

端智能体验实现方法、装置、介质及设备

技术领域

本申请涉及应用程序开发技术领域，尤其是涉及到一种端智能体验实现方法、装置、介质及设备。

5 背景技术

用户在终端（例如智能手机）通过各类客户端（例如 APP、网页、网站、小程序等）进行工作、交流、购物等，是现代生活的一个重要部分。用户在使用客户端时的即时体验，可能涉及界面设计、页面加载、功能响应、易用性、流畅度、用户反馈等一些列功能，如何基于改善用户体验进行应用程序开发和改进是本领域技术人员需要考虑的技术问题。

10 发明内容

有鉴于此，本申请提供了一种端智能体验实现方法、装置、介质及电子设备，主要目的在于通过实现端上智能以改善用户体验。

依据本申请的一个方面，提供了一种端智能体验实现方法，用于针对适用于多类型客户端的前端代码，在前端进行用户体验的感知、决策和处置，所述方法应用于多类型客户端中任意类型客户端中的一个客户端，包括：

15 启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；

运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

20 依据本申请另一个方面，提供一种端智能体验实现装置，用于针对适用于多类型客户端的前端代码，在前端进行用户体验的感知、决策和处置，所述装置应用于多类型客户端中任意类型客户端中的一个客户端，包括：

配置请求单元，用于启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；

端智能执行单元，用于运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

25 依据本申请的一个方面，提供了一种存储介质，所述存储介质中存储有计算机程序，其中，所述计算机程序被设置为运行时执行上述的端智能体验实现方法。

依据本申请的一个方面，提供了一种电子设备，包括存储器和处理器，所述存储器中存储有计算机程序，所述处理器被设置为运行所述计算机程序以执行上述的端智能体验实现方法。

附图说明

30 图 1 示出了本申请实施例提供的一种端智能体验实现方法的流程图；

图 2 示出了本申请实施例提供的端智能体验处置方案示意图；

图 3 示出了本申请实施例提供的端智能体验监测方法流程图；

图 4 示出了本申请实施例提供的端智能体验监测中复杂事件配置与匹配逻辑示意图；

图 5 示出了本申请实施例提供的端智能体验监测中时机匹配流程图；

35 图 6 示出了本申请实施例提供的端智能体验分层方法流程图；

图 7 示出了本申请实施例提供的端智能体验分层实现逻辑示意图；

图 8 示出了本申请实施例提供的端智能体验增强方法流程图；

图 9 示出了本申请实施例提供的另一种端智能体验实现方法的流程图；

图 10 示出了本申请实施例提供的一种端智能体验实现装置结构示意图。

具体实施方式

为了使本技术领域的人员更好地理解本申请方案，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清楚、完整地描述。

各类型客户端都存在用户体验的问题，因此如何改善用户体验是应用程序开发中需要不断研究的课题。

5 传统的 HTML 5 应用开发效率较高，但用户性能体验不够优秀，于是出现了使用 Web 技术开发具有类似于原生应用的用户体验的 Web 应用的尝试。小程序 (MiniApp) 的出现迅速受到用户的欢迎。MiniApp 是一种小型、免安装、快速加载的程序，通常在一个宿主应用或操作系统中运行 (如小程序、快应用)，也可以是支持跨端部署的 JS 原生应用。小程序本身受架构所限，很难做到类比原生应用的丝滑体验，特别是对于巨型小程序来说，用户操作过程中动态的体验劣化是一个难解的课题。由于小程序的用户体验问题较为突出，因此，本申请实施例

10 以小程序为例进行说明，但是可以理解，对于其他类型客户端都是可以适用的。

现有方案中，通常采用前端采集数据、服务端进行数据计算再反馈给前端的方式进行用户体验改善，该方式可以称为服务端智能或云端智能 (云端 AI) 的方案，虽然具有数据处理能力较强的特点，然而却存在着高延迟、维护成本高的局限性。

因此，本申请发明人创新性地提出，通过端上智能 (或称为“终端智能”或“前端智能”，以下简称“端智能”) 改善用户体验，其中端智能可以理解为在端上具备思考决策能力，简言之，端智能就是在端侧进行实时感知、计算、决策和干预，通过让端具备机器学习能力，带来用户体验提升。本发明实施例中的端智能体验，通过

15 数据分析以及机器学习等技术深入洞察和理解用户行为，全局探测和统筹多样的用户设备环节、复杂的运行场景，通过实施反馈循环和适时的调整干预，以优化客户端产品体验的一种综合性方案，其具有即时性、智能化、可处

20 置的特点。

经过分析以及实践，端智能具有突出的技术优点：1、实时性：端上的实时性是云上难以比拟的，对于许多交互式和流媒体场景而言，云上的延迟几乎无法满足实时的体验需求；2、离线能力：虽然 4G, 5G 已经普及，但是在实际生活中，比如车上，飞机，或者是郊区，经常出现网络连接不稳定，甚至没有网络的情况，这时端智能就可以很好规避网络差的问题；3、数据隐私：处于数据隐私的要求，很多用户数据无法被上传到云端，所以只能在端上使用；4、成本：云计算虽然提供了海量的算力，但是面向海量用户，AI 计算和网络带宽的成本是非常高昂的。

25 而现在端侧设备的普及，存在大量手机、智能车、机顶盒等具备算力的设备可以使用，来降低云上部署的成本。

本申请发明人继续研究发现，实现端智能存在几方面的挑战。第一是包大小挑战，在端上进行程技术，就涉及占用 APP 的安装包大小、存储资源，这对端计算引擎瘦身、端模型大小、端模型压缩、动态发布等能力都提出了较高的要求；第二是性能挑战，由于使用移动设备的计算资源，要兼顾用户体验与计算效率的平衡，需要针对

30 移动端设备 CPU/GPU、内存、耗电、指令集、线程调度等进行持续优化；第三是适配挑战，面对设备、系统版本的碎片化，端智能技术需要进行大量适配，确保运行时的稳定性与可用性。因此，本申请实施例提出一种端智能体验实现方案，旨在克服云端 AI 存在的缺陷，并从上述几个挑战入手，优化方案的实施效果。

此外，在现有网络应用中，同一个网络产品往往对应不同类型的客户端 (不同渠道发布)，例如，一款外卖产品，除了外卖 APP，还具有小程序、网站、网页等多种类型客户端形式。针对同一款产品前端代码，如果针对各种不同类型客户端都分别采取不同的端智能体验机制，则对代码开发以及发布效率带来很大影响。因此，本

35 申请实施例也在此方面进一步改进，以期实现跨端适用的一套端智能体验实现方案。为了达到适用于多类型客户端的方案，本申请实施例依托适用跨端的编程语言 (例如 JavaScript, JS) 进行前端代码开发，由于 JS 开发兼容 APP、小程序、H5 各类型客户端，因此可实现跨端的目的。在具体实施过程中，还需要考虑对源代码进行处理以去除各

类客户端之间的差异性问题。

参见图 1，为本申请实施例中一种端智能体验实现方法的流程图，该方法用于针对适用于多类型客户端的前端

40 代码，在前端进行用户体验的感知、决策和处置，该方法应用于多类型客户端中任意类型客户端中的一个客户端，

包括如下步骤 S101-S102。

本申请实施例中，适用于多类型客户端（也成为“跨端”）的前端代码（例如 JavaScript（JS）前端代码），是通过对源代码进行静态编译去除各类客户端之间的差异性获得的。静态编译，是指编译器在编译可执行文件时，将可执行文件需要调用的对应静态库中的部分提取出来，链接到可执行文件中，使可执行文件在运行的时候不依赖于动态链接库。在本申请实施例中，对源代码进行静态编译主要是处理 JS 源码中约束强且不能动态修改的部分，可包括三个方面：1.模块引用，例如将 JS 源码中的模块引用替换和后缀名修改；2.模版属性映射或语法兼容处理，例如，在 JS 中，模版属性映射通常是指将对象的属性映射到模版字符串中，以生成一个完整的字符串，这种映射可以用于创建动态的 HTML 字符串或者其他类型的文本；在 JS 中，如语法兼容处理目的是确保源码在不同的环境中都能运行，需要关注不同环境可能存在的语法差异，场景的语法兼容处理可采用多种方式，例如，采用严格模式、适用 JSON 对象来序列化和反序列化、使用可选链操作符、使用特定语法、使用模版导入或导出等；3.配置映射，如通过页面配置。由此，通过上述几个方面操作，抹平各类型客户端之间的差异性，为实现跨端实现端智能体验提供基础。

