

说明书

深度学习模型的元学习方法及深度学习模型的元学习系统

5 技术领域

本说明书涉及深度学习技术领域，尤其涉及一种深度学习模型的元学习方法以及深度学习模型的元学习系统。

背景技术

- 10 深度学习模型可以被广泛应用于智能投放场景，例如，广告投放场景、智能搜索场景以及智能推荐场景等。对于上述智能投放场景，可能存在冷启动问题，即由于客户转化数较少，导致技术指标（如，接收者操作特征曲线下面积（Area Under Curve，AUC）等）都普遍偏低。为了解决深度学习模型的冷启动问题，可以通过元学习的方式来帮助冷启动客户快速积累转化数。一般采用模型不可知元学习(Model Agnostic Meta Learning, MAML)算法来实现元学习过程，当
- 15 前业界的其他元学习算法基本基于 MAML 衍生出来，只改变神经网络层结构或者优化器的优化方法。由于深度学习模型的模型参数量大，相关技术中若采用现有的 MAML 优化算法进行深度学习模型的元学习，则存在效率低的问题。

背景技术部分的内容仅仅是发明人个人所知晓的信息，并不代表上述信息在本公开申请日之前已经进入公共领域，也不代表其可以成为本公开的现有技术。

20

发明内容

本说明书提供的深度学习模型的元学习方法以及深度学习模型的元学习系统，能够提高元学习效率。

- 25 第一方面，本说明书提供一种深度学习模型的元学习方法，应用于包括 N 个处理节点的集群，所述 N 为大于 1 的整数，其中，所述方法包括：获得训练数据集，所述训练数据集包括多个任务对应的训练样本；以及，通过所述 N 个处理节点并行地基于所述训练数据集对所述深度学习模型进行多次迭代训练，得到所述深度学习模型的参数，其中，在每次迭代训练中，所述 N 个处理节点中的每个处理节点使用所述训练数据集中的部分训练样本学习所述深度学习模型的部分参数，所述部分训练样本对应于所述多个任务中的同一任务。

在一些实施例中，所述深度学习模型的参数包括：嵌入层参数和密集层参数，所述每个处理节点使用所述部分训练样本学习所述嵌入层参数中的部分参数以及所述密集层参数中的全部参数。

5 在一些实施例中，所述训练数据集包括多个数据批；所述每次迭代训练的过程包括通过所述第 i 个处理节点：

确定目标数据批，并将所述目标数据批划分为支持集和查询集，所述目标数据批包括所述部分训练样本，所述目标数据批为所述多个数据批中的一个；执行内循环训练过程，所述内循环训练过程包括：确定第一嵌入层参数，基于所述支持集、所述第一嵌入层参数和当前密集层参数确定内循环损失函数，以及，通过优化所述内循环损失函数对所述第一嵌入层参数和所述当前密集层参数进行更新，其中，所述当前密集层参数被更新为中间密集层参数；以及，执行外循环训练过程，所述外循环训练过程包括：确定第二嵌入层参数，基于所述查询集、所述第二嵌入层参数和所述中间密集层参数确定外循环损失函数，以及，通过优化所述外循环损失函数来对所述第二嵌入层参数和所述中间密集层参数进行更新；其中， i 取值为小于或等于 N 的正整数，所述第一嵌入层参数和所述第二嵌入层参数分别对应所述嵌入层参数中的至少部分参数。

10
15

在一些实施例中，在所述执行内循环训练过程之前，所述方法还包括：通过与所述集群中的其他处理节点交互，获得所述支持集对应的当前嵌入层参数，并获得所述查询集对应的当前嵌入层参数；所述确定第一嵌入层参数，包括：将所述支持集对应的当前嵌入层参数确定为所述第一嵌入层参数；其中，所述第一嵌入层参数在所述内循环训练过程之后被更新为第三嵌入层参数；以及，所述确定第二嵌入层参数，包括：根据所述查询集对应的当前嵌入层参数或所述第三嵌入层参数确定所述第二嵌入层参数。

20

在一些实施例中，所述根据所述查询集对应的当前嵌入层参数或所述第三嵌入层参数确定所述第二嵌入层参数，包括：在所述查询集与所述支持集存在重叠元素的情况下，对于所述查询集的训练样本中所述重叠元素对应的嵌入层参数，将所述第三嵌入层参数确定为所述第二嵌入层参数；对于所述查询集的训练样本中除了所述重叠元素之外的元素所对应的嵌入层参数，将所述查询集对应的当前嵌入层参数确定为所述第二嵌入层参数；或者，在所述查询集与所述支持集不存在重叠元素的情况下，将所述查询集对应的当前嵌入层参数确定为所述第二嵌入层参数。

25

在一些实施例中，所述基于所述支持集、所述第一嵌入层参数和当前密集层参数确定内循环损失函数，包括：根据所述支持集、所述第一嵌入层参数和所述当前密集层参数进行前向计算，以确定所述内循环损失函数；所述通过优化所述内循环损失函数对所述第一嵌入层参数和所述当前密集层参数进行更新，包括：对所述内循环损失函数进行本地的反向梯度传递以对所述第一嵌入层参数和所述当前密集层参数进行更新。

30

5 在一些实施例中，所述通过优化所述外循环损失函数来对所述第二嵌入层参数进行更新，包括：计算所述外循环损失函数关于嵌入层参数的梯度，得到所述第 i 处理节点对应的分嵌入层梯度；通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度；对所述 N 个处理节点对应的分嵌入层梯度进行聚合计算，得到最终目标嵌入层梯度；以及，根据所述目标嵌入层梯度对所述第二嵌入层参数进行更新。

在一些实施例中，所述处理节点为图像处理单元 GPU 节点；所述通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度，包括：通过第一集合通信原语 AlltoAll，与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度。

10 在一些实施例中，所述通过优化所述外循环损失函数来对所述中间密集层参数进行更新，包括：确定所述外循环损失函数关于密集层参数的梯度，得到所述第 i 处理节点对应的分密集层梯度；通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度；对所述 N 个处理节点对应的分密集层梯度进行聚合计算，得到目标密集层梯度；以及，根据所述目标密集层梯度对所述中间密集层参数进行更新。

15 在一些实施例中，所述处理节点为图像处理单元 GPU 节点；所述通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度，包括：通过第二集合通信原语 AllReduce，与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度。

20 在一些实施例中，在所述通过所述 N 个处理节点并行地基于所述训练数据集对所述深度学习模型进行多次迭代训练之前，所述方法还包括：将所述嵌入层参数划分得到 N 个子集；以及，将第 i 个子集中的参数和所述密集层参数存储到第 i 个处理节点，其中， i 取值为小于或等于 N 的正整数。

在一些实施例中，所述训练数据集包括多个数据批；所述获得训练数据集，包括：

25 确定 P 个任务分别对应的训练样本，其中每个训练样本包含其所属任务的任务标识， P 为大于 1 的整数；以及，根据预先确定的批大小，将所述 P 个任务分别对应的训练样本拆分为 Q 个数据批，得到所述训练数据集，其中属于同一个数据批的训练样本的任务标识一致， Q 为大于 P 的整数。

30 在一些实施例中，所述处理节点为图像处理单元 GPU 节点，所述 N 个处理节点被部署于多个实体设备上，其中，在所述多次迭代训练过程中，同一实体设备内部的不同处理节点之间采用高速连接通道 NVLink 进行通信，不同实体设备之间通过远程直接内存访问 RDMA 进行通信。

在一些实施例中，上述深度学习模型为深度推荐模型 DLRM。

第二方面，本说明书提供一种深度学习模型的元学习系统，包括： N 个处理节点，所述 N 为大于 1 的整数；其中，每个处理节点包括：至少一个存储介质，存储有至少一个指令集，用

于进行深度学习模型的元学习；以及，至少一个处理器，同所述至少一个存储介质通信连接，其中，当所述深度学习模型的元学习系统运行时，每个处理节点中的所述至少一个处理器读取并执行所述至少一个指令集，以实现第一方面任意一实施例所提供的深度学习模型的元学习方法。

