	4										
ForwardLayer1	1	2	3	4												1	2	3	4											

图2

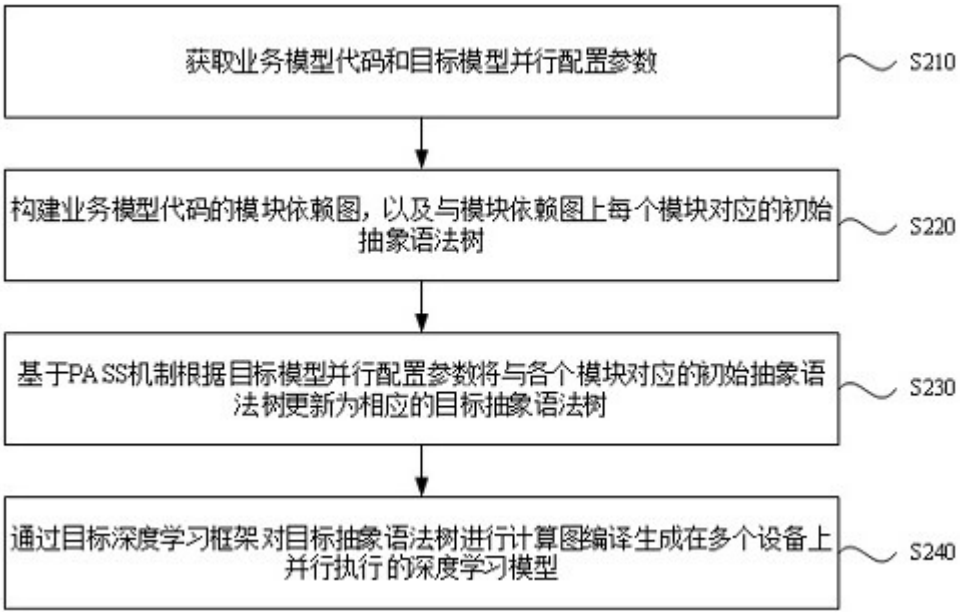

获取业务模型代码和目标模型并行配置参数
S210
构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的初始抽象语法树
S220
基于PASS机制根据目标模型并行配置参数将与各个模块对应的初始抽象语法树更新为相应的目标抽象语法树
S230
通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型
S240

