深度学习推理侧模型优化架构探索

孟伟[1] 袁丽雅[1] 韩炳涛[2] 刘涛[2]

(1. 中兴通讯股份有限公司南京研发中心,南京 210012;
2. 中兴通讯股份有限公司天津研发中心,天津 300308)

摘要: 论述了深度学习推理侧模型优化的历史起源,阐述了模型优化加速的整体架构和创新应用,提出了推理侧模型优化业务的发展建议。

关键词: 模型优化; 人工智能; 模型加速

1 引言

人工智能(Artificial Intelligence,AI)是模拟、延伸和扩展人的智能,是在机器上实现的理论、方法、技术及应用,也是当前一门新兴的学科。人工智能发源于20世纪中叶,在1956年的达特茅斯学院会议上"人工智能"这个词正式出现在世界面前,科学家从这时开始真正踏上智能研究的道路。近年来,随着数据的爆发式增长、计算能力的大幅提升以及深度学习算法的发展和成熟,人工智能已经迎来其发展的第三个浪潮。这次浪潮的基本特征在于基于大数据和强大计算能力的机器学习算法已经在计算机视觉、语音识别、自然语言处理等一系列领域中取得了突破性的进展,基于人工智能技术的应用也开始成熟。人工智能发展迅速,其在图像识别、目标检测、机器翻译、语音识别等多个领域,表现均已超过人类,尤其是在AlphaGo战胜了李世石之后,人们对人工智能可能达到的成就有了新的认识。

一段时间以来中央政府密集部署新型基础设施建设,习近平总书记强调要加快推进人工智能、5G、物联网、工业互联网等新型基础设施建设。突如其来的新冠肺炎疫情给经济社会发展带来了前所未有的冲击,要摆脱疫情的不利影响,实现稳增长、调结构、扩内需的目标,新基建恰逢其时。当前,全国多个省份正快马加鞭布局新基建,人工智能也在这个"风口"加了一把力,步入了发展的高速车道。人工智能的核心作用是加快推动相关领域的智能化转变,大幅提升效率、降低运营成本、增加系统运转的安全性。人工智能更加突出了新基建的"新",突出了高科技内涵,对于优化经济结构,促进传统产业升级改造,打造全新的人机协同、跨界融合、共创分享经济生态作用及意义重大[1]。

当前,深度学习模型多种训练框架(如TensorFlow、PyTorch、MxNet、Caffe、PaddlePaddle等)训练时大多使用GPU,采用容器云方式,进行大规模并发计算以减少模型训练收敛时间; 而在模型的应用上,则要使用深度神经网络的推理能力,完成网络前向计算,并将此计算部署为应用服务,同时产生商业价值。

2 深度学习推理侧模型优化技术产生的背景

2.1 现有模型机制的不足

深度学习模型的训练和推理在计算、部署环境、优化、时延以及能耗方面存在巨大的差异。深度学习模型在推理时,只包含前向计算过程,部署环境也存在多样性。而用户的推理应用场景多种多样,有时候推理需要多个模型联合起来共同完成,而这些模型在训练阶段为了保证训练的效率,或者由于其他一些因素(如数据、独立优化等)可能是分别独立训练出来的。所以,在推理运行时,针对特定的模型部署,AI平台除了要提供完整的工作流支持外,还要更好地支持模型推理运行时的开发与部署。在一些特殊场景下,甚至需要多个模型协作,将多模型编排在一起联合完成推理。

原始的深度学习模型编排方式基于微服务级,即将多个模型分别部署到多个推理应用侧,用户推理时依次向应用侧发送请求,在收到上一个模型的推理结果后进行处理,再向下一个模型发送推理请求(见图1)。这种方式虽然最终能得到正确的结果,但是存在以下几点不足。