5 由以上技术方案可知，本说明书提供的深度学习模型的元学习方法应用于包括 N 个处理节点的集群。具体地，首先获得包括多个任务分别对应的训练样本的训练数据集。进一步地，通过上述集群中的 N 个处理节点并行地基于上述训练数据集对深度学习模型进行多次迭代训练，得到该深度学习模型的元学习参数。其中，在每次迭代训练中，将深度学习模型的参数分布至集群的各个处理节点，每个处理节点使用训练数据集中的部分训练样本学习该深度学习模型的部分参数。可见，一方面，每个处理节点使用部分训练样本来执行训练任务，从而实现训练数据并行训练；另一方面，每个处理节点负责学习模型的部分参数，从而实现模型并行训练。因此，本说明书实施例提供一种混合并行的元学习方式，节省了显存冗余，有利于提升深度学习模型的元学习效率。

15 本说明书提供的深度学习模型的元学习方法及深度学习模型的元学习系统的其他功能将在以下说明中部分列出。根据描述，以下数字和示例介绍的内容将对那些本领域的普通技术人员显而易见。本说明书提供的深度学习模型的元学习方法以及深度学习模型的元学习系统的创造性方面可以通过实践或使用下面详细示例中所述的方法、装置和组合得到充分解释。

附图说明

20 为了更清楚地说明本说明书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，下面描述中的附图仅仅是本说明书的一些实施例，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 1 示出了根据本说明书的实施例提供的一种深度学习模型的元学习系统的结构示意图；
图 2 示出了根据本说明书的一些实施例提供的一种计算设备的结构示意图；
25 图 3 示出了根据本说明书的实施例提供的一种深度学习模型的元学习方法的流程示意图；
图 4 示出了根据本说明书的实施例提供的一种深度推荐模型的结构示意图；
图 5 示出了根据本说明书的实施例提供的数据预处理的流程示意图；
图 6 示出了根据本说明书的实施例提供的训练阶段处理阶段加载样本的示意图；
图 7 示出了根据本说明书的实施例提供的一种深度学习模型的元学习框架的示意图；
30 图 8 示出了根据本说明书的实施例提供的一种深度学习模型的元学习方法的流程示意图。

具体实施方式

以下描述提供了本说明书的特定应用场景和要求，目的是使本领域技术人员能够制造和使用本说明书中的内容。对于本领域技术人员来说，对所公开的实施例的各种局部修改是显而易见的，并且在不脱离本说明书的精神和范围的情况下，可以将这里定义的一般原理应用于其他实施例和应用。因此，本说明书不限于所示的实施例，而是与权利要求一致的最宽范围。

5 这里使用的术语仅用于描述特定示例实施例的目的，而不是限制性的。比如，除非上下文另有明确说明，这里所使用的，单数形式“一”，“一个”和“该”也可以包括复数形式。当在本说明书中使用术语“包括”、“包含”和/或“含有”意思是指所关联的特征、整数、步骤、操作、元素和/或组件存在，但不排除一个或多个其他特征、整数、步骤、操作、元素、组件和/或组的存在或在该系统/方法中可以添加其他特征、整数、步骤、操作、元素、组件和/或组。

10 考虑到以下描述，本说明书的这些特征和其他特征、以及结构的相关元件的操作和功能、以及部件的组合和制造的经济性可以得到明显提高。参考附图，所有这些形成本说明书的一部分。然而，应该清楚地理解，附图仅用于说明和描述的目的，并不旨在限制本说明书的范围。还应理解，附图未按比例绘制。

本说明书中使用的流程图示出了根据本说明书中的一些实施例的系统实现的操作。应该清
15 楚地理解，流程图的操作可以不按顺序实现。相反，操作可以以反转顺序或同时实现。此外，可以向流程图添加一个或多个其他操作。可以从流程图中移除一个或多个操作。

在本说明书中，“X 包括 A、B 或 C 中的至少一个”所表达的意思是 X 至少包括 A，或 X 至少包括 B，或 X 至少包括 C。也就是说，X 可以只包括 A、B、C 的任意一个，或者同时包括 A、B、C 的任意组合以及其他可能的内容/元素。所述 A、B、C 的任意组合可以是 A、B、C、AB、
20 AC、BC、或 ABC。

在本说明书中，除非明确说明，否则结构之间产生的关联关系可以是直接的关联关系也可以是间接的关联关系。比如，当描述“A 与 B 连接”时，除非明确说明了 A 与 B 直接连接，否则应当理解成 A 可以与 B 直接连接，也可以与 B 间接地连接；再比如，当描述“A 在 B 之上”时，除非明确说明了 A 直接在 B 的上方（AB 相邻且 A 在 B 的上方），否则应当理解成 A 可以直接
25 在 B 的上方，A 也可以间接地在 B 之上（AB 之间隔着其他元素，且 A 在 B 的上方）。以此类推。

首先，对本说明书一个或多个实施例涉及的名词术语进行解释。

元学习（Meta Learning）：深度学习中的元学习是指让机器学会如何快速适应新任务的一种技术。通常情况下，深度学习需要大量的标记数据和训练时间才能学会解决特定的任务。但是
30 在现实生活中，需要机器能够在面对新任务时快速适应和学习，而不是从头开始重新训练。元学习的目标是让机器学会如何在面对新任务时进行快速学习。它通过训练机器在一系列不同的

任务上学习，并从中提取出共性和模式。通过这种方式，机器可以学会一种“学习如何学习”的能力，这样在面对新任务时，它可以利用之前学到的知识和经验来更快地适应和解决问题。

深度学习模型：搜索广告推荐（Advertisement Search Recommendation，ASR）场景的深度学习模型，在工程技术上表现为模型的 embedding layer（词表）非常巨大，通常几十 GB 或者上百 GB，由于在一个迭代中，词表中只有少数 embedding vector 会被用于模型更新，所以也被称为稀疏模型。

深度推荐模型（Deep Learning Recommendation Model，DLRM）：主要泛指在 ASR 场景下的深度学习模型。

AlltoAll: All-To-All 是一种集合通信原语（Collective Communications），其操作每一个节点的数据会分散到集群内所有节点上，同时每一个节点也会收集集群内所有节点的数据。AlltoAll 是对 ALLGATHER 的扩展，区别是 ALLGATHER 操作中，不同节点向某一节点收集到的数据是相同的，而在 ALLTOALL 中，不同的节点向某一节点收集到的数据是不同的。

AllReduce: AllReduce 是另外一种集合通信操作，用于在多个进程之间进行数据聚合。在 AllReduce 操作中，每个进程将自己的局部数据与其他进程的局部数据进行聚合，最终得到全局的聚合结果。

一种相关技术提供的 MAML 优化算法为：在图像处理单元（Ggraph Processing Unit，GPU）集群中，基于集合通信原语 AllReduce 并行化原 MAML 算法。具体的，集群中每个设备的 GPU 内存均分别容纳模型的所有参数。但是，对于 DLRM 来说，其嵌入层参数量庞大，大部分工业界的模型单个 GPU 内存无法容纳的问题。

本说明书实施例提供的深度学习模型的元学习方法以及深度学习模型的元学习系统，能够解决上述问题。本说明书实施例提供的深度学习模型的元学习方案中，一方面，每个处理节点使用部分训练样本来执行训练任务，从而实现训练数据并行训练；另一方面，每个处理节点负责学习模型的部分参数，从而实现模型并行训练。因此，本说明书实施例提供一种混合并行的元学习方式，节省了显存冗余，有利于提升深度学习模型的元学习效率，同时还能够避免单个处理节点无法容纳待学习参数（如，DLRM 的嵌入层参数）的问题。

下面，将结合附图 1 至图 8 对本说明书实施例提供的深度学习模型的元学习系统以及深度学习模型的元学习方法进行详细的说明。

图 1 示出了根据本说明书的实施例提供的一种深度学习模型的元学习系统 001 的示意图。为了描述方便，本说明书后文将深度学习模型的元学习系统 001 简称为系统 001。该系统 001 可以是包括一个或者多个实体设备 100 的集群系统，因此，系统 001 也可以称为集群 001。每个实体设备 100 可以部署一个或者多个处理节点（如，处理节点 i 和处理节点 j）。系统 001

