(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114880116 A
(43)申请公布日 2022.08.09

(21)申请号 202210459355.1

(22)申请日 2022.04.27

(71)申请人 远光软件股份有限公司
地址 519000 广东省珠海市港湾大道科技一路3号

(72)发明人 雷太原 黄剑 吴磊 李美平

(74)专利代理机构 深圳市威世博知识产权代理事务所(普通合伙) 44280
专利代理师 杨新星

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/445(2018.01)
G06F 9/455(2006.01)

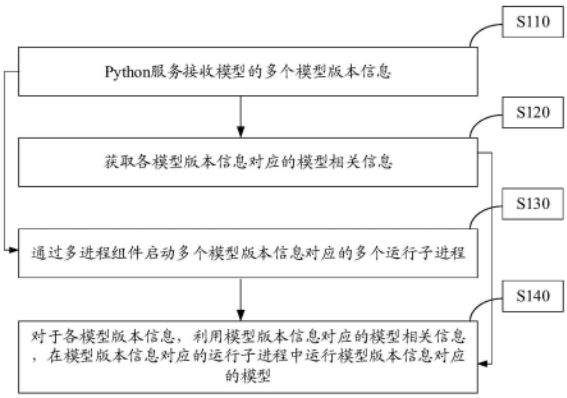
权利要求书2页 说明书8页 附图6页

(54)发明名称

一种多版本运行方法、电子设备和存储介质

(57)摘要

本申请公开了一种多版本运行方法、电子设备和存储介质，该方法包括：Python服务接收模型的多个模型版本信息；获取各模型版本信息对应的模型相关信息；通过多进程组件启动多个模型版本信息对应的多个运行子进程；对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。通过上述方式，本申请能够在同一个Python服务中运行不同版本模型。



CN 114880116 A

1.一种多版本运行的方法,其特征在于,所述方法包括:

Python服务接收模型的多个模型版本信息;

获取各所述模型版本信息对应的模型相关信息;

通过多进程组件启动多个模型版本信息对应的多个运行子进程;

对于各所述模型版本信息,利用所述模型版本信息对应的模型相关信息,在所述模型版本信息对应的运行子进程中运行所述模型版本信息对应的所述模型。

2.根据权利要求1所述的方法,其特征在于,所述获取各所述模型版本信息对应的模型相关信息,包括:

从各所述模型版本信息对应的沙盒目录中,获取各所述模型版本信息对应的模型相关信息;

所述通过多进程组件启动多个模型版本信息对应的多个运行子进程,包括:

通过多进程组件在各所述模型版本信息对应的沙盒目录中,启动各所述模型版本信息对应的运行子进程。

3.根据权利要求1所述的方法,其特征在于,所述模型相关信息包括:模型的安装文件以及模型运行时依赖的第三方库的信息;

所述在所述模型版本信息对应的运行子进程中运行所述模型版本信息对应的所述模型之前,还包括:

利用所述模型的安装文件在所述模型版本信息对应的运行子进程中安装所述模型;

利用所述模型依赖的第三方库的信息,在所述模型版本信息对应的运行子进程中加载所述模型运行时依赖的第三方库。

4.根据权利要求2所述的方法,其特征在于,所述模型相关信息还包括:模型对应的第一Python版本信息;

所述通过多进程组件在各所述模型版本信息对应的沙盒目录中,启动各所述模型版本信息对应的运行子进程,包括:

基于各所述模型对应的第一Python版本信息获取对应版本的各第一执行器;

在所述沙盒目录中采用所述各第一执行器启动各所述模型版本信息对应的各所述运行子进程。

5.根据权利要求1所述的方法,其特征在于,在所述Python服务接收模型的多个模型版本信息之前,所述方法还包括下列步骤以安装所述多个模型版本信息对应的多个模型和所述多个模型依赖的第三方库;

采用各第二执行器通过所述多进程组件启动多个模型版本信息对应的多个安装子进程;

在所述多个安装子进程中创建沙盒目录,所述沙盒目录用于存储各所述模型版本信息对应的模型相关信息。

6.根据权利要求5所述的方法,其特征在于,所述模型相关信息还包括:模型对应的第一Python版本信息;

在所述采用各第二执行器通过所述多进程组件启动多个模型版本信息对应的多个安装子进程之前,还包括:

判断各所述模型对应的第一Python版本与所述Python服务对应的第二Python版本是

否一致；

若一致，将所述Python服务的第三执行器作为所述各第二执行器；

若不一致，将与所述第一Python版本对应的各第一执行器作为所述各第二执行器。

7.根据权利要求6所述的方法，其特征在于，所述将与所述第一Python版本对应的各第一执行器作为所述各第二执行器，包括：

将所述各第二执行器的可执行文件的路径，由第三执行器的可执行文件的路径更改为所述各第一执行器的可执行文件的路径。

8.根据权利要求1所述的方法，其特征在于，所述在所述模型版本信息对应的运行子进程中运行所述模型版本信息对应的所述模型之后，还包括：

将模型运行结果和/或所述运行子进程运行时产生的异常信息，发送给所述Python服务。

9.一种电子设备，其特征在于，包括相互耦接的存储器和处理器，

所述存储器存储有程序指令；

所述处理器用于执行所述存储器中存储的程序指令，以实现权利要求1-8任一项所述的方法。

10.一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用于存储程序指令，所述程序指令能够被执行以实现如权利要求1-8任一项所述的方法。

一种多版本运行方法、电子设备和存储介质

技术领域

[0001] 本申请涉及软件服务管理技术领域，特别是涉及一种多版本运行方法、电子设备和存储介质。

背景技术

[0002] 现有技术中，运行多版本的模型主要是采用虚拟环境或docker容器来解决版本冲突的问题。例如，基于虚拟环境部署多个Python服务，每个Python服务对应一个虚拟环境，利用多个服务提供不同版本的运行环境。然而，采用上述方式时，需要部署的服务数量多，要提前安装好虚拟环境，根据模型及依赖库版本的不同需要人工增加服务节点等问题。多个不同版本的Python虚拟环境，难以在同一个Python服务中同时运行，无法解决不同版本模型在同一个Python服务中同时运行的问题。

发明内容

[0003] 本申请主要解决的技术问题是提供一种多版本运行方法、电子设备和存储介质，能够在同一个Python服务中运行不同版本模型。

[0004] 为解决上述技术问题，本申请第一方面提供了一种多版本运行的方法，该方法包括：Python服务接收模型的多个模型版本信息；获取各模型版本信息对应的模型相关信息；通过多进程组件启动多个模型版本信息对应的多个运行子进程；对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0005] 为解决上述技术问题，本申请第二方面提供了一种电子设备，该设备包括：该设备包括相互耦接的存储器和处理器，存储器存储有程序指令，处理器用于执行存储器中存储的程序指令，以实现上述第一方面所述的方法。