S101：启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；

S102：运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

本申请实施例中，为了减轻端计算处理压力，预先在服务端进行端智能体验配置，当启动客户端之后，加载端智能体验配置并存储在本地，为后续在本地进行决策等操作提供准备。

在服务端（或云端）预先配置的端智能体验配置，可以理解为包括两个维度配置。1、事件规则配置：服务端需要预先定义各种用户行为事件和相关的规则，包括事件的触发条件、规则的执行逻辑等。2、时机策略配置：在场景上时机是指在满足一系列条件的情况下进行某种特定操作的表达，在实现上，时机是复杂事件（复合事件）规则加上触发配置；策略就是触发时机需要满足的条件或者说复合事件规则模型部署：对于需要机器学习模型支持的策略，服务端负责模型的部署和下发。这些模型基于大量历史数据训练得到，用于预测用户行为或评估用户体验质量。

在客户端代码开发中，植入有端智能套件。端智能套件，可以理解为一套解决端智能体验的功能模块集合，其深入洞察和理解用户行为，结合数据分析、机器学习等技术，全面探测和统筹用户的设备环境和场景运行场景，通过实时反馈循环和适时调整干预，优化客户端体验。端智能套件中内置感知引擎、决策引擎及触点引擎，根据加载至本地的端智能体验配置，在相应处理节点自动触发对应的引擎，以实现数据的实时感知、实时决策及实时处置。

在一种实现方式中，可以基于端智能体验架构开发端智能套件。该端智能体验框架可包括如下及部分。1.端智能体验解决层，针对具体问题提供的一套解决策略和工具，它们依赖于基础能力提供的监测和决策能力，针对性地解决用户在使用过程中遇到的各种问题。按照体验维度分类，端智能体验解决层可包括体验监测、体验分层和体验增强三个功能模块。

2.端智能体验插件系统，例如可提供插件市场和插件开发的平台，允许第三方或者内部开发者针对特定功能开发插件。这个系统可以看作是一个扩展平台，依托于基础架构和解决方案提供的服务和 API 进行插件的开发和分发。

3.端智能体验内核，整个系统运作的基础，可由感知、建模、决策和处置三个部分组成，负责收集和处理信息，并作出智能响应。这个内核直接支撑着体验监测、体验增强和体验分层的智能化实施。

4.跨端智能层，是系统的高级功能层，它整合了统一事件流、模型推理、特征工程、决策引擎和触点引擎等能力。

上述各部分和层次之间的逻辑关系体现了从基础到高级的服务层次，每一层都为上层提供支持和服务，共同作用，以保证系统的整体智能化和高效运作。

可见，上述通过端智能体验架构开发的端智能套件，可以理解为以一个统一的体验基座进行构建的，与散点化方案（例如针对特定体验问题进行单独特定方案开发）不同，端智能体验实现方案以统一的体验基座进行建模，这种采用统一体验基建的方式具有降本增效的好处，并且在场景层面上，只需调用基座并渲染呈现即可场景个性化需求，具有可扩展化及个性化的优势。

下面针对 S102 中三个主要动作（数据收集、决策、处置）进行详细说明。

数据收集

端智能套件包括感知引擎，用于收集数据。对用户在上端的实时数据进行感知是端智能体验的基础。S102 中的目标数据是在不同体验解决方案（例如包括体验监测、体验分层、体验增强，后续会详细介绍）中所需要感知的不同类型数据集合。一般而言，目标数据可能包括用户基础数据、用户行为数据、场景数据、性能数据等。

用户基础数据可通过一些 API 获取，例如使用 `getSystemInfo` 获取设备型号，使用 `getNetworkType` 获取当前网络类型，使用 `onNetworkStatusChange` 监听网络状态变化的事件等。

用户行为数据可通过 UT 进行埋点，根据行为所属的事件编号放到不同的事件流中进行埋点和上报。用户行为数据相对复杂，需要一定的计算逻辑。以“页面停留时长事件”为例，涉及如下要点：（1）当用户进入页面时开始计时，用户在页面内浏览超过 xx 秒后触发页面停留事件；（2）当用户在页面内停留时长未达标，即跳出页面，停留时长会暂停计时，返回页面后继续累计计时；（3）一次页面访问（`pageAppear` 到 `pageDisAppear`）内至多只触发一次页面停留事件。

对于场景数据，可涉及页面访问（2001）、点击（2101）、曝光（2201）等行为，其他的行为则被放到了自定义行为中，再根据一定的场景规则整理出不同的场景数据。

对于性能数据，例如接口耗时可通过基础的网络 SDK 收集上报，渲染相关性能数据可依赖于客户端框架和容器（例如小程序框架和容器），结合场景/页面的渲染场景，完成各阶段的数据采集和上报。以小程序为例，具体分为：（1）小程序性能插件：采集应用/页面启动及标准生命周期耗时，自动化上报，其中，（1.1）拓展能力：通过小程序 `performanceApi`，提供部分容器启动的性能数据，主要是启动起点时间及路由耗时；（1.2）针对部分渠道缺乏容器采集 API，无法获取启动时间点，可自定义以应用 `onLaunch` 作为起点的场景 T2 耗时；（2）小程序性能组件 `mini-performance` 采集场景 T2 耗时终点，其中，（2.1）结算方式：组件基于 `IntersectionObserver` 的 API，由场景接入到“首屏完整渲染的最后一个模块”，该组件会记录自己在视图内被看见（&渲染）的事件（&时间）作为模拟 T2 的结算点位，且无额外性能损耗。另外，小程序框架提供了 App 对象的 `onError` 方法，可以用来监听全局的脚本错误，以及 `onPromiseError` 监控所有未捕获的 Promise 异常。至于页面白屏的监测，可采用了一种自定义方式，通过在接口请求完成并返回数据后启动一个计时器，如果在随后的数秒内页面没有任何元素被成功渲染，将此情况视为页面白屏并进行相应的统计。

二.决策

端智能套件包括决策引擎，用于进行体验决策，即依据下载的端智能体验配置，针对实时收集到的目标数据进行体验决策。其中，端智能体验配置可包括事件配置信息或模型配置信息，从而根据事件配置信息进行事件匹配，或根据模型进行预测。其中，事件配置信息例如可以是复杂事件配置信息（对应“体验监测”方案，后文详述），模型可以是体验分层模型（对应“体验分层”方案，后文详述），也可以是全链路动线预测模型（对应“体验增强”方案，后文详述）。本发明实施例中，事件配置以及模型训练和部署是离线的，推理是在前端推理，前端能够实现更快的响应速度，减少网络请求延迟，保护用户数据隐私，以及在没有网络连接的环境下仍能提供服务。

三.处置

端智能套件包括触点引擎，用于根据体验决策结果触发体验处置。本申请实施例提供的端智能体验方案中，

通过收集收集及体验决策从而感知到问题以后，可以直接在端上解决问题或者上报问题，可提供全面灵活的处置能力，例如某些情况下，需要通过弹窗、底部浮条等 UI 形式与用户交互，能够在应用界面推送实时消息，告知用户重要信息或者操作反馈，例如实时消息提示、用户引导等。端智能体验框架可提供完善的 UI 组件，组件高度可定制，组件以动态插件/分包的形式内置于宿主中，可以快速地信息触达到用户。端智能体验框架可以在发送或接收网络请求数据时进行拦截，可以自定义请求头或者对返回数据进行处理，从而对网络进行灵活干预。端智能体验框架可以在不同的场景下对基础库 api 进行动态代理和重写，以此来扩展或修改原有的行为。例如，可以对 hashToUrl 改变进行代理，使其在不同运行状态的机型下返回不同质量的图片。

由此，通过上述三个步骤，即可在端上进行用户体验的实时感知和处置。下面从一个“体验分层”具体实例对上述步骤进行示例性展示。

步骤 1: 准备步骤与模型训练。数据收集: 收集用户访问页面的行为数据, 包括页面加载时间、用户设备类型、网络环境等信息。特征提取: 从收集的数据中提取关键特征, 如设备性能指标 (CPU、内存使用率)、网络速度等。模型训练: 使用这些特征和页面加载时间作为训练数据, 构建机器学习模型, 预测不同机型的运行状态。模型发布: 将训练好的模型转为 ONNX 格式, 发布模型文件到 CDN, 供决策引擎加载预测使用。

步骤 2: 实时特征收集。数据收集: 在用户使用应用时, 实时监控并收集页面加载时间、设备 CPU 使用率和内存使用率等数据。数据归一化处理: 将数据的数值范围调整到一个特定的区间, 通常是 0 到 1 或者 -1 到 1 之间。

步骤 3: 利用模型进行决策。将收集好的特征数据, 根据模型预测的结果进行机型运行状态判定, 如果预测结果超过了设定的阈值, 则判定为低运行状态 (运行状态差或运行状态一般)。

步骤 4: 场景处置。接口加载优化: 对于判定为运行状态差的机型, 自动优化网络加载策略, 如延迟非关键接口的加载等; 内容适配: 根据设备运行状态, 动态调整页面内容, 例如降低图片质量或简化页面元素; 数据反馈: 将处置结果和用户体验反馈继续收集, 作为新的数据输入, 不断优化模型和策略。