可以执行本说明书描述的深度学习模型的元学习方法。上述深度学习模型的元学习方法在本说明书中的其他部分介绍。

图 2 示出了根据本说明书的一些实施例提供的一种计算设备 002 的结构示意图。上述计算设备 002 可以作为图 1 中的任意一个实体设备 100。上述计算设备 002 可以是通用计算机或专用计算机。例如，上述计算设备 002 可以是服务器、个人电脑、便携式电脑（比如笔记本计算机、平板电脑等），也可以是其他有计算能力的设备。

本说明书中的计算设备 002 可以包括一个或多个如下部件：处理器 210、存储器 220、总线 230、I/O 组件 240 以及通信端口 250。处理器 210、存储器 220、I/O 组件 240 以及通信端口 250 之间可以通过总线 230 连接。

处理器 210 可以包括一个或者多个处理核心。处理器 210 利用各种接口和线路连接整个计算设备 002 内的各个部分，通过运行或执行存储在存储器 220 内的指令、程序、代码集或指令集，以及调用存储在存储器 220 内的数据，执行本说明书描述的深度学习模型的元学习方法。可选地，处理器 210 可集成图像处理器（graphics processing unit, GPU）、中心处理器（central processing unit, CPU）和调制解调器等中的一种或几种的组合。处理器 210 可以采用数字信号处理(digital signal processing, DSP)、现场可编程门阵列(field-programmable gate array, FPGA)、可编程逻辑阵列(programmable logic Array, PLA)中的至少一种硬件形式来实现。其中，调制解调器用于处理无线通信。可以理解的是，上述调制解调器也可以不集成到处理器 210 中，单独通过一块通信芯片进行实现。

需要说明的是，本说明书实施例中，处理器 210 可以作为处理节点（worker）。具体的，每个计算设备 002 部署有至少一个处理节点（worker）。例如，参考图 1 中示出的实体设备 100 部署有两个处理节点，即处理节点 1（worker1）和处理节点 2（worker 2）。因此，仅仅为了说明问题，在图 2 中计算设备 002 中仅描述了一个处理器 210。本说明书中计算设备 002 还可以包括多个处理器，因此，本说明书中披露的操作和/或方法步骤可以如本说明书所述的由一个处理器执行，也可以由多个处理器联合执行。例如，如果在本说明书中计算设备 002 的处理器 210 执行步骤 A 和步骤 B，则应该理解，步骤 A 和步骤 B 也可以由两个不同处理器 210 联合或分开执行（例如，第一处理器执行步骤 A，第二处理器执行步骤 B，或者第一和第二处理器共同执行步骤 A 和 B）。

存储器 220 可以是非暂时性存储介质，也可以是暂时性存储介质。比如，所述数据存储装置可以包括闪存(Flash Memory)、磁盘、只读存储介质(Read-Only Memory, ROM)、随机存取存储介质(Random Access Memory, RAM)中的一种或多种。存储器 220 可用于存储指令、程序、代码、代码集或指令集。存储器 220 可包括存储程序区和存储数据区，其中，存储程序区可存储用于实现操作系统的指令、用于实现至少一个功能的指令（例如触控功能、声音播放功能、

图像播放功能等)、用于实现下述各个方法实施例的指令等,该操作系统可以是鸿蒙系统、安卓(Android)系统,包括基于 Android 系统深度开发的系统、IOS 系统,包括基于 IOS 系统深度开发的系统或其它系统。

其中,上述处理器 210 可以同上述存储器 220 通过内部通信总线 230 通信连接。上述处理器 210 用以执行上述存储器 220 所存储的至少一个指令集。当计算设备 002 运行时,至少一个处理器 220 读取上述至少一个指令集,并且根据所述至少一个指令集的指示,执行本说明书提供的深度学习模型的元学习方法。处理器 220 可以执行深度学习模型的元学习方法包含的所有步骤或者部分步骤。

I/O 组件 240 支持计算设备 002 和其他组件之间的输入/输出。例如,通过 I/O 组件 240 输入用于接收输入的指令或数据,输入设备包括但不限于键盘、鼠标、摄像头、麦克风或触控设备。又例如,通过 I/O 组件 240 输出指令或数据,输出设备包括但不限于显示设备和扬声器等。在一个示例中,上述输出设备可以为显示屏,例如通过该显示屏显示通过本说明书方案所确定的模型参数。

通信端口 250 用于计算设备 002 同外界的数据通信,比如,通信端口 250 可以用于计算设备 002 同网络 260 之间的数据通信。通信端口 250 可以是有线通信端口也可以是无线通信端口。

除此之外,本领域技术人员可以理解,上述附图所示出的结构并不构成计算设备 002 的限定,计算设备 002 可以包括比图示更多或更少的部件,或者组合某些部件,或者不同的部件布置。比如,计算设备 002 中还包括射频电路、输入单元、传感器、音频电路、无线保真(Wireless Fidelity, WiFi)模块、电源、蓝牙模块等部件,在此不再赘述。

图 3 示出了根据本说明书的实施例提供的一种深度学习模型的元学习方法 P100 的流程示意图。如前所述,系统 001 可以执行本说明书的深度学习模型的元学习方法 P100。具体地,计算设备 002 可以读取存储在其本地存储介质中的指令集,然后根据指令集的规定,执行本说明书的深度学习模型的元学习方法 P100。如图 3 所示,深度学习模型的元学习方法 P100 可以包括:

S120: 获得训练数据集,上述训练数据集包括多个任务对应的训练样本。

在示例性的实施例中,以 3 个任务为例,假如任务 1(task 1)为广告 a 投放任务、任务 2(task 2)为广告 b 投放任务、任务 3(task 3)为广告 c 投放任务。其中,task 2 的样本量较少,则用于广告 b 投放预测的模型将面临冷启动问题。本说明书实施例中,将上述 task 1、task 2 以及 task 3 三种不同任务的样本作为训练样本,以对深度学习模型进行元学习。通过系统 001 在如上 task 1- task 3 一系列不同的任务上学习,以提取出任务间共性和模式。通过这种方式,模型可以学会一种“学习如何学习”的能力,这样在面对如 task 2 时,可以利用学到的知识和经验来更快地适应和解决问题。可见,元学习的目标是让深度学习模型学会如何在面对 task 2 时进行快速学习,从而可以解决用于广告 b 投放预测的模型将面临冷启动问题。

本说明书实施例中，将采用多个任务分别对应的训练样本确定元学习的训练数据集。

如图 3 所示，深度学习模型的元学习方法 P100 可以还包括：

S140：通过集群的 N 个处理节点并行地基于上述训练数据集对深度学习模型进行多次迭代训练，得到上述深度学习模型的参数，其中，在每次迭代训练中，上述 N 个处理节点中的每个处理节点使用上述训练数据集中的部分训练样本学习上述深度学习模型的部分参数，上述部分训练样本对应于上述多个任务中的同一任务。

在示例性的实施例中，本说明书提供的深度学习模型的元学习方法应用于包括 N 个处理节点的集群（例如系统 001）。具体地，获得包括多个任务分别对应的训练样本的训练数据集后，通过上述集群 001 中的 N 个处理节点（worker）并行地基于上述训练数据集，对深度学习模型进行多次迭代训练，得到该深度学习模型的元学习参数。其中，在每次迭代训练中，深度学习模型的参数被分布至集群的各个处理节点，每个处理节点使用训练数据集中的部分训练样本学习该深度学习模型的部分参数。