图3

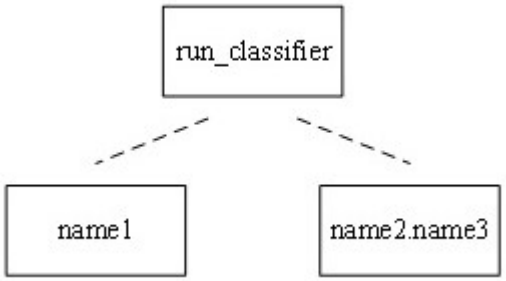

run_classifier
name1
name2.name3

图4

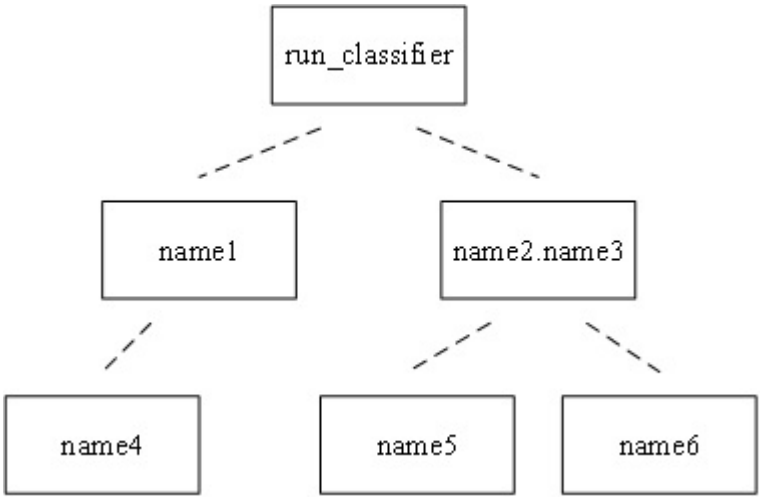

run_classifier
name1
name2.name3
name4
name5
name6

图5

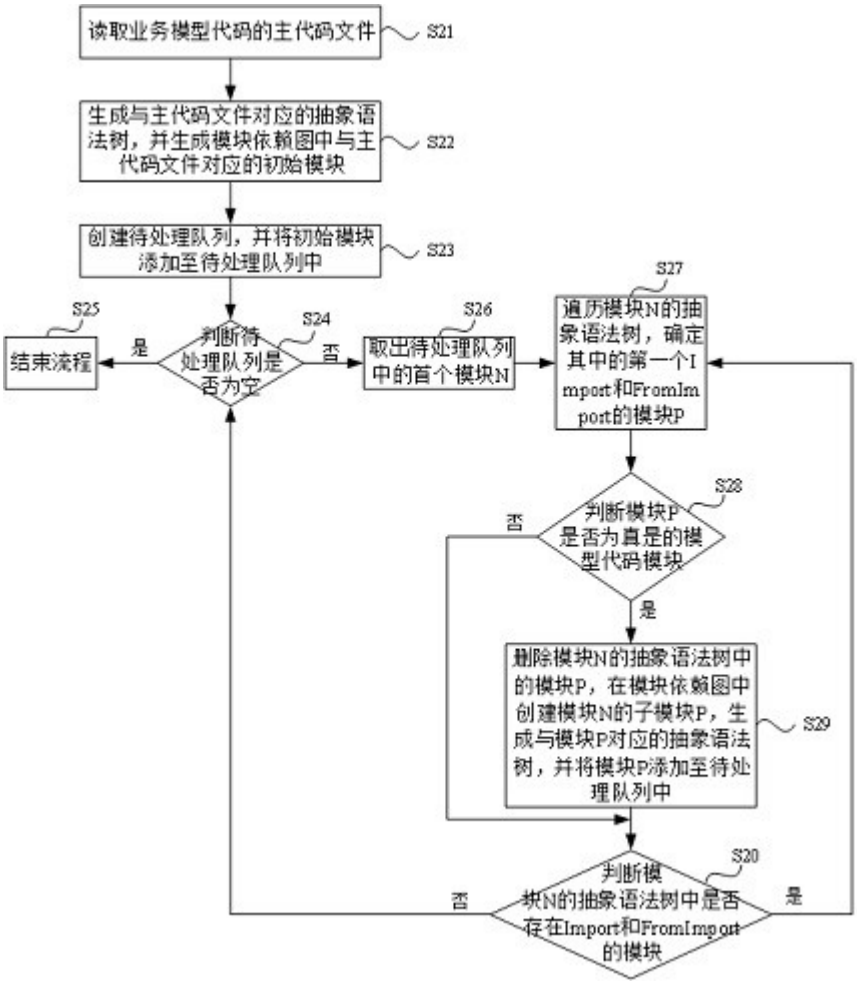

读取业务模型代码的主代码文件
S21
生成与主代码文件对应的抽象语法树，并生成模块依赖图中与主代码文件对应的初始模块
S22
创建待处理队列，并将初始模块添加至待处理队列中
S23
判断待处理队列是否为空
S24
是
否
结束流程
S25
取出待处理队列中的首个模块N
S26
遍历模块N的抽象语法树，确定其中的第一个Import和FromImport的模块P
S27
判断模块P是否为真是的模型代码模块
S28
否
是
删除模块N的抽象语法树中的模块P，在模块依赖图中创建模块N的子模块P，生成与模块P对应的抽象语法树，并将模块P添加至待处理队列中
S29
判断模块N的抽象语法树中是否存在Import和FromImport的模块
S20
否
是