[0006] 为解决上述技术问题，本申请第三方面提供了一种计算机可读存储介质，计算机可读存储介质用于存储程序指令，程序指令能够被执行以实现上述第一方面所述的方法。

[0007] 本申请的有益效果是：区别于现有技术的情况，本申请在Python服务接收模型的多个模型版本信息后，根据多个模型版本信息获取对应的模型相关信息，并利用多进程组件启动多个模型版本信息对应的运行子进程，利用多个模型版本信息对应的模型相关信息，在运行子进程中运行模型版本信息对应的模型。由于在Python服务中根据多个模型版本信息建立了多个运行子进程，在运行子进程中运行模型版本信息对应的模型，故可实现在Python服务中运行多个不同版本的模型。

附图说明

[0008] 图1是本申请提供的多版本运行方法第一实施方式的流程示意图；

[0009] 图2是本申请提供的多版本运行方法第二实施方式的流程示意图；

[0010] 图3是本申请提供的多版本运行方法第三实施方式的流程示意图；

[0011] 图4是本申请提供的多版本运行方法第四实施方式的流程示意图；

[0012] 图5是本申请提供的多版本运行方法第五实施方式的流程示意图；

[0013] 图6是本申请提供的多版本运行方法的简化示意图；

[0014] 图7是本申请提供的电子设备的框架结构示意图；

[0015] 图8是本申请计算机可读存储介质一实施方式的框架示意图。

具体实施方式

[0016] 下面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清楚、完整地描述，显然，所描述的实施例仅仅是本申请一部分实施例，而不是全部实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性的劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

[0017] 需要说明的是，本申请实施例中有涉及“第一”、“第二”等的描述，该“第一”、“第二”等的描述仅用于描述目的，而不能理解为指示或暗示其相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或者隐含地包括至少一个该特征。

[0018] 在本文中提及“实施例”意味着，结合实施例描述的特定特征、结构或特性可以包含在本发明的至少一个实施例中。在说明书中的各个位置出现该短语并不一定均是指相同的实施例，也不是与其它实施例互斥的独立的或备选的实施例。本领域技术人员显式地和隐式地理解的是，本文所描述的实施例可以与其它实施例相结合。

[0019] 参阅图1，图1是本申请提供的多版本运行方法第一实施方式的流程示意图，该方法包括：

[0020] S110：Python服务接收模型的多个模型版本信息。

[0021] 在一实施方式中，Python服务指的是用Python开发的Web、TCP网络服务应用，或者单体应用(桌面应用或命令行应用)，用来安装和运行模型。模型指的是用Python开发的算法模型、机器学习模型、深度学习模型等。可以理解的是，本申请实施例的执行主体可以为任何安装有Python服务的设备。

[0022] Python服务运行模型时，Python服务作为主进程(对用户提供入口的进程)，主进程接收多个模型计算请求，模型计算请求可以包含模型名称(如车牌识别模型、深度学习模型等)、模型的版本信息(如车牌识别模型1.0、车牌识别模型2.0等)、以及模型计算时所需的数据(如包含车牌的图片，车牌识别模型可识别出该图片中的车牌号码)。其中，模型计算请求可有用户发出，也可由系统自动发出，在此不作限定。

[0023] S120：获取各模型版本信息对应的模型相关信息。

[0024] 模型相关信息可以包含：模型的安装文件、模型对应的第一Python版本信息、以及模型依赖的第三方库的信息。其中，模型对应的第一Python版本信息可以与开发模型的Python版本信息一致，例如，若模型在Python2.7中开发，则模型对应的第一Python版本信息即为Python2.7，模型对应的第一Python版本信息也可以为模型开发时指定的Python版本信息；模型依赖的第三方库指非Python发行版提供的默认类库，一般通过pip(通用的Python包管理工具)或setuptools(用于创建和发布Python包)来安装，第三方库可以为TensorFlow1和TensorFlow2。

[0025] 在一实施方式中，用户可以根据模型的版本信息创建不同的安装目录(如沙盒目录)，将各模型版本信息对应的模型相关信息存储在不同的安装目录下，方便用户获取指定版本的模型的模型相关信息。

[0026] S130:通过多进程组件启动多个模型版本信息对应的多个运行子进程。

[0027] 在一实施方式中，将Python服务作为主进程，通过多进程组件(multiprocessing)启动多个模型版本信息对应的多个运行子进程，使得在每个子进程中运行一个模型，其中，运行子进程是在主进程中启动的一个新的进程。

[0028] 在本申请的实施例中，上述步骤S120和S130的执行顺序可以随意更换，即可以先执行步骤S120，也可以先执行步骤S130。

[0029] S140:对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0030] 在步骤S130启动子进程后，多进程组件将各模型版本信息对应的模型相关信息传给各模型版本信息对应的运行子进程，使得运行子进程中运行模型版本信息对应的模型。例如，模型版本信息包括车辆识别模型1.0和车辆识别模型2.0，获取车辆识别模型1.0和车辆识别模型2.0分别对应的模型相关信息，通过多进程组件启动运行子进程1和运行子进程2，将车辆识别模型1.0的模型相关信息传输给运行子进程1，在运行子进程1中运行车辆识别模型1.0，以及将车辆识别模型2.0的模型相关信息传输给运行子进程2，在运行子进程2中运行车辆识别模型2.0。其中，多进程组件将各模型版本信息对应的模型相关信息传给各模型版本信息对应的运行子进程时，可以通过管道、文件或TCP/IP协议来传递数据，以实现主进程与子进程间的数据共享。

[0031] 本实施方式中，Python服务接收到多个模型的版本信息后，根据模型的版本信息，找到对应的模型相关信息，并通过多进程组件启动多个模型的版本信息对应的运行子进程，将多个模型的版本信息对应的模型相关信息传输给多个模型的版本信息对应的运行子进程，在运行子进程中运行模型版本信息对应的模型，实现在Python服务中运行不同版本的模型。

[0032] 请参阅图2，图2是本申请提供的多版本运行方法第二实施方式的流程示意图，该方法包括：

[0033] S210:Python服务接收模型的多个模型版本信息。

[0034] 在一实施方式中，模型版本信息包含模型名称和模型版本，如车牌识别模型1.0、车牌识别模型2.0。

[0035] S220:从各模型版本信息对应的沙盒目录中，获取各模型版本信息对应的模型相关信息。

[0036] 其中，沙盒目录可以是模型及其第三方库的安装和运行目录，此沙盒目录不包括Python可执行程序、基础库和工具脚本等，不等于用python-m venv venv命令创建的虚拟环境。