在示例性的实施例中，上述深度学习模型可以是 DLRM。图 4 示出了根据本说明书的实施例提供的一种深度推荐模型 003 的结构示意图。参考图 4，DLRM 包括：嵌入层 310、特征转换层 320、特征交互层 330 以及深度神经网络层 340。其中，嵌入层 310 是指将高维稀疏的离散特征转换为低维稠密的连续特征的一层网络结构。在推荐系统中，通常需要处理大规模的用户和物品特征，这些特征都是离散的，例如用户 ID、物品 ID 等。这些特征直接输入到神经网络中会导致网络参数过多，计算复杂度高，同时也会影响模型的泛化能力。通常的，嵌入层 310 的参数数量庞大，例如达到几十 GB，或者成百上千 GB。除了嵌入层 310 的其他网络结构部分，参数量通常较小，如几个 GB。本说明书实施例中，将嵌入层 310 的参数记作“嵌入层参数”，将 DLRM 除了嵌入层 310 之外结构的参数记作“密集层参数”。

本说明书一个示例性的实施例中，由于嵌入层参数量庞大，可以将上述嵌入层参数拆分至多个处理节点，从而每个处理节点将负责模型的部分嵌入层参数的学习。在其他示例性的实施例中，还可以将密集层参数拆分至多个处理节点，本说明书对此不做限定。

可以理解的是，上述深度学习模型还可以是 DLRM 之外的其他深度学习模型，本说明书对此不做限定。

图 3 所示的方法 P100 提供的深度学习模型的元学习方法中，一方面，每个处理节点使用部分训练样本来执行训练任务，从而实现训练数据并行训练；另一方面，每个处理节点负责学习模型的部分参数，从而实现模型并行训练。因此，本说明书实施例提供一种混合并行的元学习方式，节省了显存冗余，有利于提升深度学习模型的元学习效率。同时，还能够避免如上述相关技术中存在的、单个处理节点无法容纳待学习参数（如，DLRM 的嵌入层参数）的问题。

需要说明的是，在执行如图 3 所示的方法 P100 之前，需先执行数据预处理。

图 5 示出了根据本说明书的实施例提供的数据预处理的流程示意图。本说明书实施例中，每个样本包含其所属任务的任务标识（task_id）。参考图 5，样本 1-样本 4 分别携带其所属的任务标识“T1”，也就是说，样本 1-样本 4 均是关于任务 T1 的样本。同理，样本 5 携带有其所属的任务标识“T2”，说明样本 5 是关于任务 T2 的样本；样本 6 和样本 7 携带有其所属的任务标识“T3”，说明样本 6 和样本 7 是关于任务 T3 的样本。

在数据预处理阶段，确定 P 个任务分别对应的训练样本，其中每个训练样本包含其所属任务的任务标识，P 为大于 1 的整数。进一步地，根据预先确定的批大小，将上述 P 个任务分别对应的训练样本拆分为 Q 个数据批，得到训练数据集，其中属于同一个数据批的样本的任务标识一致，Q 为大于 P 的整数。

其中，每个处理节点在执行每次迭代训练的过程中，其所需要的数据为至少一个数据批。数据批的大小根据实际情况确定。因此，需将多个任务分别对应的训练样本划分为多个数据批。参考图 5，可以先将多个任务对应的训练样本按照任务标识 task_id 进行排序，然后根据预先确定的批大小，可以按照任务标识（task_id）进行排序的序列将样本拆分为四个数据批，且每个数据批可以同时携带有数据批标识（batch_id）和上述任务标识（task_id）。具体的，参考图 5 中第二列，数据批 1（batch 1）包括样本 1 和样本 2，数据批 2（batch 2）包括样本 3 和样本 4，数据批 3（batch 3）包括样本 5，数据批 4（batch 4）包括样本 6 和样本 7。可见，属于同一个数据批 batch 的样本来自同一任务。从而，在上述数据批被加载至处理节点后，可以保障该处理节点在一次迭代训练过程中所需的训练样本属于同一个任务。

示例性的，由于 ASR 场景中样本量大，可以将样本存储在基于 hdd 的文件系统中，例如 Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS），可以有效节省成本。另外，基于字符串的存储格式，存在解码耗时高的问题，从而在处理节点加载样本的过程中产生相当大的延迟。因此本说明书示例性的实施例中，将人工智能软件库 Tensorflow 自带的格式 TFRecords 作为存储格式，还可以和开源深度学习框架 PyTorch 中的数据加载库 WebDataset 作为存储格式来加速反序列化，能够减少数据传输，从而减小处理节点加载样本的过程中产生的延迟。

继续参考图 5，为了有效地加载大量 kb 级的小样本，在数据预处理阶段可以对所确定的训练数据集打乱顺序，得到待加载序列。在待加载序列中，各个样本按照序列标识（offset）顺序被存储。从而在处理节点加载训练数据时，样本可以按照(offset*i, offset*i + total_samples/N)被加载至每个 worker i。本说明书实施例提供的上述基于顺序进行读访问的方式，在基于块的文件系统中具有较高的 I/O 吞吐量。

在示例性的实施例中，在处理节点执行元学习训练阶段，每个 worker 可以根据任务标识 task_id 和数据批标识 batch_id，将来自相同任务的样本进行批量集成，从而每个 worker 均匀地

加载样本。例如，参考图 6，worker 1 加载获得数据批 2 和数据批 3，worker 2 加载获得数据批 4 和数据批 1。

通过如图 5 所示的数据预处理，可以将多个任务对应的训练样本转化为上述训练数据集，进一步地还可以确定待加载序列。其中训练数据集或待加载序列中包括多个数据批，且同一个数据批中的样本均来自同一个任务。从而，集群 001 中各个处理节点可以加载训练数据集中的数据批，以执行元学习训练阶段。

图 7 示出了根据本说明书的实施例提供的一种深度学习模型的元学习框架 004 的示意图。用于执行深度学习模型的元学习方法的集群可以包括多实体设备，如 410-430。示例性的，每个实体设备部署有至少一个处理节点（worker）。例如参考图 7 中示出的实体设备 410 部署有两个处理节点，分别为处理节点 i（worker i）和处理节点 j（worker j）。

在示例性的实施例中，上述处理节点可以采用 GPU，也可以采用 CPU。由于 GPU 能够上述分布式训练提供充足的计算性能以及充足的网络带宽，因此为了提升深度学习模型的元学习训练效率，本说明书实施例中以处理节点全部采用 GPU 为例进行说明。

在深度学习模型的元学习过程中，每个处理节点执行一次迭代训练包括先执行内循环训练过程以及再执行外循环训练过程。其中内外循环训练过程均需要两方面数据，一方面为此次迭代训练所需的训练数据，即数据批，另一方面为此次迭代训练中待学习的模型参数。具体地，参考图 7，以 worker i 为例，在一次迭代训练中的内循环训练过程中，一方面需要确定数据批 a 中的训练样本，另一方面需要确定待学习的模型参数： ξ_i^{Sup} 、 θ_i 。其中，确定此次迭代过程所需的训练数据（即数据批）的实施例已在上文中进行了详细介绍，在此不再赘述。接下来结合图 8 所提供的实施例介绍确定一次迭代过程待学习的模型参数的实施例。

图 8 示出了根据本说明书的实施例提供的一种深度学习模型的元学习方法 P200 的流程示意图。集群 001 中的任一处理节点可以执行本说明书的深度学习模型的元学习方法 P200。具体地，集群 001 中的任一处理节点可以读取存储在其本地存储介质中的指令集，然后根据指令集的规定，执行本说明书的深度学习模型的元学习方法 P200。参考图 8，方法 P200 包括：

在初始化阶段（即，在处理节点执行迭代训练之前），执行 S210：将嵌入层参数划分得到 N 个子集；以及，将第 i 个子集中的参数和密集层参数存储到第 i 个处理节点，其中，i 取值为小于或等于 N 的正整数。

本说明书提供的实施例中，由于嵌入层参数量庞大，因此集群 001 中任一处理节点可以将所有嵌入层参数划分为多个子集，从而每个 worker 将负责一个子集合对应的参数，即部分嵌入层参数。同时，在初始化阶段，由于密集层参数量较小，每个 worker 将存储密集层参数的完整拷贝和负责学习模型的全部密集层参数。