如前所述的, 根据用户体验角度, 可以包括体验监测、体验分层和体验增强三个维度。参见图 2, 为本申请实施例提供的端智能体验解决方案示意图。体验监测, 是通过用户对用户体验数据实时的采集、分析和预警, 主动去发现用户在使用客户端过程中的体验问题 (例如是否发生了特定的复杂事件), 并进行实时反馈和现场处置, 从而提升用户体验。体验分层, 是从用户实际使用角度进行用户设备运行状态评估, 与以往僵化地根据出厂硬件信息对设备分级不同, 智能体验分层以应用实际运行的多维度耗时数据为依据, 利用体验分层模型 (通过端智能体验配置获取到), 对用户设备数据进行综合实时加权打分, 并根据打分进行相应的处置。体验增强: 基于用户实时交互行为信息, 利用全链路动线预测模型 (通过端智能体验配置获取到), 实现实时用户未来轨迹预估, 对用户可能访问的分包、页面进行预加载等干预, 提升用户体验惊喜感。

因此, 在一种实现方式中, 上述处置方案包括: 体验监测处置方案、体验分层处置方案, 和/或, 体验增强处置方案。

所述体验监测处置方案包括: 根据体验决策确定发生复杂事件时, 以 UI 交互方式在界面中进行实时消息推送和/或进行页面跳转操作。

所述体验分层处置方案包括: 根据体验决策确定设备运行状态为低运行状态时, 进行接口加载优化和/或页面内容动态调整方案。其中, 体验决策的预测结果可分为运行状态良好、运行状态一般、运行状态差三个级别, 其中, 运行状态良好属于高运行状态, 运行状态一般和运行状态差属于低运行状态, 其中, 运行状态一般可理解为运行状态中等水平, 具有体验改善的空间, 而运行状态差则是亟需进行体验改善。本领域技术人员可以根据具体的性能参数设定和区分各个级别的运行状态。在确定低运行状态时, 可以执行对应的处置方案, 例如, 通过接口加载优化方案进行改善, 包括优化网络加载策略等方式; 再如, 页面内容动态调整方案可包括: 降低图片质量、简化页面元素, 和/或, 简化页面交互。

所述体验增强处置方案包括: 根据体验决策得到用户未来行为轨迹, 确定用户可能访问的分包资源或页面资

源，对所述分包资源或页面资源进行预加载操作。

在一种实现方式中，可以通过设置不同处置方案的优先级，来确定它们之间的执行逻辑关系。因此，上述方法还包括：设置不同处置方案对应不同的优先级，根据优先级确定不同处置方案之间的执行逻辑关系。例如，设置体验分层处置方案对应第一优先级；根据体验分层处置方案的处理结果，确定是否执行体验监测处理方案和/或体验增强处置方案。这种处理方式的好处在于，首先通过体验分层处置方案进行处置，例如针对低性能设备进行降低图片质量等处置，在经过上述体验分层处置之后，如果设备运行状态仍为“较差”状态（可再次通过分层体验模型进行预测得到），那么此时认为当前不适合进行体验监测以及体验增强，以节省资源开销，减轻数据处理压力。

此外，在一种实现方式中，在执行处置方案之后还可以包括：确定执行的处置方案以及收集用户反馈数据，将所述处置方案以及用户反馈数据发送至服务端，作为新输入使服务端优化所述端智能体验配置。由此，将处置结果和用户体验反馈继续收集，作为新的数据输入，不断优化模型和策略。

此外，为了优化实现效果，通过白盒+黑盒的模式进行决策判定。

白盒模型：在线运行时权重已经具体，无需加载推理模型进行推理。白盒模型主要是在用户打开客户端（例如小程序）的冷启阶段，用于初筛低性能设备，以避免模型运行给设备带来额外的负担。这一策略确保了在后续流程中可以通过更高级的模型进行进一步的运行状态诊断和体验优化。黑盒模型：这是一种通过多层非线性变换、抽象全局特征学习等方法构建的深层网络架构，模拟人脑神经系统的分层处理信息的方式，用于解决分类、回归等问题。例如本申请实施例中，全链路动线预测模型可主要依赖于黑盒模型，如深度学习网络（例如 LSTM），来预测用户的行为路径。这种模型通过分析用户的历史行为数据，尝试预测用户接下来可能的操作或偏好。由于深度学习模型通常涉及复杂的网络结构和大量的参数，其内部决策逻辑对于用户来说是不透明的，因此被称作黑盒模型。这类模型在预测用户未来行为轨迹、实现智能预加载等方面发挥重要作用，旨在提升用户体验的流畅性和个性化水平。体验分层模型则更多地使用白盒模型，如 GBDT 算法，来根据设备的运行状况对用户体验进行分层管理。通过分析设备的性能参数（如内存使用、处理器性能等），模型能够实时评估设备的性能水平，并据此调整应用的表现，比如降低图片质量或简化交互效果，以适应不同性能级别的设备。白盒模型的决策过程是透明的，允许开发者明确理解模型的决策逻辑和依据，便于调优和维护。

因此，在一种实现方式中，通过白盒模型或黑盒模型进行体验监控、体验分层或体验增强的体验决策，以及，根据白盒模型进行初步决策，并根据白盒模型初步决策结果确定是否利用黑盒模型进一步进行精细决策。

在一种实现方式中，设置体验分层模型为白盒模型模式；在客户端冷启动阶段，运行白盒模型模式的体验分层模型，根据体验分层模型的预测结果，确定是否继续执行端智能套件或关闭端智能套件；相应的，设置全链路动线预测模型为黑盒模型模式；在客户端热启动阶段，运行黑盒模型模式的全链路动线预测模型。

可见，本申请实施例提供的端智能体验实现方法，强调的是一种跨端、即时、智能的体验优化方法，它不仅关注于提升性能的某一方面，而是从用户体验的全局角度出发，通过实时感知用户状态并动态调整体验，以提供更优质的服务。其核心优势可包括：

1、跨端智能体验：支持跨端使用，例如小程序、不同技术栈如 web 端等，其中，通过静态编译将前端代码转为多端支持代码，能够在不同的平台和设备上提供一致的优化效果；

2、即时反应能力：能够实时感知用户的行为和设备状态，如网络情况、内存使用等，并根据这些信息做出即时的调整。这种即时反应能力能够在用户使用过程中不断优化体验，而不是仅在页面加载时进行一次性优化；

3、智能决策与自动优化：端智能体验框架内置了黑盒模型和白盒模型，能够根据收集到的数据进行智能分析，从而做出针对性的优化决策。这种基于数据驱动和优化方法，能够更精准地识别和解决性能瓶颈。

4、全局策略支持：提供了一系列全局策略，如图片质量降级、包预加载、弱网切换等，这些策略能够在不同场景下灵活使用，从而实现端到端的性能优化。

综上，与传统的性能优化相比，传统方法往往侧重于针对特定问题的技术优化，如代码分割、懒加载、资源压缩等。而本发明实施例基于端智能体验框架，则更加注重于通过实时的数据分析和智能决策，从用户体验的角度出发，提供一种全面、动态和自适应的优化方案。这种方法不仅能够解决传统性能优化中遇到的一些局限性问

题，还能够根据实际的使用场景和用户需求，提供更加个性化和优质的服务。

下面以体验监测、体验分层、体验增强三个角度，对本申请实施例提供的端智能体验实现方法进行示例性说明。

首先，参考图 3-图 5，对体验监测进行说明。

体验监测，是通过用户对用户体验数据实时的采集、分析和预警，主动去发现用户在使用客户端过程中的体验问题（例如是否发生了特定的复杂事件），并进行实时反馈和现场处置，从而提升用户体验。

参见图 3，示出了本申请实施例提供的端智能体验监测方法流程图。

S301：启动客户端，向服务端发送端智能体验配置请求，获取到复杂事件配置信息并进行初始化，复杂事件配置信息包括至少一项目标复杂事件的事件配置信息和时机配置信息；

S302：启动感知引擎，基于前端代码中的统一事件流处理工具，捕捉埋点数据，根据埋点数据确定是否触发了事件配置信息对应的目标复杂事件；

S303：响应于目标复杂事件的触发，运行决策引擎，以依据时机配置信息针对埋点数据进行时机匹配，确定是否满足目标复杂事件的时机；

S304：在时机匹配成功后，运行触点引擎，以触发目标复杂事件对应的体验监测处置方案。

复杂事件可以理解为组合事件，例如，在大数据分析领域实际应用中，可能需要检测“连续登录失败事件”的发生，该事件实际上是“登录失败”和“登录失败”的组合事件，再比如可能需要检测用户下单支付行为，其也是组合事件，“下单事件”之后一段时间内发生“支付事件”，其中还包括了时间上的限制。类似上述多个事件的组合事件，被称为复杂事件。