[0037] 在开发模型时，用户会指定模型的版本、其对应的第一Python版本、以及其依赖的第三方库的信息。发布模型时，用户会提供模型的配置(service.yml)以及第三方库依赖文件(requirements.txt)，其中，模型配置中记录了模型相关信息(如模型名称、模型版本、模型安装文件等)和模型对应的第一Python版本(如Python2.7)，第三方库依赖文件中记录了

当前版本模型所依赖的第三方依赖库的信息。在将模型部署到Python服务时，系统会根据模型的版本信息，自动创建不同的沙盒目录，将模型相关信息以及其依赖的第三方库的信息均存储在沙盒目录中，以实现不同版本的模型、第三方库之间的隔离。用户需获取某个版本的模型的模型相关信息时，只需按照模型版本信息获取对应的沙盒目录即可获取对应的模型相关信息。

[0038] S230：通过多进程组件在各模型版本信息对应的沙盒目录中，启动各模型版本信息对应的运行子进程。

[0039] 具体地，需要运行某个版本的模型(如车牌识别模型2.0)时，Python服务作为主进程，在车牌识别模型2.0对应的沙盒目录中通过多进程组件启动运行子进程。

[0040] 在一实施方式中，运行子进程需要采用与模型对应的第一Python版本信息对应的第一执行器启动。因此，在启动运行子进程之前需根据模型对应的第一Python版本信息获取对应版本的第一执行器，采用第一执行器在沙盒目录中启动运行子进程。例如，用户需要运行车牌识别模型2.0，其对应的第一Python版本信息为Python2.7，则需要获取Python2.7版本的第一执行器，才可以启动车牌识别模型2.0的运行子进程。

[0041] S240：对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0042] 步骤S230启动运行子进程后，可以根据沙盒目录中存储的模型相关信息以及模型依赖的第三方库的信息，在运行子进程中加载模型以及模型依赖的第三方库。在一实施方式中，可以在运行子进程中通过site模块或sys.path模块来动态加载模型依赖的第三方库及模型，运行子进程执行模型接口来运行模型进行模型计算。

[0043] 可以理解地，本实施方式可以同时在多个沙盒目录中启动多个运行子进程，并在运行子进程中运行多个不同版本的模型。在本申请的实施例中，上述步骤S220和S230的执行顺序可以随意更换，即可以先执行步骤S220，也可以先执行步骤S230。

[0044] 本实施方式可使不同版本的模型及第三方依赖库在同一个Python服务中共存，它们通过沙盒目录隔离且互不影响，再通过多进程组件启动运行子进程，在运行子进程中加载沙盒目录中的第三方库及模型，执行模型计算并返回结果，运行子进程可以与主进程共享数据，但它们各自独立运行互不干扰。

[0045] 请参阅图3，图3是本申请提供的多版本运行方法第三实施方式的流程示意图，该方法包括：

[0046] S310：Python服务接收模型的多个模型版本信息。

[0047] S320：获取各模型版本信息对应的模型相关信息，其中，模型相关信息包括：模型的安装文件以及模型运行时依赖的第三方库的信息。

[0048] S330：通过多进程组件启动多个模型版本信息对应的多个运行子进程。

[0049] S340：利用模型的安装文件在模型版本信息对应的运行子进程中安装模型。

[0050] S350：利用模型依赖的第三方库的信息，在模型版本信息对应的运行子进程中加载模型运行时依赖的第三方库。

[0051] S370：对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0052] 本实施例中，模型版本信息对应的模型相关信息包括模型的安装文件以及模型依

赖的第三方库的信息，系统可以在启动各运行子进程之前或之后，获取各模型版本信息对应的模型相关信息。

[0053] Python服务启动运行子进程后，利用模型相关信息中的模型的安装文件在运行子进程中安装模型，进一步地，为了避免模型在运行过程中出现报错的现象，可以利用模型依赖的第三方库的信息，在运行子进程中加载模型运行时依赖的第三方库，在模型和第三方库加载完成后，即可自动运行模型。可以理解地，系统可以在运行子进程中先安装模型再加载第三方库，也可以在运行子进程中先加载第三方库，再安装模型。在各运行子进程中安装好对应的模型并加载模型依赖的第三方库后，即可利用模型相关信息在运行子进程中运行对应的模型。

[0054] 可以理解地，本申请同样可以启动多个运行子进程，分别在各个运行子进程中加载相关的模型及其依赖的第三方库，使得系统可以自动运行多个不同版本的模型。

[0055] 请参阅图4，图4是本申请提供的多版本运行方法第四实施方式的流程示意图，该方法包括：

[0056] S410：判断各模型对应的第一Python版本与Python服务对应的第二Python版本是否一致；若一致，将Python服务的第三执行器作为各第二执行器；若不一致，将与第一Python版本对应的各第一执行器作为各第二执行器。

[0057] S420：采用各第二执行器通过多进程组件启动多个模型版本信息对应的多个安装子进程。

[0058] S430：在多个安装子进程中创建沙盒目录，沙盒目录用于存储各模型版本信息对应的模型相关信息。

[0059] 在一实施方式中，用户需要运行多个不同版本的模型，则需要先准备用于不同版本的模型运行的运行环境。在一具体实施方式中，用户可以先在服务器上安装多个不同版本的Python，如Python2.7、Python3.6、Python3.7等，不同版本的Python表示Python语言的运行环境，用于执行在不同开发环境中开发的不同版本的Python代码，每个运行环境包含可能包含不同的类库，在系统中安装多个版本的Python后，用户即可以在计算机的任何目录下访问任意版本的Python。

[0060] 在另一具体实施方式中，用户还可以为Python服务安装多个不同版本的虚拟环境，相同版本的虚拟环境是可以共享共用的。然而，为了避免虚拟环境的冗余，模型依赖的第三方库不会安装在虚拟环境中。

[0061] 进一步地，将各版本的模型以及其依赖的第三方库安装到Python服务中，在安装过程中，用户为了隔离不同版本的模型，利用多进程组件启动多个安装子进程，在多个安装子进程中分别创建沙盒目录，将模型相关信息存储在沙盒目录中。

[0062] 在一实施方式中，Python服务作为主进程，利用多进程组件启动多个安装子进程。以启动单个安装子进程为例，多进程组件默认用主进程的执行器作为安装子进程的执行器启动安装子进程，因此在启动安装子进程之前，需要修改安装子进程的执行器来实现多版本的Python的隔离。修改安装子进程的执行器时，先判断模型对应的第一Python版本与Python服务对应的第二Python版本是否一致，若一致，将Python服务的第三执行器作为第二执行器，若不一致，将与第一Python版本对应的第一执行器作为第二执行器，采用第二执行器启动安装子进程。例如，Python服务对应的第二Python版本为Python2.7，模型对应的