另外，在初始化阶段还设置内循环训练中的步长超参数 α ，以及外循环步长超参数 β ，以及下述公式中 N 仍然代表集群中处理节点的个数。

参考图 8，方法 P200 还包括：在处理节点执行一次迭代训练过程中，执行 S220-S260。

在 S220 中，确定目标数据批，并将目标数据批划分为支持集和查询集。

- 5 在示例性的实施例中，参考图 5 和图 6，训练数据集被打乱次序后，得到待加载序列。进一步地，待加载序列中的数据批被均匀地加载至各个处理节点。例如参考图 6，被加载至 worker 1 的样本为属于任务 1 的数据批 2 以及属于任务 2 的数据批 3。在 S220 中 worker 1 可以把属于任务 1 的数据批 2 确定为目标数据批，或者 worker 1 可以把属于任务 2 的数据批 3 确定为目标数据批。
- 10 进一步地，假如 worker 1 把属于任务 2 的数据批 3 确定为目标数据批，则进一步地将数据批 3 划分为两部分，一部分样本记作支持集(support set)，另一部分样本记作查询集(query set)。其中，支持集(support set)将用于此次迭代训练过程中的内循环训练，查询集(query set)将用于此次迭代训练过程中的外循环训练。以 worker i 为例进行说明，该处理接节点某次迭代循环对应的目标数据批 B_i ，进一步地，拆分为支持集 D_i^{Sup} 和查询集 D_i^{Query} 。具体地，worker i 可以将目标数据批 B_i 平均拆分，得到支持集 D_i^{Sup} 和查询集 D_i^{Query} 。还可以按照其他预设的拆分方式，
- 15 本说明书实施例对将目标数据批拆分为支持集和查询集的具体实施方式不做限定。

示例性的，参考图 7，worker i 所确定的目标数据批为数据批 a ，进一步地，将数据批 a 划分为支持集 $a1$ 以及查询集 $a2$ ；worker j 所确定的目标数据批为数据批 b ，进一步地，将数据批 b 划分为支持集 $b1$ 以及查询集 $b2$ 。

- 20 在 S230 中，通过与集群中的其他处理节点交互，获得支持集对应的当前嵌入层参数，并获得查询集对应的当前嵌入层参数。

- 以 worker i 为例进行说明，由于集群 001 中所有处理节点共同学习模型的嵌入层参数，因此 worker i 在当前次内循环训练过程中的支持集 D_i^{Sup} ，其所对应的嵌入层参数可能并不存在 worker i 节点。从而需要与集群 001 中其他处理节点进行通信，以获取其他处理节点所存储的嵌入层参数。示例性的，参考图 7，在进行上述数据交互之前，worker i 中的支持集 $a1$ 所对应的嵌入层参数可能并不存在 worker i 节点。从而需要与集群 001 中其他处理节点（包括图 7 中所示的处理节点 j ）进行通信，以获取其他处理节点所存储的嵌入层参数，如图 7 中所示的 ξ_j^{Sup} 。同理，参考图 7，worker j 中的支持集 $b1$ 所对应的嵌入层参数可能并不存在 worker j 节点。从而需要与集群 001 中其他处理节点（包括图 7 中所示的处理节点 i ）进行通信，以获取其他处理节点所存储的嵌入层参数，如图 7 中所示的 ξ_i^{Sup} 。
- 25
- 30

在示例性的实施例中，在处理节点采用 GPU 的情况下，为了充分利用 worker 之间的带宽，通过集合通信原语 AlltoAll 进行通信。具体地，执行 AlltoAll 集合通信原语来交换所有 worker

之间的嵌入层参数，以使 worker i 获取内循环过程所需学习的嵌入层参数 ξ_i^{Sup} ，同时，还为此次迭代训练中的外循环过程中的查询集预先获取其所需学习的嵌入层参数 ξ_i^{Query} 。本说明书实施例中，参考图 7，将执行一次集合通信原语 AlltoAll，为 worker i 同时获取内循环训练过程中支持集 D_i^{Sup} 对应的当前嵌入层参数 ξ_i^{Sup} ，以及在外循环训练过程中查询集 D_i^{Query} 对应的当前嵌入层参数 ξ_i^{Query} 。本说明书实施例中，将两个通信密集型的嵌入查找操作聚合在一起，能够有效降低通信频率。

在 S240 中，确定第一嵌入层参数，基于支持集、第一嵌入层参数和当前密集层参数确定内循环损失函数；以及，通过优化内循环损失函数对第一嵌入层参数和当前密集层参数进行更新，当前密集层参数被更新为中间密集层参数，当前第一嵌入层被更新为第三嵌入层参数。

其中，S240 所描述的训练过程也可以称为内循环训练过程。其中，第一嵌入层参数对应于嵌入层参数中的至少部分参数。在内循环训练过程中实现对第一嵌入层参数和密集层参数的更新。在一些实施例中，可以将支持集对应的当前嵌入层参数确定为第一嵌入层参数，例如图 7 中 worker i 所确定的第一嵌入层参数为 ξ_i^{Sup} ，worker j 所确定的第一嵌入层参数为 ξ_j^{Sup} 。在一些实施例中，上述当前密集层参数可以是 S210 中存储至该处理节点的密集层参数。例如图 7 中 worker i 所确定的当前密集层参数为 θ_i ，worker j 所确定的当前密集层参数为 θ_j 。可以理解的是， θ_i 和 θ_j 在更新前完全相同，仅为拷贝。

具体地，worker i 根据上述支持集 D_i^{Sup} 、上述支持集 D_i^{Sup} 对应的当前嵌入层参数 ξ_i^{Sup} 以及当前密集层参数 θ_i 进行前向计算，以确定内循环损失函数 L_i^{Sup} ，如公式(1)：

$$L_i^{\text{Sup}} \leftarrow L(f(\xi_i^{\text{Sup}}, \theta_i; D_i^{\text{Sup}})) \quad (1)$$

进一步地，对上述内循环损失函数 L_i^{Sup} 进行本地的反向梯度传递以对所述第一嵌入层参数和所述当前密集层参数进行更新。具体地，对上述支持集 D_i^{Sup} 对应的当前嵌入层参数 ξ_i^{Sup} （即上述第一嵌入层参数）进行更新得到第三嵌入层参数 $\xi_i'^{\text{Sup}}$ ，并对上述当前密集层参数 θ_i 进行更新得到所述中间密集层参数 θ_i' ，分别如公式(2)和公式(3)：

$$\xi_i'^{\text{Sup}} \leftarrow \xi_i^{\text{Sup}} \leftarrow \alpha \nabla_{\xi_i^{\text{Sup}}} L_i^{\text{Sup}} \quad (2)$$

$$\theta_i' \leftarrow \theta_i \leftarrow \alpha \nabla_{\theta_i} L_i^{\text{Sup}} \quad (3)$$

因此，worker i 经过上述内循环训练后得到与特定任务相关的模型参数（task-specific model parameters） $\xi_i'^{\text{Sup}}$ 和 θ_i' ，上述特征任务是指上述目标数据批 B_i 所属的任务。

进一步地，上述迭代训练过程还包括外循环训练：worker i 基于查询集 D_i^{Query} ，以及该查询集 D_i^{Query} 所需要的嵌入层参数和密集层参数进行本地的前向计算和后向计算，worker i 基于该训练数据进行全局的梯度更新，可以确定在此次迭代训练过程后深度学习模型更新后的参数，即

可以得到此次迭代训练后的模型元学习参数（meta parameters） $[\xi, \theta]$ 。参考图 8，上述外循环训练过程具体包括 S250 和 S260。