在本申请实施例中，复杂事件配置信息可以通过一个端智能配置平台预先或实时配置，该端智能配置平台可以是一个可视化平台，可以设置在服务端。通过该端智能配置平台，用户（例如研发人员）可进行复杂事件信息配置，例如，端智能配置平台提供时机决策的配置，包括事件配置和时机配置，事件配置指的是端上用户事件的匹配规则，比如使用“spm == a.b.c && click”来对应某个固定位置的点击事件，时机配置是事件的序列，用来对端上的用户事件流的发生序列进行复杂匹配，比如“点击进店”然后“加购”然后“返回首页”。特别的，为了针对不同类型客户端进行个性化配置，可以通过端智能配置平台，针对不同类型客户端配置对应的复杂事件集合，其中，复杂事件集合包括多项复杂事件，每项复杂事件包括事件配置信息和时机配置信息。

参见图 4，为端智能体验监测中复杂事件配置与匹配逻辑示意图。在该例子中，首先在配置管理平台进行复杂事件配置，包括事件配置和时机配置，其中，事件配置（可理解为特征配置）是指对用户行为的累积和统计，如 3 天内进店铺的次数；时机配置是指可触发决策和响应的特定条件，如两次进店铺并加购。在配置管理平台完成配置后，可以将配置信息写入一个数据库 DB，然后，由一个配置网关服务从该 DB 读取数据。在客户端启动之后，客户端自动从该配置网关服务中加载配置信息。获取到配置信息后，客户端的端智能套件根据该配置信息进行相关决策。具体的，通过决策引擎进行事件映射/过滤、模式匹配、行为累积等，从而完成时机决策。决策引擎确定时机匹配成功后，运行触点引擎以触发对应的处置方案，例如，通 UI 交互方式在界面中进行实时消息推送，或者针对当前操作失败事件自动进行页面跳转操作（例如针对多次支付失败进行前置收银台跳转）。

当客户端启动之后，会自动加载复杂事件配置信息，如步骤 S301 中，通过向服务端发送请求，获取复杂事件配置信息并进行初始化，其中复杂事件配置信息包括至少一项目标复杂事件的事件配置信息和时机配置信息。其中，复杂事件配置请求中可携带有经纬度信息、端信息以及用户 ID，其中端信息可以是指客户端类型信息，服务端接收到上述复杂事件配置请求后，可依据端信息进行客户端类型判断，从而匹配该类型客户端对应的复杂事件

配置信息并下发。

本申请实施例中，统一事件流处理工具，是指基于超级位置埋点的事件流标准化处理工具，可以根据各类型客户端的埋点信息，将用户行为统一归纳为标准的事件，包含埋点信息、事件 ID (eventId)、参数信息 (properties) 等信息字段。其他的页面或插件里的函数通过监听统一事件，即可获得当前用户在端上进行的操作。

5 其中，超级位置埋点包含如下几个方面的设置。

事件标记

例如，通过 a.b.c.d 用来标记某个页面中的某个事件，其中，a 位代表用户端层级、b 位代表端内的某个页面、c 位代表某个页面中的某个区块、d 位代表某个区块中的某个具体的点击位置。

事件类型

10 上述 abcd 位仅能标记一个具体的位置，而区分不同的事件类型，基于特定事件类型记录不同的通用参数信息，才能覆盖到所有的统计需求，同时也有助于降低数据加工的难度和资源消耗，例如可采用如下三种类型：页面类型：为了统计页面颗粒度的用户行为，当进入和退出页面时，需要记录用户 id、设备 id、时间戳、进入还是退出等参数，则可以统计每个页面的 UV、PV、用户停留时长等数据；点击类型：长按、滑动等都算作点击类型；曝光类型：指的是对于某个特定模块级别的曝光，比如说商品的曝光、视频的曝光，结合点击类型的事件一起分析，
15 则可以计算出某个商品、帖子、视频的点击率。

跳转关系

在跳转关系上，只要在 URL 中记录上一跳和上两跳的关系即可。例如，对于“在购物车页面点击了第一个商品”，则可以上报一次点击事件；但上一跳可能是从某个商品详情过来的，则可以将上一跳进行记录，上两跳可能是从首页的“猜你喜欢”信息流的某商品跳转过来的，则可以将上两跳进行记录，基于此，将用户的动线进行了串联。
20

流量通道

例如，针对首页的某一个频道包含了多个页面，此时引入流量通道。比如，把一个秒杀频道包含的多个页面，都标记为一个流量通道，那只要将这三个页面的浏览人数进行去重，则得到了这个频道的总 UV，其他的点击、页面总访问数等也是同理在统计逻辑上处理即可。

25 其中，在上述步骤 S302 在统一事件流处理工具捕捉到埋点数据之后，还可以包括：对埋点数据进行客户端类型分析及格式分析，并对埋点数据进行标准化处理，使得不同类型客户端的数据具有统一的格式和结构，其中，对埋点数据进行标准化处理可具体包括：解析埋点数据的埋点信息、事件 ID、参数信息；对埋点信息、事件 ID 以及参数信息进行统一格式封装，转换为标准格式的事件信息。

可见，本申请实施例中的统一事件流处理工具，可以理解为客户端标准化事件流处理工具，利用跨端（APP、小程序或 H5）源代码中的超级位置埋点，捕捉埋点数据，并根据埋点数据进行封装然后转化为标准化事件。通过引入统一事件流处理工具，监听处理工具触发的目标复杂事件，当目标复杂事件发生时，开始触发复杂事件时机决策引擎的过滤和匹配。这种统一事件流处理方式，是实现跨端处理复杂事件的重要一环，通过基于超级位置埋点捕获埋点数据，可以在客户端捕获到各种类型各种位置的事件信息，并通过对埋点数据进行标准化处理，使得后续利用统一的复杂事件时机决策引擎基于同一套复杂事件匹配规则库和相同的匹配逻辑进行复杂事件检测和处置成为可能。
35

本申请实施例中，决策引擎用于针对复杂事件进行时机决策（可称为“时机决策引擎”），可基于自研的复杂事件匹配规则库并被声明为静态函数的复杂事件处理组件，其中，自研的复杂事件匹配规则库可采用 JavaScript 开发以兼容多类型客户端。自研复杂事件匹配规则库的好处在于，可以根据场景需求进行个性化配置或更改，据此复杂事件匹配规则库，进行具体的个性化规则匹配。其中，为了实现跨端适配的时机决策引擎，可通过前述 JS

静态编译的处理方式，将复杂事件处理组件声明为静态函数，以去除多端之间的差异性，得到跨端时机决策引擎。

在一种实现方式中，复杂事件时机决策引擎的具体处理过程为：根据预先配置的复杂事件匹配规则，对目标复杂事件进行事件处理和时机匹配，其中，复杂事件匹配规则包括事件规则、模式规则和时机规则。例如，针对触发的目标复杂事件，根据事件规则进行事件过滤，并针对过滤后的事件进行事件转换，得到标准格式的事件 ID；

- 5 根据模式规则对转换后的事件进行模式过滤，保留时机规则对应的事件，以及，对保留下的事件进行模式匹配，其中，根据模式规则对应的多条 rule 规则和 item 规则进行规则匹配以确定一轮模式匹配是否成功，根据多轮模式匹配进度确定模式匹配是否成功；模式匹配成功之后，根据时机规则进行疲劳度检测，其中，根据疲劳度缓存信息进行最新疲劳度检测，若疲劳度检测通过则确定时机匹配成功。

其中，复杂事件处理中的模式匹配，可以理解为两方面：每个简单事件的特征以及简单事件之间的组合关系。

- 10 此外，还可以扩展模式的功能，例如，匹配检测的时间限制，每个简单事件是否可以重复出现，对于事件可重复出现的模式，遇到一个匹配后是否跳过后面的匹配等。模式匹配根据各个事件中的近邻关系及条件，检测连续的事件或不连续但先后发生的事件，模式还可能有时限限制，如果在设定时间范围内没用满足匹配条件，就会导致模式匹配超时。

参见图 5，示出了时机决策引擎处理过程的一个示例。

- 15 S501，事件触发：监听到统一事件流处理工具触发的目标复杂事件。

S502，事件过滤：根据事件规则进行过滤，比如当前只配置了点击事件，那么所有的曝光事件、滚动事件都将被过滤，减轻后续匹配压力。

S503，事件转换：将过滤之后的事件转化为可以进行匹配的统一事件，其中把前端的事件映射为后台配置的事件的 ID。

- 20 S504，模式（Pattern）过滤：在一个时机中存在多个复杂规则，每个规则配置的事件不一样，为了降低匹配压力，这里进一步过滤，只留此规则用到的事件。

S505，模式匹配：例如通过深度优先遍历规则，依次对事件进行匹配，包含多条 rule 规则。其中，模式匹配进一步包括 rule 匹配和 item 匹配，其中，rule 规则是指复杂规则中的一条规则，包含多个 item；item 匹配：item 与后台配置的事件一一对应，这里进行底层的事件匹配，完成后按层级返回匹配结果。