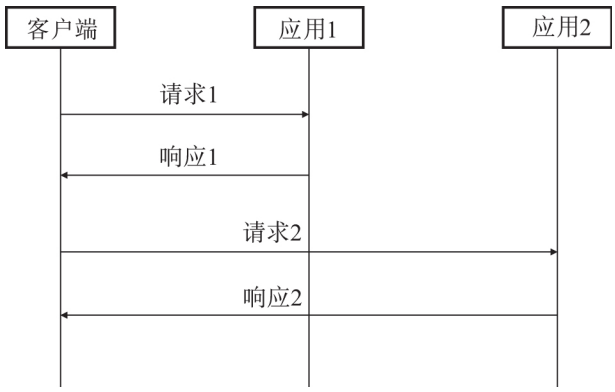

图 1 传统模型部署模式

(1) 整个时延较大,有试验证明,整个深度学习推理过程的大部分时间都花在了网络传输上,真正的推理运算耗时非常小。

(2) 客户端需要进行多次发送处理,复杂性高,甚至需要用户单独制作、上传特殊的镜像。

(3) 需要部署多个应用,资源消耗大,配置复杂。

2.2 推理侧模型优化的需求

各种算法在训练框架上训练后得到各种模型,其服务应用可能要部署在不同的硬件平台,使用不同的推理框架,对应用场景有着不同的要求,如低时延。所以,在训练中达到很好的收敛效果的深度学习模型距离真正的投入应用还有很多工作要做,原因主要基于以下几点。

(1) 针对不同设备的推理框架有很多,致使用户难以选择,且需要付出较多的学习成本。

(2) 不同应用场景的部署条件不同,有基于容器化部署的场景,也有基于嵌入式硬件部署的场景,同样的模型服务,不同的部署方案要掌握不同的技术。

(3) 根据性能需求有很多的模型调优工作。

(4) 推理服务应用于不同硬件,需要多类异构计算引擎的支持。

3 深度学习推理侧模型优化技术框架

推理侧模型优化技术是一种可以将深度学习模型从训练完成到部署再到特定硬件并提供应用服务的端到端工具链,其应用目的是为了将模型从研发状态快速部署到生产应用环境。推理侧模型优化技术需要和多种推理引擎协作,支持多种硬件,提供统一对外的推理接口,并提供多种灵活的部署方案,以及工程化的自适应参数优化方案,为用户提供快速、高性能的应用服务提供助力。推理侧模型优化包含以下具体技术。

3.1 模型优化技术

推理侧模型优化技术一般支持剪枝和量化。剪枝算法一般采用过滤器剪枝方案,对卷积层的剪枝效果表现出色。在量化上,可以采用小数据集量化和校准算法,例如仅需要 100 个样本图像,就可在不到 1 min 的时间内,完成 ResNet50 模型的量化过程,并且量化后精度损失很小。量化后的模型,可以生成为 TF lite 或 TensorRT 格式模型,运行在各类执行硬件上。

3.2 模型编译技术

推理侧模型优化模型编译器可以支持 TensorFlow、PyTorch、Keras 等多类训练框架,将其产生的模型编译为指定运行时和硬件环境依赖的模型文件,同时也支持 ONNX 文件格式的模型编译。模型编译器的软件架构设计可以以各模型表达方式和模型端到端编译为目标,形成 DAG 图,以算法形式完成图搜索过程,自动寻找最优的编译(模型文件转换)路径,后期可以和模型优化特性结合在一起使用,形成自动优化/编译过程。

3.3 推理引擎集成技术

推理侧模型优化推理引擎集成了多款深度学习模型的推理运行,如 TensorFlow Serving、TensorRT、OpenVINO、TensorFlow Lite 等,并且集成模型管理、版本更新策略控制等功能。推理引擎对外可以支持传输接口,例如 http 和 grpc 接口,并在支持例如 TensorFlow Lite 等框架运行时的嵌入式版本中,对外提供模型调用的 API 接口,为嵌入式系统提供更高效的推理效率。推理引擎同时需要支持多模型实例调度方案,为边缘侧资源受限的系统部署提供了有效的推理解决方案。

3.4 Benchmark 自动测试框架技术

自动测试框架是为了完成不同运行时间、相同硬件环境下的标准化性能测试。依据推理侧模型优化模块完善的模型编译和引擎的工程化积累,自动测试框架能够提供高效且自动化的测试流程,为业界标准化测试提供良好的工具。

4 深度学习推理侧模型优化技术的相关国际标准[2]

深度学习推理侧模型优化技术已经被成功地引用在 5G 智能网络架构中,并于 2020 年 8 月在国际电信联盟 ITU-T SG13 Q20 立项,现已成为 5G 网络迈向智能化的关键步骤。

4.1 模型优化需求

b-ITU-T Supplement 55 to Y.3170-series 中描述了包括 IMT-2020 在内的未来网络中的机器学习(Machine Learning,ML)用例。ITU-T Y.3172 提出了包括 IMT-2020 在内的未来网络中 ML 的体系结构框架。ITU-T Y.ML-IMT2020-MP 中描述了 ML 市场与包括 IMT-2020 在内的未来网络的集成。但是,未来网络中的模型服务仍需要解决以下挑战。