在 S250 中，确定第二嵌入层参数。

在深度学习中，梯度失效（stale gradients）问题指的是当网络的参数更新速度过慢时，信息同步更新不及时，导致梯度信息过时，从而影响网络的训练效果。本说明书实施例中，为了避免产生梯度失效问题，没有直接将在 S230 中查询集 D_i^{Query} 对应的当前嵌入层参数 ξ_i^{Query} ，确定为外循环训练所需更新的参数。而是先比较此次迭代训练的支持集 D_i^{Sup} 和查询集 D_i^{Query} 是否存在重叠元素。其中，上述元素可以指特征值，则两个样本之间具有重叠元素代表两个样本之间具有相同的特征值。如查询集 D_i^{Sup} 中的一个样本 S1 为“上海、女性、挎包”，查询集 D_i^{Query} 中的一个样本 S10 为“上海、男性、剃须刀”，则支持集 D_i^{Sup} 和查询集 D_i^{Query} 之间存在重叠元素“上海”，也就是说，查询集存在样本 S10 中特征值“上海”与支持集中样本 S1 中特征值“上海”重叠。在支持集 D_i^{Sup} 和查询集 D_i^{Query} 存在重叠元素的情况下：对于查询集的训练样本中的该重叠元素所对应的嵌入层参数，将在内循环训练过程中更新得到的第三嵌入层参数 $\xi_i'^{Sup}$ ，确定为在外循环训练过程中待更新的第二嵌入层参数 ξ_i^{Query} ；例如，将在内循环训练过程中更新后样本 S1 中特征值“上海”对应的第三嵌入层参数，确定为上述查询集中的样本 S10 中的特征值“上海”对应的第二嵌入层参数。通过该方式有利于外循环过程中参数更新速度，有利于及时更新同步信息，从而解决梯度失效（stale gradients）问题。同时，对于查询集的训练样本中非重叠元素所对应的嵌入层参数，将查询集对应的当前嵌入层参数确定为第二嵌入层参数；例如，上述查询集的样本 S10 中还包含特征值“剃须刀”，但是支持集的所有训练样本中均不包含该特征值，则上述查询集的样本 S10 所包含的特征值“剃须刀”，则采用通过 S230 中预取得的上述查询集对应的当前嵌入层。在支持集 D_i^{Sup} 和查询集 D_i^{Query} 不存在重叠元素的情况下，将查询集 D_i^{Query} 对应的当前嵌入层参数 ξ_i^{Query} （即 S230 中通过集合通信原语预先获取的）确定为第二嵌入层参数 ξ_i^{Query} 。

示例性的，参考图 7，在查询集 a2 与支持集 a1 存在重叠元素的情况下，则将 worker i 在内循环训练得到的第三嵌入层参数 $\xi_i'^{Sup}$ ，确定为外循环中待学习的第二嵌入层参数具体为 ξ_i^{Query} ，以避免梯度失效问题；在查询集 a2 与支持集 a1 不存在重叠元素的情况下，则将在 S230 中通过集合通信原语预先获取的当前嵌入层参数 ξ_i^{Query} 确定为第二嵌入层参数 ξ_i^{Query} 。同理，参考图 7，在查询集 b2 与支持集 b1 存在重叠元素的情况下，则将 worker j 在内循环训练得到的第三嵌入层参数 $\xi_j'^{Sup}$ ，确定为外循环中待学习的第二嵌入层参数具体为 ξ_j^{Query} ，以避免梯度失效问题；在查询集 b2 与支持集 b1 不存在重叠元素的情况下，则将在 S230 中通过集合通信原语预先获取的当前嵌入层参数 ξ_j^{Query} 确定为第二嵌入层参数 ξ_j^{Query} 。

在 S260 中，基于查询集、第二嵌入层参数和中间密集层参数确定外循环损失函数；以及，通过优化外循环损失函数来对第二嵌入层参数和中间密集层参数进行更新。

示例性的，在外循环训练中待学习的密集层参数为在内循环中学习得到的密集层参数。参考图 7，worker i 在外循环训练中待学习的密集层参数 θ'_i ，具体可以为其在内循环中学习得到的中间密集层参数 θ'_i ；同理，worker j 在外循环训练中待学习的密集层参数 θ'_j ，具体可以为其在内循环中学习得到的中间密集层参数 θ'_j 。

具体地，worker i 根据上述查询集 D_i^{Query} 、上述第二嵌入层参数 ξ_i^{Query} 和中间密集层参数 θ'_i 进行前向计算，以确定外循环损失函数 L_i^{Query} ，如公式(4)：

$$L_i^{Query} \leftarrow L(f(\xi_i^{Query}, \theta'_i; D_i^{Query})) \quad (4)$$

进一步地，通过 S1-S4 确定在此次迭代训练后深度学习模型或 DLRM 的嵌入层参数。S1：worker i 计算上述外循环损失函数 L_i^{Query} 关于嵌入层参数 ξ 的梯度，得到 worker i 对应的分嵌入层梯度，如公式(5)：

$$\nabla_{\xi} L_i^{Query} \quad (5)$$

S2：worker i 通过与集群中的其他处理节点交互，以获得集群中其他节点对应的分嵌入层梯度。

示例性的，在各个处理节点为 GPU 的情况下，参考图 7，可以通过第一集合通信原语 AlltoAll，与集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度。

S3：对所获得的分嵌入层梯度进行聚合计算，得到目标嵌入层梯度；以及 S4：根据目标嵌入层梯度更新所述第二嵌入层参数，得到第 i 处理节点在此次迭代训练后对应的嵌入层元学习参数。如公式(6)：

$$\xi \leftarrow \xi + \beta \sum_{i=1}^N \nabla_{\xi} L_i^{Query} \quad (6)$$

其中，集群的 N 个处理节点分别对应的嵌入层参数组成该深度学习模型或 DLRM 的嵌入层参数。可以理解的是，外循环训练后得到的嵌入层参数为模型中嵌入层的元学习参数。

接下来，通过 S1'-S4' 确定在此次迭代训练后深度学习模型或 DLRM 的密集层参数。S1'：worker i 计算上述外循环损失函数关于密集层参数的梯度，得到 worker i 对应的分密集层梯度；如公式(7)：

$$\nabla_{\theta} L_i^{Query} \quad (7)$$

S2'：worker i 通过与集群中的其他处理节点交互，以获得集群中其他节点对应的分密集层梯度。

示例性的，在各个处理节点为 GPU 的情况下，参考图 7，可以通过第二集合通信原语 AllReduce，与集群中的其他处理节点交互，以获得集群中其他节点对应的分密集层梯度。

S3': 对所获得的分密集层梯度进行聚合计算, 得到目标密集层梯度; 以及 S4': 根据所述目标密集层梯度更新所述中间密集层参数, 得到第 i 处理节点在此次迭代训练后对应的密集层元学习参数。如公式(8):

$$\theta \leftarrow \theta \leftarrow \beta \sum_{i=1}^N \nabla_{\theta} L_i^{\text{Query}} \quad (8)$$

其中, 集群的每个处理节点对应的密集层参数均为该深度学习模型或 DLRM 的嵌入层参数。可以理解的是, 外循环训练后得到的密集层参数为模型中密集层的元学习参数。

需要说明的是, 传统 MAML 算法对于外循环训练过程中对嵌入层参数以及密集层参数的更新方式分别如公式(9)和公式(10):

$$\xi \leftarrow \xi \leftarrow \beta \nabla_{\xi} \sum_{i=1}^N L_i^{\text{Query}} \quad (9)$$

$$\theta \leftarrow \theta \leftarrow \beta \nabla_{\theta} \sum_{i=1}^N L_i^{\text{Query}} \quad (10)$$

可见, 传统 MAML 算法需要一个中心节点来执行 $\nabla_{\theta} \sum_{i=1}^N L_i^{\text{Query}}$ 并更新元学习参数。本说明书实施例中, 可以根据导数规则, 将梯度和求和运算符的顺序互换, 得到上述公式(6)和公式(8)。从而, 本说明书实施例中 worker 可以在本地计算梯度, 然后通过集合通信原语将它们聚合到元学习参数中。因此, 相较于上述传统 MAML 算法, 本说明书实施例提供的方案在外循环训练过程中传输的数据量减少, 计算复杂度降低, 有利于提升深度学习模型的元学习效率。

继续参考图 8, worker i 通过执行 S220-S260 完成一次迭代训练, 通过执行 S270 判断是否需要继续迭代训练, 若需要则重复执行 S220-S260, 直至通过 S270 判断不再需要继续迭代训练, 则最后一次迭代训练所确定的 $[\xi, \theta]$ 即为深度推荐模型的元学习参数。