- 25 S506，查看匹配进度：每完成一个模式匹配进度加一，全部完成进入下一个环节。

S507，匹配结果：确定匹配成功或者失败。

S508 疲劳度检测：匹配成功之后检测疲劳度，比如规则设置为：3 天内触发三次下单；若 3 天内如果下单小于等于 3 次，疲劳度为真（true），允许后续回调行为；若 3 天内如果下单大于 3 次，疲劳度为假（false），不再允许进行后续回调行为。其中，可将历史疲劳度缓存到本地，下次进入客户端会加载。

- 30 步骤 509：决策完成。

在一种实现方式中，可在时机匹配成功之后，自动调用复杂事件时机决策引擎预先注册的回调函数（例如 callback 函数），通过回调函数启动触点引擎，以执行与目标复杂事件对应的处置方案。其中，如前描述的，端智能配置平台可实现复杂事件配置，与此对应的，可以通过端智能配置平台，接收目标复杂事件的时机匹配成功信息，并根据目标复杂事件的场景属性和/或目标复杂事件对应类型客户端的端属性，确定对应的处置方案。其中，

35 场景属性可以包括应用场景类型信息，客户端端属性用以区分客户端类型，因此，通过上述方式可以实现针对目标复杂事件进行个性化处置方案的处置，其中，处置方案例如可以是提示窗或浮层的方式。

以“用户连续三次支付失败自动切换至前置收银台”为例，对体验监测进行示例性说明。

步骤 1：在端智能配置平台，配置用户支付失败事件（如果已经配置，可复用），此处假设该事件为事件 F；

步骤 2：在端智能配置平台，配置用户支付失败事件发生次数为三次，事件 D 匹配次数为 3；

- 40 步骤 3：客户端启动客户端之后，请求配置信息并进行初始化；

步骤 4: 用户在支付失败一次（统一事件流处理工具通过埋点确定）之后，触发时机决策引擎进行时机匹配，次数记为 1；

步骤 5: 用户在支付失败两次之后，触发时机决策引擎进行时机匹配，次数记为 2；

步骤 6: 用户在支付失败三次之后，触发时机决策引擎进行时机匹配，次数记为 3；

5 步骤 7: 匹配度检测通过后，调用注册的回调函数（例如 callback 函数）；

步骤 8: 运行触点引擎，自动切换至前置收银台，前置收银台能在商户页面选择具体支付平台完成支付，避免跳转到中间页，帮助优化网银支付体验。前置收银台支持在 Web、WAP、App 端集成。

在上述“用户连续三次登录失败”例子中，时机决策引擎是根据匹配规则进行匹配的引擎，只是进行数据解析和规则匹配，可以理解为用户声明为静态函数但会缓存一部分执行过程的核心组件。其中关于匹配规则以及复杂事件之间的关联，可以在服务端进行预先配置，而执行过程中的中间环节（触发复杂事件与时机匹配）则没用关联，可以理解为用户各自为黑盒的实现方式。再以“监测页面同位置反复点击无响应”为例，首先进行监测策略的配置，例如，先根据场景需要配置监测策略，约定匹配规则与触发时机：事件匹配逻辑：通过 ab 位区分页面，通过 c 位区分场景，通过 d 位区分坑位，通过 e 个时机配置即可覆盖页面上所有可点击区域；触发策略：2s 内连续 3 次及以上点击；然后，通过埋点将用户异常实时上报，对能做出处置的异常实时进行干预，例如，针对不同点位做出针对性处理，如页面降级，跳转至 H5 页面详情页等。由此，相比于传统监控很难感知偶发性异常相比，本申请实施例体验监测可基于用户会话第一时间发现问题，并在客诉产生前先行介入处理，改善用户体验。

另外，以“提单手机号智能纠错”为例，在用户下单场景下，通过相似度分数分布进行决策，当确定用户错填手机号的可能性较高时（例如打分在 0.7-1 之间），通过手机号纠偏，提醒用户一键修改手机号（前端可实现），例如，通过气泡提示进行纠偏，并且可以预判用户将填入的手机号而给予一键填入的功能。

20 此外，体验监测还有许多其他场景实例，例如弱 WIF 场景自动切换移动网络等，本申请实施例不一一说明。

在一种实现方式中，上述方法还包括如下步骤：判断目标复杂事件是实时行为事件或累计行为事件，若是实时行为事件，则实时获取用户行为数据，并实时基于用户行为数据对应的埋点数据进行复杂事件触发判断及时机匹配，若是累计行为事件，则分多次获取用户行为数据，并对用户行为数据进行累积，以及，基于当前累积的用户行为数据对应的埋点数据进行复杂事件触发判断及时机匹配。可见通过上述方式，将用户行为数据作为匹配的依据，可以实现更智能化的决策。与针对用户的实时用户行为事件进行处理不同，针对累计行为事件（比如：一周内点单 5 次）的处理，可以实现领域联动：时机决策的过程中可以使用累积的用户数据，并提供完善的接入方案。

可见，通过上述体验监测技术方案，本申请实施例针对适用于多类型客户端的 JavaScript 前端代码，在前端检测复杂事件并处置，其中在客户端启动之后，向服务端发送复杂事件配置请求，获取复杂事件配置信息并进行初始化，然后基于前端代码中的统一事件流处理工具，捕捉埋点数据，根据所述埋点数据确定是否触发了所述事件配置信息对应的目标复杂事件；接着，响应于目标复杂事件的触发，运行决策引擎，以依据所述时机配置信息针对所述埋点数据进行时机匹配，确定是否满足所述目标复杂事件的时机；最后，在时机匹配成功后，触发回调函数，以执行目标复杂事件对应的处置方案。本申请实施例中，相较于主要依赖服务端进行复杂事件检测及处置的方式，在客户端侧进行复杂事件处理，由此可针对实时获取的数据进行实时性的规则匹配，由此缩减处理时长，具有低延迟以及维护成本低的优势。

并且，与现有技术复杂事件处理方案仅能适用一个类型客户端的方式不同，本申请实施例也在此方面进行改进，以期实现跨端适用的一套复杂事件处理方案。为了达到适用于多类型客户端的复杂事件处理方案，本申请实施例依托 JavaScript (JS) 进行前端代码开发，由于 JS 开发兼容 APP、小程序、H5 各类型客户端，因此可实现跨端的目的，通过一套事件处理机制即可对各类型客户端通用，大大减轻了开发压力。

40 此外，为了方便地针对不同类型客户端进行个性化复杂事件配置，可通过一个端智能配置平台进行可视化配

置，从而方便地配置复杂事件的匹配规则，使策略更加灵活和易用。借助端智能配置平台，达到“输入配置”-“输出时机匹配结果”的简单实现方式。

接下来，参考图 6-图 7，对体验分层进行说明。

参见图 6，为本申请实施例提供的端智能体验分层方法流程图。

- 5 S601：启动客户端，向服务端发送端智能体验配置请求，获取到体验分层模型；
S602：启动感知引擎，获取设备硬件静态数据、设备实时状态数据以及场景实时运行数据；
S603：触发决策引擎，将获取的设备硬件静态数据、设备实时状态数据以及场景实时运行数据输入至体验分层模型，预测设备运行状态；

S604：运行触点引擎，根据设备运行状态的预测结果触发对应的体验分层处置方案。

- 10 在实际应用中，试图通过对用户设备的运行状态进行甄别，对低运行状态的设备给予策略上的优化，以期达到正常的运行状态及用户体验。以小程序为例，受限于小程序容器及框架，以及繁杂的场景逻辑，在性能、体验方面较难做到各个机型的体验一致，基于长尾理论，把解决的重点放在了低性能设备上，以少量的资源辅助，解决体验上的拉胯问题。现有方式中针对低性能设备的判断，通过基于以下几个关键因素：硬件规格、价格、性能和功能等，这些判断标准是相对静态的，并不能根据用户设备的实时运行状态做更细致的区分。比如一个高性能手机在打开了几个大型游戏 app 之后，再打开目标小程序，页面非常卡顿时，传统的判别方式仍认为此刻是高性能设备。而本申请实施例会结合页面渲染的表现来进行实时判断。而本申请实施例的体验分层方案中，从用户实际使用角度进行用户设备运行状态评估，与以往僵化地根据出厂硬件信息对设备不同，智能体分层以应用实际运行的多维度耗时数据为依据，利用端智能体验分层模型，对用户设备数据进行综合实时加权打分，并根据打分进行相应的处置。