第一Python版本也为Python2.7，则将Python服务的第三执行器作为第二执行器，以启动安装子进程；Python服务对应的第二Python版本为Python2.7，模型对应的第一Python版本也为Python3.7，将与第一Python版本对应的第一执行器作为第二执行器，来启动安装子进程。本领域公知的，执行器的版本与Python版本一致。其中，将与第一Python版本对应的各第一执行器作为各第二执行器，包括：将各第二执行器的可执行文件的路径，由第三执行器的可执行文件的路径更改为各第一执行器的可执行文件的路径。可以理解地，若用于启动多个安装子进程，针对每个安装子进程，均需进行上述判断，并依据判断执行响应的操作。具体步骤请参见上述说明，在此不再赘述。

[0063] Python服务启动多个安装子进程后，可以在每个安装子进程中创建沙盒目录，根据模型相关信息，将模型以及模型依赖的第三方库安装在沙盒目录中，也就是说，将模型的安装文件和模型依赖的第三方库的文件存储在沙盒目录中。例如，针对汽车识别模型1.0，汽车识别模型1.0对应的第一Python版本为Python3.7，Python服务对应的第二Python版本为Python2.7，将与第一Python版本对应的第一执行器作为第二执行器启动安装子进程1后，在安装子进程1中创建对应的沙盒目录，将汽车识别模型1.0及其依赖的第三方库安装在沙盒目录中。同理，针对汽车识别模型2.0，启动其对应的安装子进程2后，在安装子进程2创建对应的沙盒目录，用以安装汽车识别模型2.0及其依赖的第三方库。

[0064] 进一步地，对于一些常用的第三方库可以安装在通用的沙盒目录，如scikit-learn安装在/data/sandbox/scikit-learn，TensorFlow 1安装在/data/sandbox/tf_1，TensorFlow 2安装在/data/sandbox/tf_2，一些用到这些常用第三方库的模型，在启动运行子进程的时候动态加载这些通用的第三方库，可以减少模型之间通用第三方库的重复安装，减少第三方库的冗余。

[0065] S440：Python服务接收模型的多个模型版本信息。

[0066] S450：获取各模型版本信息对应的模型相关信息。

[0067] S460：通过多进程组件启动多个模型版本信息对应的多个运行子进程。

[0068] S470：对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0069] 步骤S440-S470请参见多版本运行方法第一实施方式的步骤S110-S140，在此不再赘述。

[0070] 本实施方式中，通过多进程及虚拟环境相结合解决Python服务中不同版本的模型运行时的冲突问题，虚拟环境可以使不同Python版本、模型及第三方库之间相互独立，多进程可以启动不同虚拟环境的执行器，实现在Python服务中运行不同版本的模型，使得开发人员和实施人员可以无感知的在Python服务中运行不同版本的Python、模型及第三方库。

[0071] 请参阅图5，图5是本申请提供的多版本运行方法第五实施方式的流程示意图，该方法包括：

[0072] S510：Python服务接收模型的多个模型版本信息。

[0073] S520：获取各模型版本信息对应的模型相关信息。

[0074] S530：通过多进程组件启动多个模型版本信息对应的多个运行子进程。

[0075] S540：对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型。

[0076] S550:将模型运行结果和/或运行子进程运行时产生的异常信息,发送给Python服务。

[0077] 步骤S510-S540请参见多版本运行方法第一实施方式的步骤S110-S140,在此不再赘述。

[0078] 各运行子进程运行过程中,若产生异常信息,则会将异常信息发送给Python服务,由Python服务对异常信息进行分析,进一步地,在模型运行结束后,将运行的结果也发送给Python服务。

[0079] 请参阅图6,图6是本申请提供的多版本运行方法的简化示意图。

[0080] 在实际应用过程中,Python服务接收到请求后,可以先判断是否需要安装模型,若是,则执行如6所示的安装模型步骤。具体地,安装模型的步骤可以依次包括:读取模型配置、加载指定版本Python、启动安装子进程、创建沙盒目录、安装第三方库、安装模型。其中,读取模型配置可以包括读取模型相关信息(如模型名称、模型版本、模型安装文件、模型对应的第一Python版本信息等)、以及模型依赖的第三方库的信息;加载指定版本的Python可以包括:根据模型对应的第一Python版本信息,在系统中加载第一Python版本的Python,第一Python版本的Python用于为模型提供运行环境;启动安装子进程可以包括:判断模型对应的第一Python版本与Python服务对应的第二Python版本是否一致,若一致,将Python服务的第三执行器作为第二执行器,若不一致,将与第一Python版本对应的第一执行器作为第二执行器,采用第二执行器启动安装子进程;安装第三方库以及安装模型可以为将模型依赖的第三方库的信息以及模型的安装文件存储在沙盒目录中。

[0081] 若不需要安装模型,则进一步判断是否运行模型,若是,则系统监测是否已经安装需要运行的模型,若系统已经安装需要运行的模型,则执行模型运行的步骤,若系统未安装需要运行的模型,则会自动执行安装模型的步骤,安装结束后既可以执行模型运行的步骤。具体地,模型运行的步骤可以依次包括:加载模型配置、加载指定版本Python、启动运行子进程、加载第三方库、加载模型、执行模型计算。其中,加载模型配置可以包括:获取模型相关信息(如模型名称、模型版本、模型安装文件、模型对应的第一Python版本信息等)、以及模型依赖的第三方库的信息;加载指定版本的Python可以包括:根据模型对应的第一Python版本信息,在系统中加载第一Python版本的Python,第一Python版本的Python用于为模型提供运行环境;启动安装子进程包括:根据模型对应的第一Python版本信息获取对应版本的第一执行器,采用第一执行器启动运行子进程;加载第三方库以及加载模型可以包括:在运行子进程中加载模型以及模型依赖的第三方库;最后在子进程中执行模型计算,将计算结果返回给Python服务。

[0082] 请参阅图7,图7是本申请提供的电子设备的框架结构示意图。

[0083] 电子设备70包括相互耦接的存储器71和处理器72,存储器71存储有程序指令,处理器72用于执行存储器71中存储的程序指令,以实现上述任一方法实施方式的步骤。在一个具体的实施场景中,电子设备70可以包括但不限于:微型计算机、服务器,此外,电子设备70还可以包括笔记本电脑、平板电脑等移动设备,在此不做限定。

[0084] 具体而言,处理器72用于控制其自身以及存储器71以实现上述任一组织体系构建方法实施方式的步骤。处理器72还可以称为CPU(Central Processing Unit,中央处理单元)。处理器72可能是一种集成电路芯片,具有信号的处理能力。处理器72还可以是通用处

理器、数字信号处理器(Digital Signal Processor,DSP)、专用集成电路(Application Specific Integrated Circuit,ASIC)、现场可编程门阵列(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。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。另外,处理器72可以由集成电路芯片共同实现。