(1) 为异构硬件环境的 ML 模型启用可互操作的优化机制。

(2) 确保能针对不同用例场景灵活部署 ML 模型。

(3) 在服务模型和 ML 管道的其他组件之间提供有效的交互。

基于以上需求,可以提出一种包含参考点和序列图的服务框架架构,提供一种敏捷机制,以在包括 IMT-2020 的未来网络中为 ML 提供服务。

4.2 基于 5G 网络深度学习推理侧模型优化的框架

5G 网络深度学习推理侧模型优化器服务框架的主要功能组件如图 2 所示。其中,模型存储库是服务框架从中获取 ML 模型的存储库(注: 图 2 将模型存储库视为模型市场内部的通用组件)。模型适配器由模型优化器和服务引擎生成器组成,用于调整模型以在具有特定硬件环境的服务平台上使用。模型优化器负责 ML 模型的推理优化,根据 ML 模型的拓扑和部署环境为 ML 底层网络提供服务,以实现优化目标,其导出的模型格式可能与原始模型不同。服务引擎生成器生成用于 ML 模型的服务引擎,可在特定的运行时加载 ML 模型工件,以便 ML 模型可以在目标硬件上运行并根据模型配置执行推理; 可基于调度策略来调度模型,以便多个服务模型可以有效地执行推理; 可根据模型更新策略来管理模型的版本; 可将模型推论以及监视功能公开为 ML 管道可用的服务,可以是联机或批处理预测服务。服务平台是部署模型的平台,通过分配资源来运行服务模型。

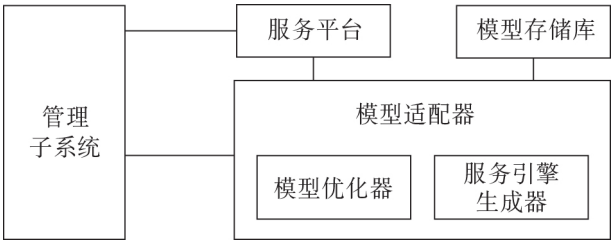

图 2 5G 网络深度学习推理侧模型优化器组件

将 ML 模型部署到服务平台的方法有如下几种。

(1) 嵌入式部署: 意味着 ML 模型是在 ML 应用程序内部构建和打包的,其中整个 ML 管道都在应用程序内进行管理。在这种方法中,ML 模型及其运行时库充当应用程序的依赖项,而如何执行推理则取决于实现。这样的部署减少了消耗模型推断的延迟,并且通常用于资源受限的场景(如 IoT 设备)中。

(2) 整体式 MLaaS(机器学习即服务)部署: 在 Monolithic MLaaS 部署方法中,将 ML 模型包装在服务引擎映像中,然后将其整体部署到服务平台上,以提供可被 ML 管道使用的模型推理服务。大规模服务 ML 管道的大型集中式服务平台可以选择这种部署方法。

(3) 模型独立的 MLaaS 部署: 独立于模型的 MLaaS 部署是指已经托管在服务平台上的服务引擎通过从存储模型的服务平台上的文件路径加载 ML 模型来提供模型推理服务的方法。这样的部署仅需要获取 ML 模型,当执行推理时,它将共享相同的服务引擎。

后两种部署模型的服务平台为服务引擎提供了常规支持,例如流量路由、扩展和监视、服务模型管理。

4.3 高级架构

电信网络中 ML 模型的服务架构与 b-ITU-T FGML5G MKT 中指定的网络中 ML 市场整合的架构一致(见图 3)。所有参考点均与 ITU-T Y.ML-IMT2020-MP 和 ITU-T Y.3172 一致。

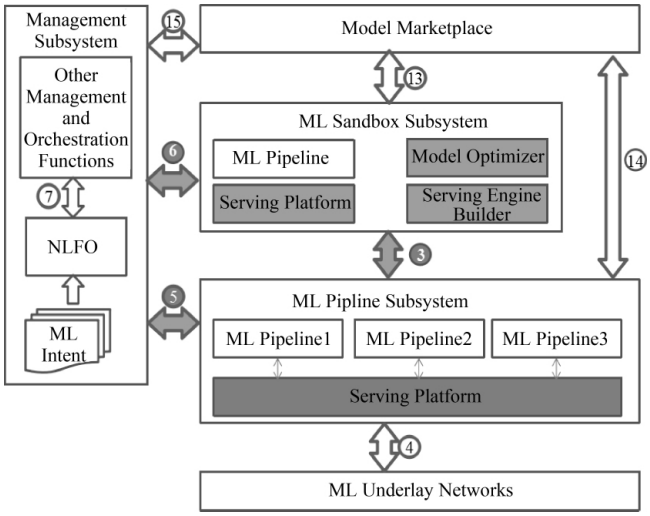

图 3 5G 机器学习模型优化高级架构

如图 3 所示,模型优化器是作为 ML 沙箱中的新

增强功能添加的，允许对内部 ML 市场中通过接口 13 获得的模型进行优化。优化的模型在 ML 沙箱子系统中的服务平台上进行部署和评估，然后通过参考点 3 作为 ML 管道子系统中的服务平台上的服务模型部署。