- 20 针对体验分层方案，需要首先在服务端进行数据收集和处理，并训练完成体验分层模型，然后，将体验分层模型下发至前端，在客户端运行过程中，基于获取的目标数据，在前端实时地通过体验分层模型进行分层预测，并基于用户设备运行状态进行场景降级等处置。下面从数据收集和模型结构确定两个角度对方案进行介绍。

一、数据收集及处理

- 25 数据的采集和处理是模型训练的关键环节。针对数据处理可以做以下流程上的划分：特征维度划分、数据采集与上报、数据读取、数据清洗、数据划分、数据预处理。经过这些步骤给到模型标准的输入。

(1) 数据维度划分

与现有低性能设备判定方案不同，本方案中的数据不仅包括手机型号、像素比等静态数据，还包括手机实时动态运行的数据，如当前可用内存、告警次数、各页面加载耗时、各阶段内存的变化情况等。手机运行好坏更具实时性的特性，该方案能够及时捕捉到运行状态不佳的时刻，辅以降级手段，让运行状态不再劣化，甚至转好。

- 30 以体验分层模型为例，所收集的数据可分为静态数据和动态数据，静态数据包括用户手机的硬件条件，如手机型号、配置、像素比等，动态数据分为设备相关与场景相关数据。

- 本申请实施例中，采集的数据按照类型可划分为：设备硬件静态数据、设备实时状态数据以及场景实时运行数据，其中，设备硬件静态数据例如包括手机型号、内存容量等参数；设备实时状态数据，是指动态变化的设备数据，例如电量或网络状态等参数；场景实时运行数据，是指与场景运行有关的数据，包括但不限于首页场景、店铺场景等，例如包括首页加载耗时等参数。本申请实施例对于各类型数据的具体参数不作限定，例如包括参数 a、参数 b、参数 c 等。

(2) 数据采集与上报

数据采集方式按照特征维度不同而不同。静态数据只读取一次进行缓存，减少耗时，动态数据在采集节点时做提取动作，保证精准。

- 40 以小程序为例，数据上报方式支持不同维度：

普通埋点数据提供天维度报表，适用于 T+1 离线分析场景；
highway 可分钟级别回传数据，适用于在线实时分析的场景；
answer 灵活度比较高，可自定义数据形态，适用于更丰富的场景；
mtop 央带，灵活度高，实时性高，但受限于接口 qps 及触发时机，适用于端云协同场景；

5 (3) 数据预处理

数据的质量，直接决定了模型的预测效果和泛化能力，决定了模型的天花板。因此数据处理涉及到几个维度：准确性、完整性、一致性、实效性、可信性、可解释性。针对体验分层模型的数据虽然经过了基本的数据清洗，但仍包含了大量的噪音。

因此在模型之前需要进行预处理，通过填补缺失值、平滑噪音数据、平滑或者删除离群点、解决不一致数据等10 等手段，从而获取到标准的、干净的、平滑的数据。另外，还可以通过分箱的方式将连续数据离散化，可按一定规则将连续的数据分到对应的桶中。分箱规则有无监督分箱（等频分箱、等距分箱）、有监督分箱（卡方分箱、决策树分箱）等。常用等频和等距。结合体验分层数据特点，可采取等频分箱。

二、模型结构及训练

本申请实施例中，可采用卷积神经网络（CNN）构建体验分层模型。CNN 的训练过程包括前向传播（网络根15 据输入数据和当前权重计算输出），计算损失（基于输出和真实标签），反向传播（根据损失计算每一层的梯度），以及权重更新（使用梯度下降算法或其他优化算法调整网络权重）。

经过进一步研究和探索发现，体验分层模型首先要解决的是分类问题，经典算法的如 GBDT、CNN、DNN 等是候选模型。通过先后尝试了卷积神经网络（CNN）和深度神经网络（DNN）。他们各具优劣势。CNN 主要设计用来处理具有空间连续性的数据（如图像），通过引入卷积操作和池化操作，能够有效地处理图像等空间数据，20 但是在其他类型的数据上可能不如 DNN 高效。DNN 在结构上更加简单和通用，但可能面临巨大的参数空间和过拟合问题，但是可以优化避免这种问题。基于采集的是手机静态数据和内存等动态数据，在 DNN 上的综合表现优于 CNN。结合两个算法的优势，在优化方案中，可以借助 CNN 进行特征提取，然后将提取的特征输入到 DNN 中进行最终的模型训练。

在一种实现方式中，通过“白盒+黑盒模型”的方式训练得到体验分层模型。如前描述的，白盒模型，其在线25 运行时权重已经具体，无需加载推理模型进行推理。白盒模型主要是在用户打开客户端（例如小程序）的冷启阶段，用于初筛低性能设备，以避免模型运行给设备带来额外的负担。这一策略确保了在后续流程中可以通过更高级的模型进行进一步的运行状态诊断和体验优化。黑盒模型：这是一种通过多层非线性变换、抽象全局特征学习等方法构建的深层网络架构，模拟人脑神经系统的分层处理信息的方式，用于解决分类、回归等问题。具体的，可以通过“白盒模型训练 GBDT 结构+黑盒模型训练 DNN 结构”的方式，得到体验分层模型。这种方式的好处在于，首先通过使用白盒模型（如 GBDT 算法），来根据设备的运行状况对用户体验进行分层管理。通过分析设备的性能参数（如内存使用、处理器性能等），模型能够实时评估设备的性能水平，并据此调整应用的表现，比如降低图片质量或简化交互效果，以适应不同性能级别的设备。白盒模型的决策过程是透明的，允许开发者明确理解模型的决策逻辑和依据，便于调优和维护。然后，通过使用黑盒模型（如 DNN）通常涉及复杂的网络结构和大量的参数，其内部决策逻辑对于用户来说是不透明的，这类模型预测能力强，旨在提升用户体验的流畅性和个性化水平。因此，通过“白盒+黑盒”分别训练不同网络结构得到体验分层模型的方式，可以发挥各自模型的特点，35 通过白盒模型初筛低性能设备，通过黑盒模型进行准确预测，以实时分级处理方案。

因此在一种实现方式中，通过训练第一网络结构（例如 GBDT 或 CNN）得到第一体验分模型，以及训练第二网络结构（如 DNN）得到第二体验分层模型，在决策过程中，通过第一体验分层模型进行白盒预筛以及通过第二体验分层模型进行黑盒精筛，从而预测出设备运行状态。例如，通过白盒预筛，初步判定 15%设备为运行状态不佳，可降级处置，然后，通过黑盒精筛，精细化判定剩余 85%设备，仍有 30%可降级处置。这种“白盒+黑盒”相40

结合的体验分层方式，可全面地进行设备分层处置，实验证明，可使 50%设备显著节流，平均内存告警次数减少 10%，大大提高客户端运行流畅度、改善用户体验。

在端智能体验分层方案中，还有一个技术关键在于，将在服务端训练好的体验分层模型进行发布和部署，使得客户端在运行过程中可以获取到体验分层模型并利用模型进行实时预测，相较于服务端或云端从前端获取数据然后再进行模型预测的方式，在前端进行预测可减少数据上报的步骤，从而加快预测的速度，减少延迟，并且，这种在前端进行模型预测方式，不限于网络状态，即使在网络卡顿或故障时，依托前端数据处理能力，也可以完成模型预测。因此，在客户端运行体验分层模型，是一个技术点。

参见图 7，示出了本申请实施例提供的端智能体验分层实现逻辑示意图。整个过程包括模型准备、环境准备、参数准备、推理打分四个主要环节。

模型准备环节，是离线训练得到体验分层模型，如前述的在服务端进行数据采集与处理，以及模型结构确定，从而利用处理后的数据训练模型结构，得到体验分层模型。

环境准备环节，是指在客户端进行运行模型的环境准备，主要包括启动引擎。以小程序为例，首先，根据小程序启动之后，根据端智能配置信息进行模型下载（例如，通过端智能配置平台的配置能力，将模型及相关控制参数配置在平台上，然后，将模型保存在一个 DB 中，通过一个配置网关服务从该 DB 读取模型；在客户端启动之后，客户端自动从该配置网关服务中加载模型，可参考图 4 及说明），然后，进行 Tensor（张量）准备；接着，通过接入图形程序库（例如 WebGL）、编程方案（例如 WASM，使用非 Java 编程语言编写代码并且能在浏览器上运行的技术方案）、图像处理（webGPU）等功能，以实现动态切换 JS 引擎和 NA 引擎。

在参数准备环节，主要进行模型推理参数准备，可包括计算图解析、算子加载、权重加载、加载就绪等步骤。

在推理运行环节，主要将推理参数输入模型得出预测结果，可包括 Tensor 输入、输入预处理、算子运算、得出结果等步骤。

由此，通过上述四个环节，即可采用体验分层模型进行实时推理和打分，由此判定用户设备（例如手机）实时运行状态的好坏。

需要说明的是，以小程序为例，由于小程序场景可运行的模型文件格式受限为 .xnn，因此需将其他格式的进行模型结构优化、异构模型编译，其本质是算子运算的转换。受限于小程序的运行环境与框架，模型的复杂度应简单为好，复杂的模型结构一是模型文件庞大加载耗时长，二是运算复杂耗费机器资源。故在模型设计时本着简单计算的原则进行设计，避免使用复杂算子及复杂流程。