需要说明的是, 上述混合并行算法可以实现大型 Meta DLRM 的并行化训练, 由于集合通信原语 AlltoAll 和 AllReduce 是高度连接的通信模式, 而数据中心中基于套接字的网络会阻碍通信效率。因此, 为了避免网络阻塞而影响元学习计算速度, 本说明书实施例中在不同实体设备之间数据传输通过远程直接内存访问 (RemoteDirect Memory Access, RDMA) 进行数据传输, 以实现可扩展的高速通信。对于部署于同一实体设备内部处理节点之间的数据传输, 利用高速连接通道 NVLink 代替高速串行计算机扩展 (Peripheral Component Interconnect express, PCIe) 总线 (如, 系统内存), 从而可以获得更高的带宽。

相比于相关技术提供的分布式训练架构参数服务器 (Parameter Server), 本说明书实施例提供的方案提出的分布式通信拓扑不再使用 server 节点, 而是充分利用 worker 间节点的带宽, 使得其能高效地在 GPU 集群进行分布式训练。相比于相关技术中基于集合通信原语 AllReduce 并行化原 MAML 算法, 本说明书实施例提供的方案将大规模稀疏模型中的嵌入层参数切分到了不同的处理节点上, 解决了单个设备的 GPU 内存无法容纳的问题; 然后通过 AlltoAll 集合通信原语, 利用优化后网络 (NVLink 和 RDMA) 的高带宽进行集合通信, 保证了计算性能和通信效率。另外, 本说明书实施例提供的方案再每次迭代训练过程中, 将两个通信密集型的嵌入查找

(embedding lookup) 操作聚合在一起, 能够有效降低通信频率; 在一次迭代训练过程中, 若支持集与查询集存在元素重叠, 通过内循环训练所确定的嵌入层参数作为内循环训练待学习的嵌入层参数, 避免了梯度失效 (stale gradients) 影响模型效果的问题; 以及, 本说明书实施例提供的方案还优化了外循环训练过程中的更新公式, 使得通信量和计算复杂度大幅下降, 间接提高了训练速度。

综上所述, 本说明书实施例提供的深度学习模型的元学习方法以及深度学习模型的元学习系统, 一方面, 本说明书实施例提出了一套在 GPU 集群上通用和高效训练大规模分布式稀疏模型的元学习分布式算法和方案, 并首次引入了混合并行进入元学习的分布式训练中; 本说明书实施例的分布式算法针对元学习和稀疏的深度学习模型的特点, 提供了上述一系列针对性的计算效率和通信效率的优化, 比如分布式算法, 算子合并, 外循环优化, 网络优化等。另一方面, 训练效率的提高, 带来了训练数据和任务数量的提高, 使得训练得到的模型效果提升。例如, 相比于在 CPU 上使用参数服务器架构, 本说明书实施例提供的方案可以加快 22% 的训练速度, 节省 62.29% 的计算费用开销。又例如, 在某平台的主页展示广告中, 在线实验结果表明, 本说明书实施例提供的方案相较于相关技术, 基于使用 16 亿条样本, 模型交付时间从 3.7 小时减少到 1.2 小时, 训练后模型的转化率 (Conversion Rate, CVR) 和千次广告展现的费用 (Cost Per Mile, CPM) 分别提高了 6.48% 和 1.06%, 其中, 模型效果收益主要来自于支持更大的训练样本和更多的元学习任务。

本说明书另一方面提供一种非暂时性存储介质, 存储有至少一组用来进行信号处理的可执行指令。当所述可执行指令被处理器执行时, 所述可执行指令指导所述处理器实施本说明书所述的深度学习模型的元学习方法 P100 的步骤。在一些可能的实施方式中, 本说明书的各个方面还可以实现为一种程序产品的形式, 其包括程序代码。当所述程序产品在声学系统上运行时, 所述程序代码用于使声学系统执行本说明书描述的深度学习模型的元学习方法 P100 的步骤。用于实现上述方法的程序产品可以采用便携式紧凑盘只读存储器 (Compact Disc Read Only Memory, CD ROM) 包括程序代码, 并可以在声学系统上运行。然而, 本说明书的程序产品不限于此, 在本说明书中, 可读存储介质可以是任何包含或存储程序的有形介质, 该程序可以被指令执行系统使用或者与其结合使用。所述程序产品可以采用一个或多个可读介质的任意组合。可读介质可以是可读信号介质或者可读存储介质。可读存储介质例如可以为但不限于电、磁、光、电磁、红外线、或半导体的系统、装置或器件, 或者任意以上的组合。可读存储介质的更具体的例子包括: 具有一个或多个导线的电连接、便携式盘、硬盘、随机存取存储器 (Random Access Memory, RAM)、只读存储器 (Read-Only Memory, ROM)、可擦式可编程只读存储器 (Erasable Programmable ROM, EPROM) 或闪存)、光纤、便携式紧凑盘只读存储器 (CD ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。所述计算机可读存储介质可以包括在基带中或者作为载波一部分传播的数据信号, 其中承载了可读程序代码。这种传播的数据信号

可以采用多种形式，包括但不限于电磁信号、光信号或上述的任意合适的组合。可读存储介质还可以是可读存储介质以外的任何可读介质，该可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。可读存储介质上包含的程序代码可以用任何适当的介质传输，包括但不限于无线、有线、光缆、射频(Radio Frequency, RF)等等，或者上述的任意合适的组合。可以以一种或多种程序设计语言的任意组合来编写用于执行本说明书操作的程序代码，所述程序设计语言包括面向对象的程序设计语言—诸如 Java、C++等，还包括常规的过程式程序设计语言—诸如“C”语言或类似的设计语言。程序代码可以完全地在声学系统上执行、部分地在声学系统上执行、作为一个独立的软件包执行、部分在声学系统上部分在远程计算设备上执行、或者完全在远程计算设备上执行。

上述对本说明书特定实施例进行了描述。其他实施例在所附权利要求书的范围内。在一些情况下，在权利要求书中记载的动作或步骤可以按照不同于实施例中的顺序来执行并且仍然可以实现期望的结果。另外，在附图中描绘的过程不一定要求示出特定顺序或者连续顺序才能实现期望的结果。在某些实施方式中，多任务处理和并行处理也是可以的或者是可能有利的。

综上所述，在阅读本详细公开内容之后，本领域技术人员可以明白，前述详细公开内容可以仅以示例的方式呈现，并且可以不是限制性的。尽管这里没有明确说明，本领域技术人员可以理解本说明书需求囊括对实施例的各种合理改变，改进和修改。这些改变，改进和修改旨在由本说明书提出，并且在本说明书的示例性实施例的精神和范围内。

此外，本说明书中的某些术语已被用于描述本说明书的实施例。例如，“一个实施例”，“实施例”和/或“一些实施例”意味着结合该实施例描述的特定特征，结构或特性可以包括在本说明书的至少一个实施例中。因此，可以强调并且应当理解，在本说明书的各个部分中对“实施例”或“一个实施例”或“替代实施例”的两个或更多个引用不一定都指代相同的实施例。此外，特定特征，结构或特性可以在本说明书的一个或多个实施例中适当地组合。

应当理解，在本说明书的实施例的前述描述中，为了帮助理解一个特征，出于简化本说明书的目的，本说明书将各种特征组合在单个实施例、附图或其描述中。然而，这并不是说这些特征的组合是必须的，本领域技术人员在阅读本说明书的时候完全有可能将其中一部分设备标注出来作为单独的实施例来理解。也就是说，本说明书中的实施例也可以理解为多个次级实施例的整合。而每个次级实施例的内容在于少于单个前述公开实施例的所有特征的时候也是成立的。

本文引用的每个专利，专利申请，专利申请的出版物和其他材料，例如文章，书籍，说明书，出版物，文件，物品等，除了与其相关的任何历史起诉文件、可能与本文件不一致或相冲突的任何相同的、或者任何可能对权利要求的最宽范围具有限制性影响的任何相同的历史起诉文件，均可以通过引用结合于此，并用于现在或以后与本文件相关联的所有目的。此外，如果