[0085] 请参阅图8,图8是本申请计算机可读存储介质一实施方式的框架示意图。

[0086] 计算机可读存储介质80存储有程序指令81,程序指令81被处理器执行时,用以实现上述任一方法实施例中的步骤。

[0087] 计算机可读存储介质80具体可以为U盘、移动硬盘、只读存储器(ROM,Read-Only Memory)、随机存取存储器(RAM,Random Access Memory)、磁碟或者光盘等可以存储计算机程序的介质,或者也可以为存储有该计算机程序的服务器,该服务器可将存储的计算机程序发送给其他设备运行,或者也可以自运行该存储的计算机程序。

[0088] 以上所述仅为本申请的实施方式,并非因此限制本申请的专利范围,凡是利用本申请说明书及附图内容所作的等效结构或等效流程变换,或直接或间接运用在其他相关的技术领域,均同理包括在本申请的专利保护范围内。

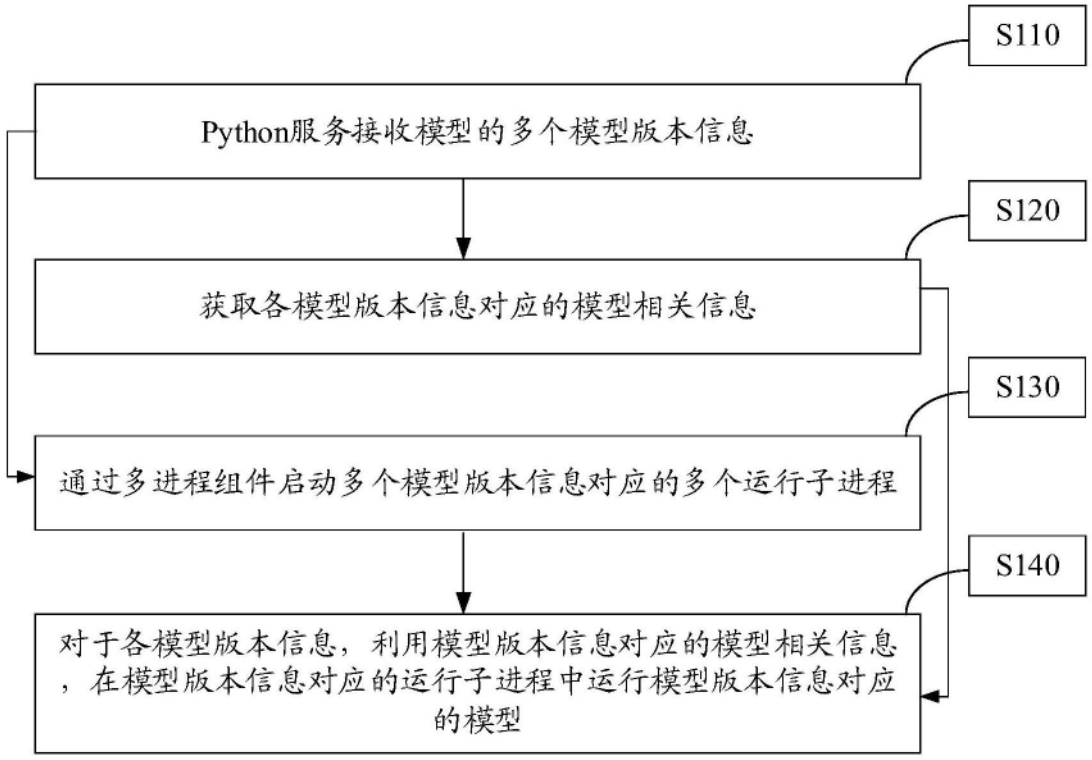

S110
Python服务接收模型的多个模型版本信息
S120
获取各模型版本信息对应的模型相关信息
S130
通过多进程组件启动多个模型版本信息对应的多个运行子进程
S140
对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型

图1

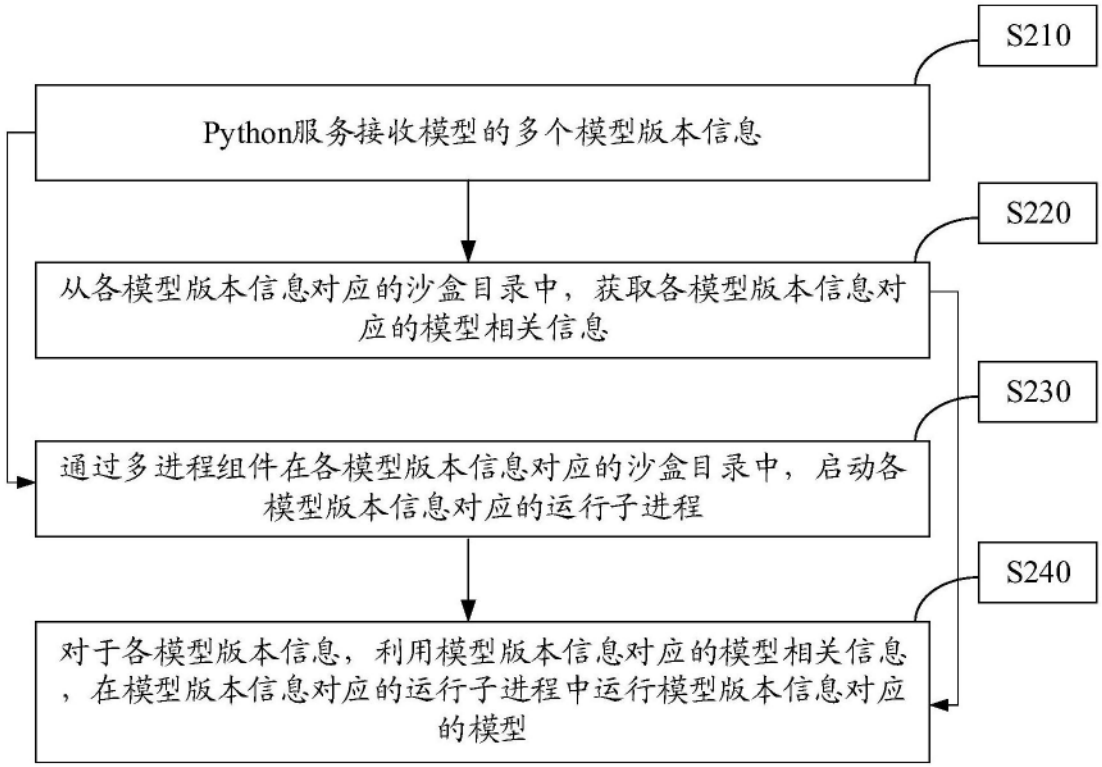

S210
Python服务接收模型的多个模型版本信息
S220
从各模型版本信息对应的沙盒目录中，获取各模型版本信息对应的模型相关信息
S230
通过多进程组件在各模型版本信息对应的沙盒目录中，启动各模型版本信息对应的运行子进程
S240
对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型