因此，在获取到体验分层模型之后，上述方法还可以包括：环境准备环节：基于下载的体验分层模型进行张量准备，基于模型数据预处理阶段的逻辑对应到 JS 中，载入 XNN 解析器，以实现动态 JS 引擎和 NA 引擎；参数准备环节，基于客户端支持的模型格式进行模型参数准备，包括模型结构优化以及异构模型编译。

模型结构优化 指的是通过改进模型的架构来提高效率和减少资源消耗的过程。这通常涉及以下几个方面：剪枝(Pruning): 移除不重要的权重或神经元来减少模型大小。量化(Quantization): 降低模型参数的精度来减少模型大小和加速推理，例如从 32 位浮点数转为 8 位整数。结构化简(Simplification): 对模型进行简化，去除过于复杂的部分以提升运行效率。知识蒸馏(Knowledge Distillation): 将大型模型的知识转移到更小型的模型中。自动化模型压缩(AutoML for model compression): 使用自动化算法找到较优的模型压缩方法。

异构模型编译: 与模型结构优化密切相关，指的是为不同的硬件平台（如 GPU、CPU、TPU 或者 FPGAs 等）编译深度学习模型的过程。随着硬件设备的多样化，特定硬件上高效运行的模型可能在另外一种硬件上性能不佳。异构编译通过转换模型使其适应不同硬件的特性，从而实现模型的可移植性和性能优化。

总之，模型结构优化主要关注在模型内部减少复杂性和提高效率，而异构模型编译着重于跨不同硬件平台的模型可移植性和性能优化。这两个方面在将深度学习模型部署到实际生产环境中非常关键。

在一种实现方式中，体验分层模型是预先在服务端训练完成并发布至内容分发网络，其中，决策引擎根据体

验分层模型的配置信息，访问内容分发网络，加载体验分层模型至本地。例如，通过端智能配置平台的配置能力，将模型及相关控制参数配置在平台上，然后，将模型保存在一个 DB 中，通过一个配置网关服务从该 DB 读取模型；在客户端启动之后，客户端自动从该配置网关服务中加载模型，可参考图 4 及说明。

例如，体验分层模型预测结果分为运行状态良好、运行状态一般、运行状态差三个级别，针对后两个级别，可以采取路径简化、富交互降级、功能降级等处置方案。路径简化可以是关闭非关键功能或降低非关键功能的优先级，例如，如果非关键功能对用户下单或者完成订单履约没影响，可以直接关闭这些功能，或者如果非关键功能对用户下单或完成订单履约影响较小，可以降低这些功能的优先级。富交互降级可包括降低动画帧率、降低图片分辨率、简化交互效果等方面。降低动画帧率：动画帧率越高，对机器性能要求越高，因此可以根据用户机器的性能实时调整动画帧率，以达到流畅运行的效果；降低图片分辨率：根据用户机器的性能实时调整图片分辨率，减少图片渲染对内存的消耗；简化交互效果：一些交互效果对机器性能要求较高，可以根据用户机器的性能实时简化或关闭这些交互效果（如 lottie 动画效果）。功能降级，是指对客户端中非阻塞用户下单或完成订单履约的功能进行降级。

本申请实施例提供的体验分层方案中，是一种跨端、即时、动态、智能、全链路的低性能设备判别策略，它不仅关注于设备的静态数据，而且会从设备运行过程中的各个维度数据出发，通过实时感知计算，为低性能设备的判定提供更精准的数据依据。与传统评估低性能设备方式相比，传统方式更固化、不够灵活，而本申请实施例注重与通过实时的数据分析和算法决策，从用户真实体验出发，提供一种更智能、全面和精准的评估方案，该方案突破了传统方案的局限，更具个性化、智能化与即时性的特点。

最后，结合图 8 介绍体验增强方案。

参见图 8，示出了本申请实施例提供的端智能体验增强方法流程图。

S801：启动客户端，向服务端发送端智能体验配置请求，获取到全链路动线预测模型；

S802：启动感知引擎，获取用户行为数据、用户基础数据、场景数据及性能数据；

S803：触发决策引擎，将获取的用户行为数据、用户基础数据、场景数据及性能数据输入至全链路动线预测模型，预测用户未来行为轨迹；

S804：运行触点引擎，根据用户未来行为轨迹的预测结果触发对应的体验增强处置方案。

体验增强，是指基于用户实时交互行为信息，利用全链路动线预测模型，实现实时用户未来轨迹预估，对用户可能访问的分包、页面进行预加载等干预，提升用户体验惊喜感。

针对体验增强方案，需要首先在服务端进行数据收集和处理，并训练完成全链路动线预测模型，然后，将全链路动线预测模型下发至前端，在客户端运行过程中，基于获取的目标数据，在前端实时地通过全链路动线预测模型进行未来行为预测，并基于未来行为预测结果进行分包预加载等处置。

关于数据收集和模型结构确定两个角度，与体验分层方案原理和处理过程类似，可参考体验分层方案相关描述，与之区别在于数据特征维度划分以及模型结构选取上。由于体验增强主要考虑预测用户未来行为，因此，数据特征维度主要包括用户行为数据、用户基础数据、场景数据以及性能数据，有关各种数据获取方式可参考前面描述。对于全链路动线预测模型的网络结构，通过研究和实践，可以选择计算能力较强的网络结构，例如深度学习网络，例如 LSTM 等，这种模型通过分析用户的历史行为数据，尝试预测用户接下来可能的操作或偏好，以此来预测用户的行为路径。由于深度学习模型通常涉及复杂的网络结构和大量的参数，其内部决策逻辑对于用户来说是不透明的，因此可采用黑盒模型的方式。因此在一种实现方式中，设置体验增强决策对应的全链路动线预测模型为黑盒模型模式，在客户端热启动阶段，运行黑盒模型模式的全链路动线预测模型。

关于将在服务端训练好的全链路动线预测模型进行发布和部署，以及客户端下载并在本地进行实时预测，可参考体验分层方案相关描述（图 7），其原理和实现过程类似，此处不赘述。

另外，在一种实现方式中，全链路动线预测模型是预先在服务端训练完成并发布至内容分发网络，其中，决

策引擎根据体验分层模型的配置信息，访问内容分发网络，加载体验分层模型至本地。例如，通过端智能配置平台的配置能力，将模型及相关控制参数配置在平台上，然后，将模型保存在一个DB中，通过一个配置网关服务从该DB读取模型；在客户端启动之后，客户端自动从该配置网关服务中加载模型，可参考图4及说明。

例如，通过全链路动线预测模型出用户未来行为轨迹，可以针对预测结果进行相应的体验增强处置，例如体验增强处置方案包括：根据体验决策得到用户未来行为轨迹，确定用户可能访问的分包资源或页面资源，对分包资源或页面资源进行预加载操作。

在一个具体场景下，可以针对频道分包进行智能预加载。在客户端界面中，首页下一跳页面通常为分包，硬性全量预加载不仅浪费流量，也会占用内存，甚至造成卡死。利用端智能体验增强方案，预测下一跳跳地址，按就近原则做好分包预加载，就能较好平衡内存与体验间的矛盾。

与上述在客户端应用的端智能体验实现方法相对应，本申请实施例还提供一种应用在服务端的端智能体验实现方法。参见图9，为应用于服务端的端智能体验实现方法流程图。

该端智能体验实现方法，用于针对端智能体验前端代码进行配置和触点，其中，所述前端代码适用于多类型客户端并在前端进行用户体验的感知、决策和处置，所述方法应用于服务端，包括：

S901：接收客户端发送的端智能配置请求，匹配出端智能配置并下发至客户端，以使得客户端执行端智能体验决策和处置；

其中客户端执行体验决策和处置包括：运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；并运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

S902：响应于回调函数的触发，接收客户端反馈的处置方案执行结果。

关于在服务端执行的端智能体验实现方法，可参考前文原理及描述，此处不赘述。

参见图10，示出了本申请实施例提供的一种端智能体验实现装置结构示意图。

该装置用于针对适用于多类型客户端的前端代码，在前端进行用户体验的感知、决策和处置，所述装置应用于多类型客户端中任意类型客户端中的一个客户端，包括：

配置请求单元1001，用于启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；

端智能执行单元1002，用于运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

本实施例中的各单元的具体示例可以参考上述实施例及可选实施方式中所描述的示例，本实施例在此不再赘述。

本申请的实施例还提供了一种存储介质，该存储介质中存储有计算机程序，其中，该计算机程序被设置为运行时执行上述任一项方法实施例中的步骤。

可选地，在本实施例中，上述存储介质可以被设置为存储用于执行以上方法实施例的计算机程序。

可选地，在本实施例中，上述存储介质可以包括但不限于：U盘、只读存储器（Read-Only Memory，简称为ROM）、随机存取存储器（Random Access Memory，简称为RAM）、移动硬盘、磁碟或者光盘等各种可以存储计算机程序的介质。