5 深度学习推理侧模型优化技术的开源实践

作为 LF AI 中首个聚焦深度学习模型推理阶段的项目，Adlik 的宗旨是使深度学习模型能够高效地运行在多种部署环境下。利用 Adlik，开发者可以方便地将主流训练框架如 TensorFlow、Keras、Caffe、PyTorch 等训练出的模型进行编译和优化，并根据模型部署的硬件环境自动选择优化的运行时环境，从而提升模型的推理效率，减少时延和能耗。

5.1 Adlik 应用场景

Adlik 在架构上可以分为模型优化模块、模型编译模块和推理引擎模块（Inference Engine）3 部分。训练好的模型，通过 Adlik 模型优化模块处理，生产优化后的模型，然后通过模型编译模块，完成模型格式转换，生成最终推理引擎支持的模型格式。

Adlik 可支持 3 种部署场景并提供相应的特性支持。

（1）云侧：Adlik 支持原生容器化部署方案，优化和编译完成的模型，可以和 Adlik 服务引擎镜像一起打包，发布为应用服务镜像，并在指定硬件的容器云上运行。

（2）边缘侧：Adlik 支持在启动的 Adlik 服务引擎服务上，加载优化和编译完成的模型，并支持模型版本管理、自动升级，以及多模型实例调度功能，以减少边缘侧计算资源的占用。

（3）端侧：Adlik 可以为用户提供 C/C++的 API 接口，支持用户直接在计算引擎上调用完成了优化和编译的模型，并提供模型编排能力，具备低延时和小体积的特性，可以在指定硬件上运行模型应用。

5.2 Adlik 在支持异构硬件计算上的设计思路[3]

目前，Adlik 支持的推理引擎包括常见的用于深度学习推理的 TensorFlow Serving、TensorRT、OpenVINO 以及 CNNA（FPGA 特定运行时），并计划支持 Tf Lite。而支持的异构硬件包括 GPU、CPU（x86）以及 FPGA，计划支持 CPU（ARM）。Adlik 在框架层上提供了多种形式，支持用户扩展异构硬件的运行和部署。

Adlik Serving 提供了模型的推理服务，以插件的方式部署和隔离各种运行时的环境。Adlik Serving 内置常见的运行时组件包括 TensorFlow Serving、OpenVINO、TensorRT、CNNA（FPGA 特定运行时）、Tf Lite 等，各类应用可按需加载，开箱即用。

部署推理引擎时，需要根据具体场景灵活地选择推理运行时及其相应的异构硬件。例如，在 CPU 嵌入式环境下部署时，因为只存在 CPU 环境，此时用户可以选择 TensorFlow Serving on CPU 或 OpenVINO on CPU 两种部署方式，如果环境使用 ARM 的 CPU 架构，那么也可以选择 Tf Lite on CPU（ARM）的部署方式。

更为高级地，用户也可以通过 Adlik 提供的 Serving SDK 开发用户自定义推理运行时环境，并在 Adlik Serving 框架下执行推理服务，满足极高的时延性的部署环境。Adlik Serving SDK 提供了模型上载、模型升级、模型调度、模型推理、模型监控、运行时隔离等基础模型管理的功能特性，及其用户定制与开发推理服务的 C++ API。在用户需要根据自己的需求定制开发自己的模型和运行时，Serving SDK 提供了标准的扩展点，以方便用户高效地定制新的模型和运行时环境。基于 Serving SDK，用户也可以开发组合式的模型，在进程内控制多模型之间的交互，而模型之间的运行时可以相互独立。例如，模型 1 的输出可以作为模型 2 的输入，模型 1 和模型 2 分别为 TensorFlow Serving 和 TensorRT 运行时的模型。