在与任何所包含的材料相关联的术语的描述、定义和/或使用与本文档相关的术语、描述、定义和/或之间存在任何不一致或冲突时，使用本文件中的术语为准。

最后，应理解，本文公开的申请的实施方案是对本说明书的实施方案的原理的说明。其他修改后的实施例也在本说明书的范围内。因此，本说明书披露的实施例仅作为示例而非限制。

- 5 本领域技术人员可以根据本说明书中的实施例采取替代配置来实现本说明书中的申请。因此，本说明书的实施例不限于申请中被精确地描述过的实施例。

权 利 要 求 书

1. 一种深度学习模型的元学习方法，其中，所述方法应用于包括 N 个处理节点的集群，所述 N 为大于 1 的整数，其中，所述方法包括：

获得训练数据集，所述训练数据集包括多个任务对应的训练样本；以及

5 通过所述 N 个处理节点并行地基于所述训练数据集对所述深度学习模型进行多次迭代训练，得到所述深度学习模型的参数，其中，在每次迭代训练中，所述 N 个处理节点中的每个处理节点使用所述训练数据集中的部分训练样本学习所述深度学习模型的部分参数，所述部分训练样本对应于所述多个任务中的同一任务。

10 2. 根据权利要求 1 所述的方法，其中，所述深度学习模型的参数包括：嵌入层参数和密集层参数，

所述每个处理节点使用所述部分训练样本学习所述嵌入层参数中的部分参数以及所述密集层参数中的全部参数。

15 3. 根据权利要求 2 所述的方法，其中，所述训练数据集包括多个数据批；所述每次迭代训练的过程包括通过第 i 个处理节点：

确定目标数据批，并将所述目标数据批划分为支持集和查询集，所述目标数据批包括所述部分训练样本，所述目标数据批为所述多个数据批中的一个；

20 执行内循环训练过程，所述内循环训练过程包括：确定第一嵌入层参数，基于所述支持集、所述第一嵌入层参数和当前密集层参数确定内循环损失函数，以及，通过优化所述内循环损失函数对所述第一嵌入层参数和所述当前密集层参数进行更新，其中，所述当前密集层参数被更新为中间密集层参数；以及

25 执行外循环训练过程，所述外循环训练过程包括：确定第二嵌入层参数，基于所述查询集、所述第二嵌入层参数和所述中间密集层参数确定外循环损失函数，以及，通过优化所述外循环损失函数来对所述第二嵌入层参数和所述中间密集层参数进行更新；

其中，i 取值为小于或等于 N 的正整数，所述第一嵌入层参数和所述第二嵌入层参数分别对应所述嵌入层参数中的至少部分参数。

4. 根据权利要求 3 所述的方法，其中，

30 在所述执行内循环训练过程之前，所述方法还包括：通过与所述集群中的其他处理节点交互，获得所述支持集对应的当前嵌入层参数，并获得所述查询集对应的当前嵌入层参数；

- 所述确定第一嵌入层参数，包括：将所述支持集对应的当前嵌入层参数确定为所述第一嵌入层参数；其中，所述第一嵌入层参数在所述内循环训练过程之后被更新为第三嵌入层参数；以及
- 所述确定第二嵌入层参数，包括：根据所述查询集对应的当前嵌入层参数或所述第三嵌入层参数确定所述第二嵌入层参数。
- 5 5. 根据权利要求 4 所述的方法，其中，所述根据所述查询集对应的当前嵌入层参数或所述第三嵌入层参数确定所述第二嵌入层参数，包括：
- 10 在所述查询集与所述支持集存在重叠元素的情况下，对于所述查询集的训练样本中所述重叠元素对应的嵌入层参数，将所述第三嵌入层参数确定为所述第二嵌入层参数；对于所述查询集的训练样本中除了所述重叠元素之外的元素所对应的嵌入层参数，将所述查询集对应的当前嵌入层参数确定为所述第二嵌入层参数；或者
- 在所述查询集与所述支持集不存在重叠元素的情况下，将所述查询集对应的当前嵌入层参数确定为所述第二嵌入层参数。
- 15 6. 根据权利要求 3 所述的方法，其中，所述基于所述支持集、所述第一嵌入层参数和当前密集层参数确定内循环损失函数，包括：
- 根据所述支持集、所述第一嵌入层参数和所述当前密集层参数进行前向计算，以确定所述内循环损失函数；
- 20 所述通过优化所述内循环损失函数对所述第一嵌入层参数和所述当前密集层参数进行更新，包括：
- 对所述内循环损失函数进行本地的反向梯度传递以对所述第一嵌入层参数和所述当前密集层参数进行更新。
- 25 7. 根据权利要求 3 所述的方法，其中，所述通过优化所述外循环损失函数来对所述第二嵌入层参数进行更新，包括：
- 确定所述外循环损失函数关于嵌入层参数的梯度，得到所述第 i 个处理节点对应的分嵌入层梯度；
- 通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度；
- 30 对所述 N 个处理节点对应的分嵌入层梯度进行聚合计算，得到目标嵌入层梯度；以及
- 根据所述目标嵌入层梯度对所述第二嵌入层参数进行更新。
8. 根据权利要求 7 所述的方法，其中，所述处理节点为图像处理单元 GPU 节点；

所述通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度，包括：

通过第一集合通信原语 AlltoAll，与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分嵌入层梯度。

5

9. 根据权利要求 3 所述的方法，其中，所述通过优化所述外循环损失函数来对所述中间密集层参数进行更新，包括：

确定所述外循环损失函数关于密集层参数的梯度，得到所述第 i 个处理节点对应的分密集层梯度；

10 通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度；
对所述 N 个处理节点对应的分密集层梯度进行聚合计算，得到目标密集层梯度；以及
根据所述目标密集层梯度对所述中间密集层参数进行更新。

10. 根据权利要求 9 所述的方法，其中，所述处理节点为图像处理单元 GPU 节点；

15 所述通过与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度，包括：

通过第二集合通信原语 AllReduce，与所述集群中的其他处理节点交互，以获得所述集群中其他节点对应的分密集层梯度。

20 11. 根据权利要求 2 所述的方法，其中，在所述通过所述 N 个处理节点并行地基于所述训练数据集对所述深度学习模型进行多次迭代训练之前，所述方法还包括：

将所述嵌入层参数划分得到 N 个子集；以及

将第 i 个子集中的参数和所述密集层参数存储到第 i 个处理节点，其中， i 取值为小于或等于 N 的正整数。

25

12. 根据权利要求 1 所述的方法，其中，所述训练数据集包括多个数据批；所述获得训练数据集，包括：

确定 P 个任务分别对应的训练样本，其中每个训练样本包含其所属任务的任务标识， P 为大于 1 的整数；以及

30 根据预先确定的批大小，将所述 P 个任务分别对应的训练样本拆分为 Q 个数据批，得到所述训练数据集，其中属于同一个数据批的训练样本的任务标识一致， Q 为大于 P 的整数。

13. 根据权利要求 1 所述的方法，其中，所述处理节点为图像处理单元 GPU 节点，所述 N 个处理节点被部署于多个实体设备上，其中，在所述多次迭代训练过程中，同一实体设备内部的不同处理节点之间采用高速连接通道 NVLink 进行通信，不同实体设备之间通过远程直接内存访问 RDMA 进行通信。

5

14. 根据权利要求 1 所述的方法，其中，所述深度学习模型为深度推荐模型 DLRM。

15. 一种深度学习模型的元学习系统，包括：N 个处理节点，所述 N 为大于 1 的整数，
每个处理节点包括：

10 至少一个存储介质，存储有至少一个指令集，用于进行深度学习模型的元学习；

 以及

 至少一个处理器，同所述至少一个存储介质通信连接，

 其中，当所述深度学习模型的元学习系统运行时，每个处理节点中的所述至少一个处理器
读取并执行所述至少一个指令集，以实现如权利要求 1 至 14 中任意一项所述的深度学习模型的

15 元学习方法。