本申请的实施例还提供了一种电子设备，包括存储器和处理器，该存储器中存储有计算机程序，该处理器被设置为运行计算机程序以执行上述任一项方法实施例中的步骤。

可选地，上述电子设备还可以包括传输设备以及输入输出设备，其中，该传输设备和上述处理器连接，该输入输出设备和上述处理器连接。

可选地，在本实施例中，上述处理器可以被设置为通过计算机程序执行上述任一项方法实施例中的步骤。
可选地，本实施例中的具体示例可以参考上述实施例及可选实施方式中所描述的示例，本实施例在此不再赘述。

上述本申请实施例序号仅仅为了描述，不代表实施例的优劣。

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

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

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

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

 所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用，可以存储在一个计算机可读存储介质中。基于这样的理解，本申请的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得
20 一台计算机设备（可为个人计算机、服务器或者网络设备等等）执行本申请各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：U 盘、ROM、RAM、移动硬盘、磁碟或者光盘等各种可以存储程序代码的介质。

 以上所述仅是本申请的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本申请原理的前提下，还可以做出若干改进和润饰，这些改进和润饰也应视为本申请的保护范围。

权 利 要 求 书

- 1.一种端智能体验实现方法，其特征在于，用于针对适用于多类型客户端的前端代码，在前端进行用户体验的感知、决策和处置，所述方法应用于多类型客户端中任意类型客户端中的一个客户端，包括：
- 启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；
- 5 运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。
- 2.根据权利要求 1 所述的方法，其特征在于，所述处置方案包括：体验监测处置方案、体验分层处置方案，和/或，体验增强处置方案；
- 10 所述体验监测处置方案包括：根据体验决策确定发生复杂事件时，以 UI 交互方式在界面中进行实时消息推送和/或进行页面跳转操作；
- 所述体验分层处置方案包括：根据体验决策确定设备运行状态为低运行状态时，进行接口加载优化和/或页面内容动态调整方案；
- 所述体验增强处置方案包括：根据体验决策得到用户未来行为轨迹，确定用户可能访问的分包资源或
- 15 页面资源，对所述分包资源或页面资源进行预加载操作。
- 3.根据权利要求 2 所述的方法，其特征在于，还包括：
- 设置不同处置方案对应不同的优先级，根据优先级确定不同处置方案之间的执行逻辑关系；其中，设置体验分层处置方案对应第一优先级；根据体验分层处置方案的处理结果，确定是否执行体验监测处理方案
- 和/或体验增强处置方案。
- 20 4.根据权利要求 2 所述的方法，其特征在于，还包括：
- 通过白盒模型或黑盒模型进行体验监控、体验分层或体验增强的体验决策，以及，根据白盒模型进行初步决策，并根据白盒模型初步决策结果确定是否利用黑盒模型进一步进行精细决策。
- 5.根据权利要求 4 所述的方法，其特征在于，还包括：
- 设置体验分层决策对应的体验分层模型为白盒模型模式，在所述客户端冷启动阶段，运行所述白盒模型模式的所述体验分层模型，根据所述体验分层模型的预测结果，确定是否继续执行端智能套件或关闭端智能套件；
- 25 设置体验增强决策对应的全链路动线预测模型为黑盒模型模式，在所述客户端热启动阶段，运行所述黑盒模型模式的所述全链路动线预测模型。
- 6.根据权利要求 1-5 任一项所述的方法，其特征在于，所述端智能体验配置包括复杂事件配置信息，所述复杂事件配置信息包括至少一项目标复杂事件的事件配置信息和时机配置信息；
- 30 所述启动端智能套件中的感知引擎，实时获取目标数据，包括：启动感知引擎，基于前端代码中的统一事件流处理工具，捕捉埋点数据，根据所述埋点数据确定是否触发了所述事件配置信息对应的目标复杂事件；
- 所述触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策，包括：
- 35 响应于目标复杂事件的触发，运行决策引擎，以依据所述时机配置信息针对所述埋点数据进行时机匹配，确定是否满足所述目标复杂事件的时机；
- 所述运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案，包括：在时机匹配成功后，运行触点引擎，以触发目标复杂事件对应的体验监测处置方案。
- 7.根据权利要求 6 所述的方法，其特征在于，在统一事件流处理工具捕捉到埋点数据之后，还包括：

- 对埋点数据进行客户端类型分析及格式分析，并对埋点数据进行标准化处理，使得不同类型客户端的数据具有统一的格式和结构；
- 其中，所述对埋点数据进行标准化处理，包括：解析埋点数据的埋点信息、事件 ID、参数信息；对所
述埋点信息、事件 ID 以及参数信息进行统一格式封装，转换为标准格式的事件信息。
- 5 8.根据权利要求 6 所述的方法，其特征在于，
所述决策引擎，是基于自研的复杂事件匹配规则库并被声明为静态函数的复杂事件处理组件，其中，
所述自研的复杂事件匹配规则库采用 JavaScript 开发以兼容多类型客户端。
- 9.根据权利要求 1-5 任一项所述的方法，其特征在于，所述端智能体验配置包括体验分层模型；
所述启动端智能套件中的感知引擎，实时获取目标数据，包括：启动感知引擎，获取设备硬件静态数
10 据、设备实时状态数据以及场景实时运行数据；
所述触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策，包括：
触发决策引擎，将获取的设备硬件静态数据、设备实时状态数据以及场景实时运行数据输入至所述体验分
层模型，预测设备运行状态；
所述运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案，包括：运行触点引擎，
15 根据设备运行状态的预测结果触发对应的体验分层处置方案。
- 10.根据权利要求 9 所述的方法，其特征在于，所述体验分层模型，是基于卷积神经网络对数据样本进
行特征提取、将提取的特征输入到深度神经网络模型中进行训练而得到的。
- 11.根据权利要求 9 所述的方法，其特征在于，
所述体验分层模型，是通过训练第一网络结构得到第一体验分层模型以及训练第二网络结构得到第二
20 体验分层模型，在所述决策引擎决策过程中，利用第一体验分层模型作为白盒模型进行白盒预筛以及通过
第二体验分层模型作为黑盒模型进行黑盒精筛，从而预测出设备运行状态。
- 12.根据权利要求 1-5 任一项所述的方法，其特征在于，所述端智能体验配置包括全链路动线预测模型；
所述启动端智能套件中的感知引擎，实时获取目标数据，包括：启动感知引擎，获取用户行为数据、
用户基础数据、场景数据及性能数据；
25 所述触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策，包括：
触发决策引擎，将获取的用户行为数据、用户基础数据、场景数据及性能数据输入至所述全链路动线预测
模型，预测用户未来行为轨迹；
所述运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案，包括：运行触点引擎，
根据用户未来行为轨迹的预测结果触发对应的体验增强处置方案。
- 30 13.一种端智能体验实现方法，其特征在于，用于针对端智能体验前端代码进行配置和触点，其中，所
述前端代码适用于多类型客户端并在前端进行用户体验的感知、决策和处置，所述方法应用于服务端，包
括：
接收客户端发送的端智能配置请求，匹配出端智能配置并下发至客户端，以使得客户端执行如下步骤：
运行前端代码中的端智能套件，其中，启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套
35 件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；并运行端智能套件中的触点引
擎，根据体验决策结果触发对应的处置方案；以及，
响应于回调函数的触发，接收客户端反馈的处置方案执行结果。
- 14.一种端智能体验实现装置，其特征在于，用于针对适用于多类型客户端的前端代码，在前端进行用
户体验的感知、决策和处置，所述装置应用于多类型客户端中任意类型客户端中的一个客户端，包括：

配置请求单元，用于启动客户端，向服务端发送端智能体验配置请求，获取端智能体验配置并存储到本地；

端智能执行单元，用于运行前端代码中的端智能套件，包括：启动端智能套件中的感知引擎，实时获取目标数据；触发端智能套件中的决策引擎，依据所述端智能体验配置，针对目标数据进行体验决策；以及，运行端智能套件中的触点引擎，根据体验决策结果触发对应的处置方案。

15.一种存储介质，其特征在于，所述存储介质中存储有计算机程序，其中，所述计算机程序被设置为运行时执行权利要求 1 至 13 任一项中所述的方法。

16.一种电子设备，包括存储器和处理器，其特征在于，所述存储器中存储有计算机程序，所述处理器被设置为运行所述计算机程序以执行权利要求 1 至 13 任一项中所述的方法。