图2

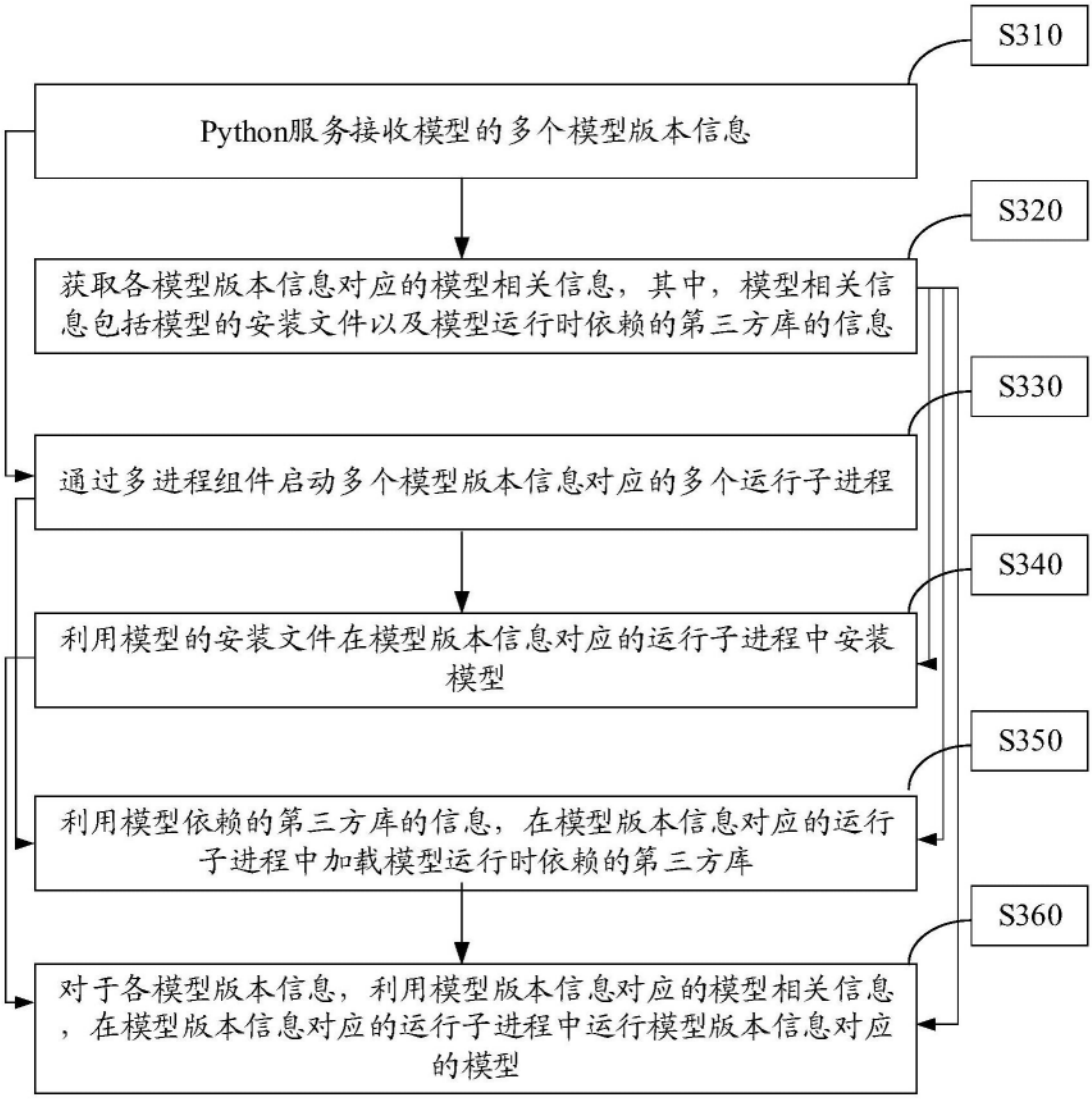

S310
Python服务接收模型的多个模型版本信息
S320
获取各模型版本信息对应的模型相关信息，其中，模型相关信息包括模型的安装文件以及模型运行时依赖的第三方库的信息
S330
通过多进程组件启动多个模型版本信息对应的多个运行子进程
S340
利用模型的安装文件在模型版本信息对应的运行子进程中安装模型
S350
利用模型依赖的第三方库的信息，在模型版本信息对应的运行子进程中加载模型运行时依赖的第三方库
S360
对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型