图6

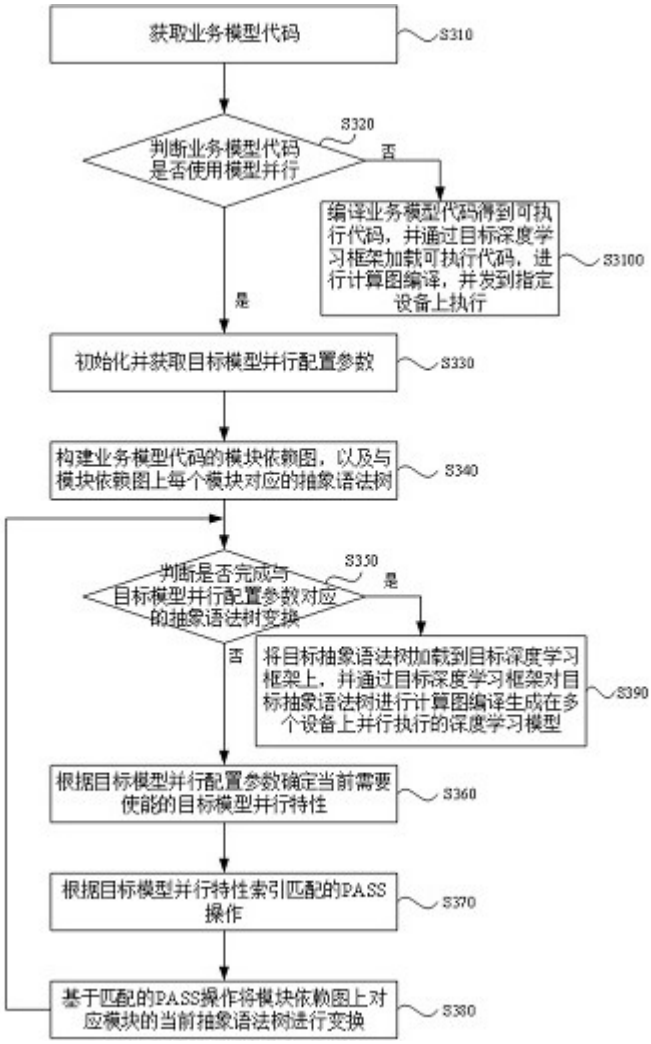

获取业务模型代码
S310
判断业务模型代码是否使用模型并行
S320
否
是
编译业务模型代码得到可执行代码，并通过目标深度学习框架加载可执行代码，进行计算图编译，并发到指定设备上执行
S3100
初始化并获取目标模型并行配置参数
S330
构建业务模型代码的模块依赖图，以及与模块依赖图上每个模块对应的抽象语法树
S340
判断是否完成与目标模型并行配置参数对应的抽象语法树变换
S350
是
否
将目标抽象语法树加载到目标深度学习框架上，并通过目标深度学习框架对目标抽象语法树进行计算图编译生成在多个设备上并行执行的深度学习模型
S390
根据目标模型并行配置参数确定当前需要使能的目标模型并行特性
S360
根据目标模型并行特性索引匹配的PASS操作
S370
基于匹配的PASS操作将模块依赖图上对应模块的当前抽象语法树进行变换
S380

图7

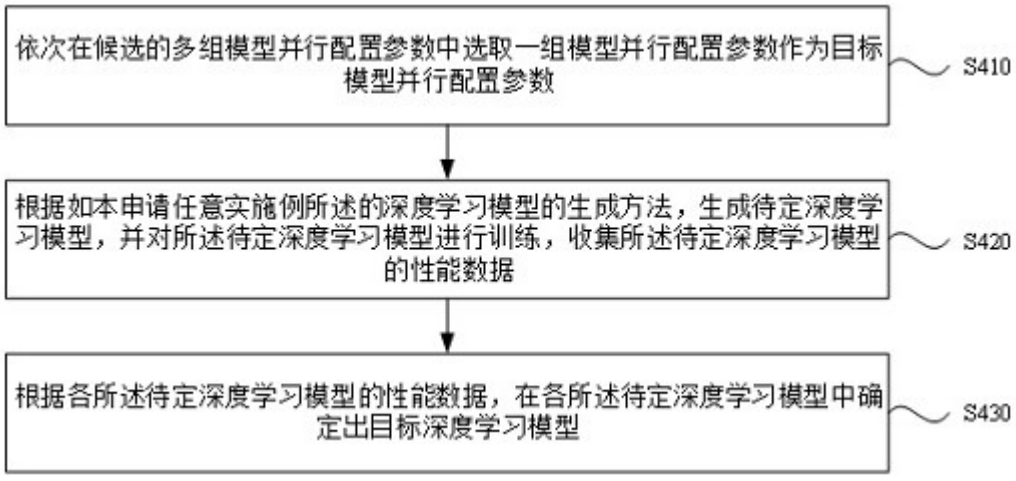

依次在候选的多组模型并行配置参数中选取一组模型并行配置参数作为目标模型并行配置参数
S410
根据如本申请任意实施例所述的深度学习模型的生成方法，生成待定深度学习模型，并对所述待定深度学习模型进行训练，收集所述待定深度学习模型的性能数据
S420
根据各所述待定深度学习模型的性能数据，在各所述待定深度学习模型中确定出目标深度学习模型
S430

图8


510
模型代码及模型并行配置获取模块
520
抽象语法树构建更新模块
530
抽象语法树加载执行模块

图9

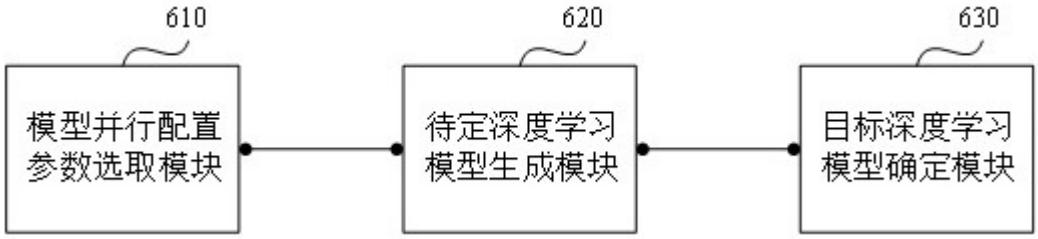

610
模型并行配置参数选取模块
620
待定深度学习模型生成模块
630
目标深度学习模型确定模块

图10

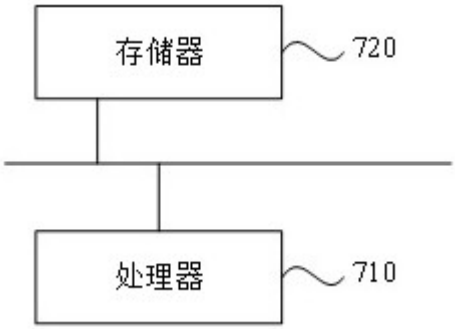

存储器
720
处理器
710

图11