图3

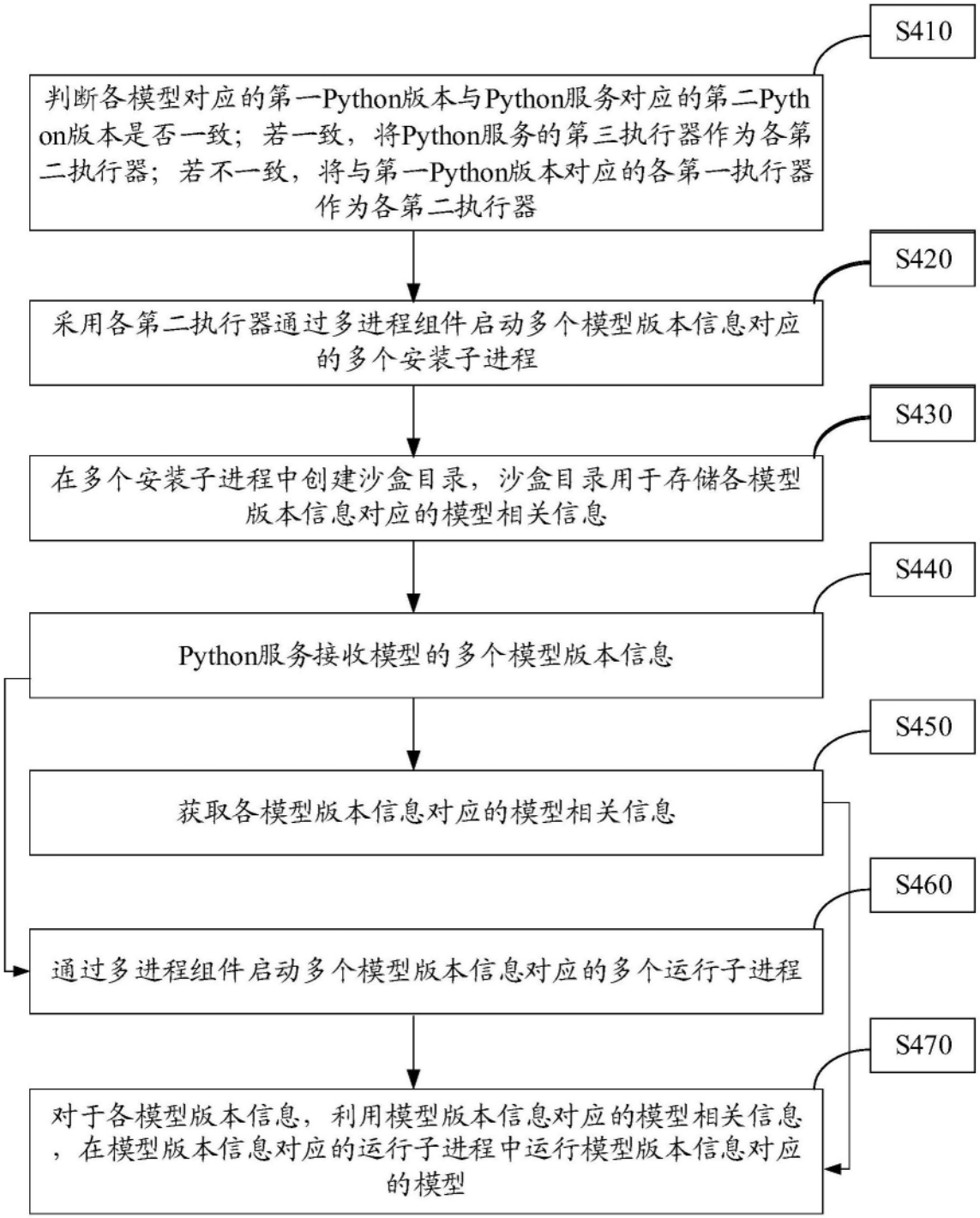

S410
判断各模型对应的第一Python版本与Python服务对应的第二Python版本是否一致；若一致，将Python服务的第三执行器作为各第二执行器；若不一致，将与第一Python版本对应的各第一执行器作为各第二执行器
S420
采用各第二执行器通过多进程组件启动多个模型版本信息对应的多个安装子进程
S430
在多个安装子进程中创建沙盒目录，沙盒目录用于存储各模型版本信息对应的模型相关信息
S440
Python服务接收模型的多个模型版本信息
S450
获取各模型版本信息对应的模型相关信息
S460
通过多进程组件启动多个模型版本信息对应的多个运行子进程
S470
对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型

图4

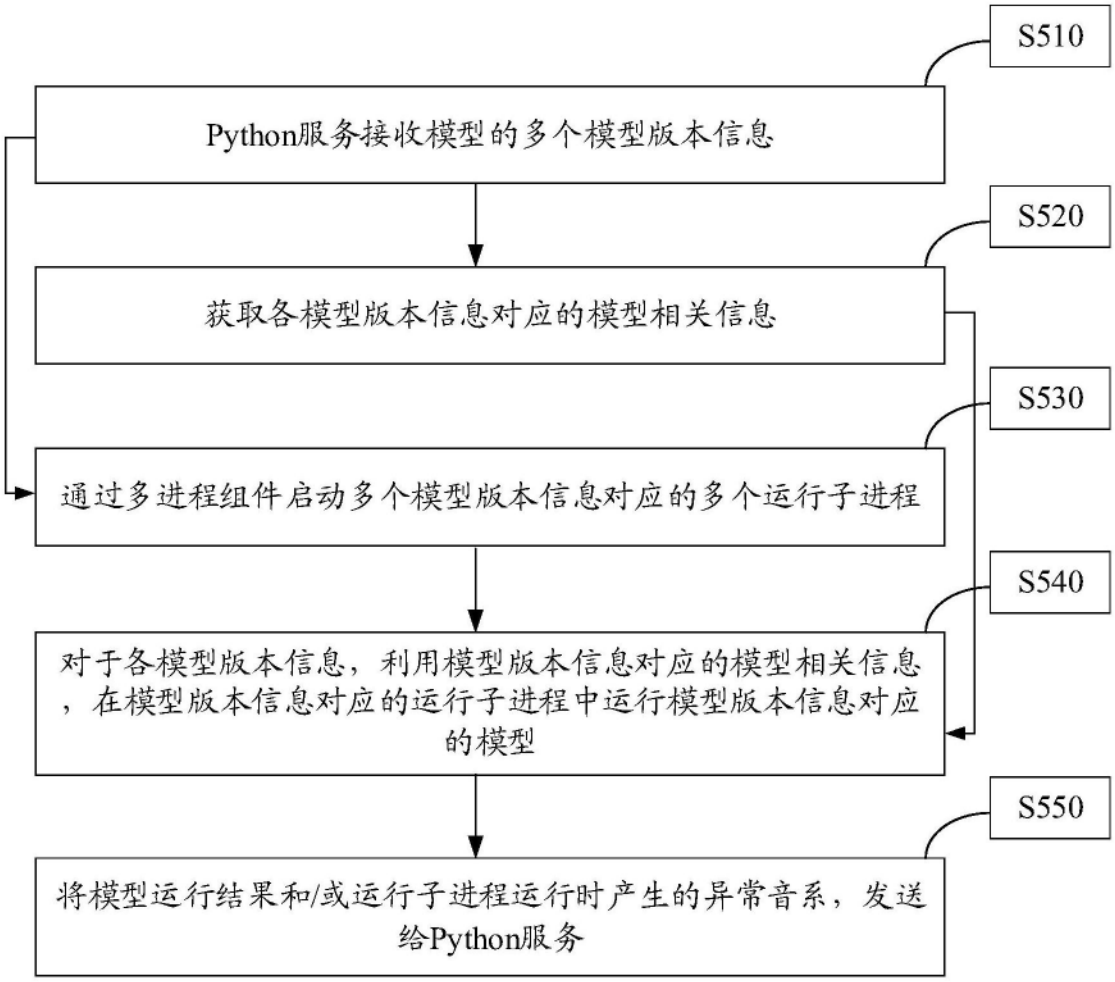

S510
Python服务接收模型的多个模型版本信息
S520
获取各模型版本信息对应的模型相关信息
S530
通过多进程组件启动多个模型版本信息对应的多个运行子进程
S540
对于各模型版本信息，利用模型版本信息对应的模型相关信息，在模型版本信息对应的运行子进程中运行模型版本信息对应的模型
S550
将模型运行结果和/或运行子进程运行时产生的异常音系，发送给Python服务

图5

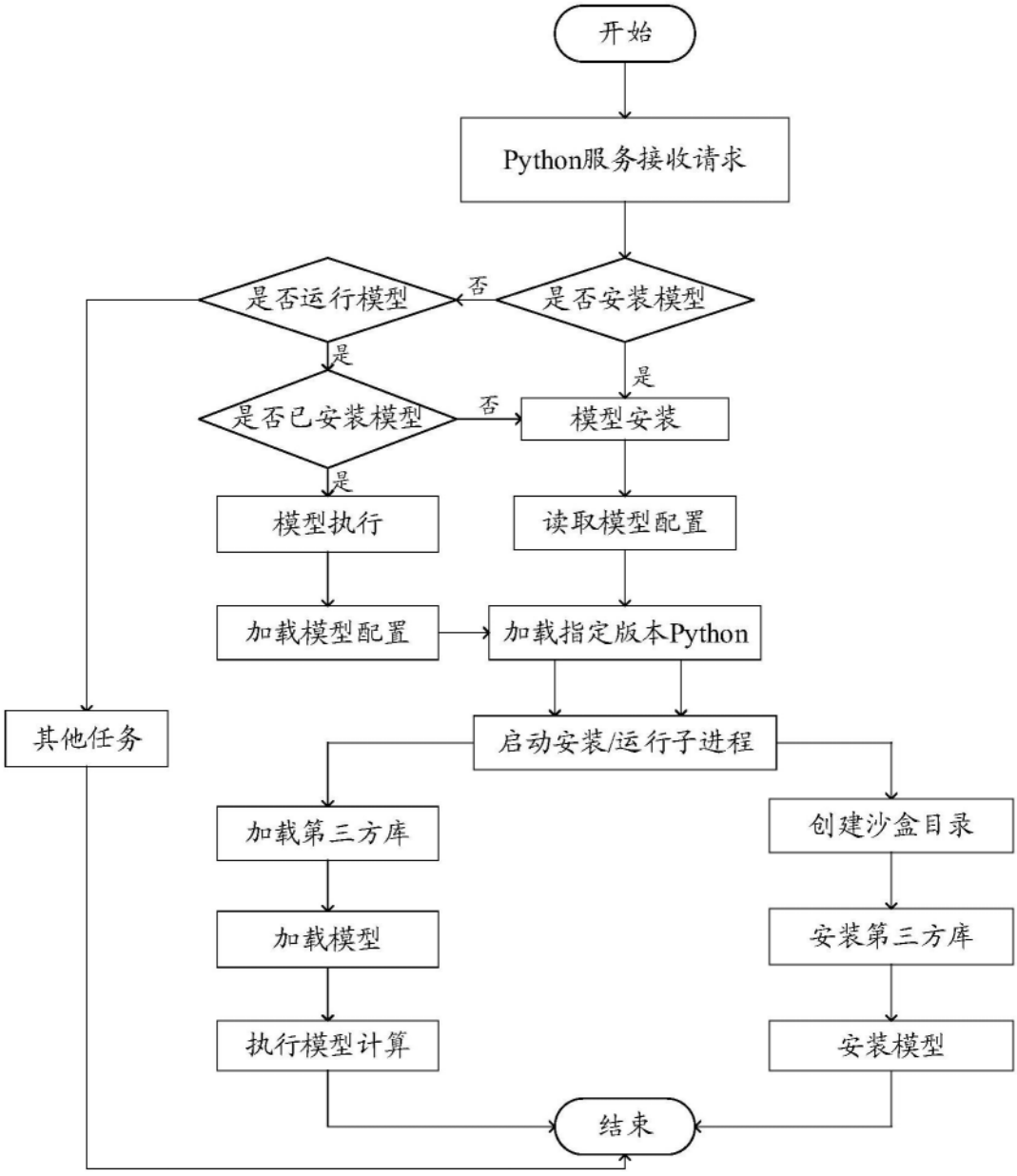

开始
Python服务接收请求
是否运行模型
否
是否安装模型
是
是否已安装模型
否
模型安装
是
模型执行
读取模型配置
加载模型配置
加载指定版本Python
其他任务
启动安装/运行子进程
加载第三方库
创建沙盒目录
加载模型
安装第三方库
执行模型计算
安装模型
结束

图6

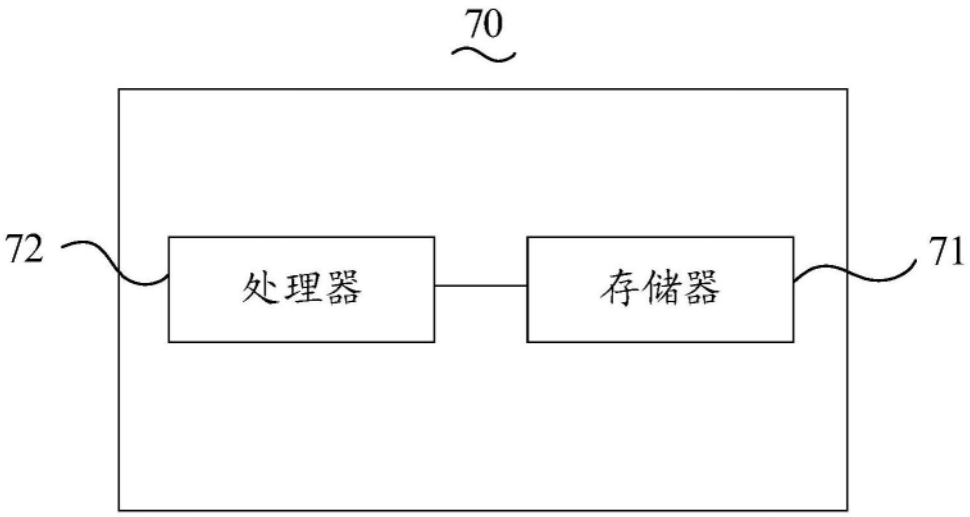


图7

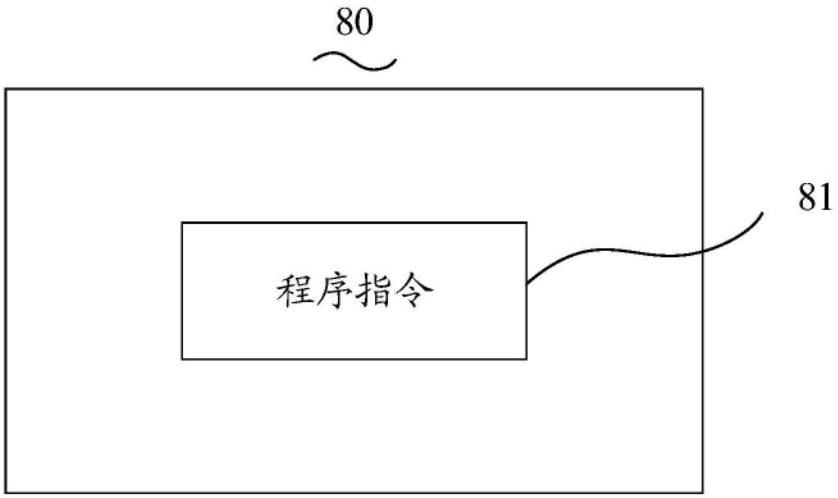


图8