默认地，Adlik Serving/Serving SDK 不包含任何运行时组件，实现最小的依赖管理，应用根据部署环境灵活选择组装。也就是说，Adlik Serving/Serving SDK 可以提供不同运行时组合的镜像集合，应用根据具体的部署环境选择合适的基础镜像。

以新增一款运行时引擎为例，介绍 Adlik 的框架是如何支持创建新运行时的，这对很多有自定义运行时和指定硬件平台的场景有重要的意义。假设 A 是要支持的新运行时。

（1）定义 A_Model 类。这个指定的 A_Model 就是要创建的新运行时。

（2）定义具体计算引擎的实现类，A_Processor 类，A_Processor 类可以直接调用 A 计算引擎的执行 API。

（3）调用 Adlik 提供的 API 接口，将 A_Processor 类注册到 Adlik 运行时调度器，作为可调度的一个推理引擎。

（4）调用 Adlik 提供的 API 接口，将 A_Model 类注

册到 Adlik 系统,这样 Adlik 在启动后就能够使用基于 A_Model 的运行时来执行推理任务

在图 4 简单的流程示意图中,Server 收到推理请求后,Runtime 可以识别到 A_Model 运行时的请求数据,通过调度器和具体调度算法,找到注册在调度器的 A_Processor 实现类,最终调到 A 引擎提供的计算 API,完成整个推理过程。图 4 所示框架提供了丰富的管理和运行时调度功能,用户只需要关注计算引擎实现即可。

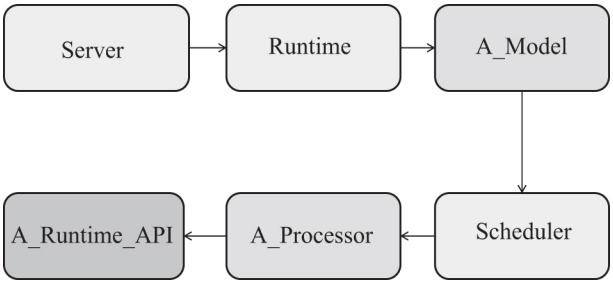

图 4 Adlik 流处理流程

实现上述代码后,用户可以根据指定硬件平台依赖的编译器对代码进行编译,即可生成指定硬件上可执行的推理应用程序,完成对异构硬件的支持。

5.3 Adlik 在支持 FPGA 异构计算上的实践

在计算密集型任务中(在深度学习模型推理领域),相比起仅具有数据并行能力的 GPU,FPGA 基于特定的门阵列硬件结构,可以同时实现数据并行和流水线并行,使得计算的延迟更小;对于特定结构的硬件电路,FPGA 的计算延迟也相对稳定。同时,与 CPU 和 GPU 相比,FPGA 具有较低的功耗,这在大规模的计算中具有一定的优势。此外,FPGA 还具有动态可重配置逻辑资源的功能,可以根据不同的配置文件,加载相应的编译文件,比 ASIC 更具灵活性。因此,FPGA 深度学习模型推理领域日益受到重视,并在卷积神经网络(Convolutional Neural Networks,CNN)的实现领域中占据了很重要的位置。

FPGA 的特点是善于进行并行乘加计算,所以在实施深度学习计算时,需要考虑将深度学习算法进行适当优化,将卷积、池化、下采样等大运算量的算子在 FPGA 中实现,从而提升整体计算引擎的效率和算力。实践中已在 xilinx zcu102 单板上实现了卷积神经网络中卷积算子、池化算子、下采样算子以及全连接算子。此外,Resnet 网络中涉及的 Scale 和 Batchnorm 算子也在建模中得到适当优化,得以在 FPGA 中实现。

Adlik 采用了笛卡尔架构进行研究和实践。ARM 与逻辑资源之间由 AXI 接口连接,网络配置信息由 axi-lite 接口传送,图像数据和权重则通过 DMA 由 PS 传送至 PL 外挂的 DDR 中,在卷积计算中由 PL 与 DDR 通过 AXI4 协议通信存取。

Adlik 在整体设计上使用了 Int8 量化和 DSP 复用技术、多点滑窗卷积计算以及多算子融合技术,结合 FPGA 传统设计中常用的流水线及乒乓存取技术,使加速器获得了更高的算力,以及更短的网络处理时间。

(1) Int8 量化和 DSP 复用技术

基于 Xilinx xczu9eg 系列芯片中的 DSP 资源的特点,可以实现 DSP 的乘加复用。该系列芯片的 DSP 资源可以实现 27×18 bit 的乘法计算,且某一级 DSP 的乘法输出结果可以级联到下一级 DSP 的加法输入端,从而实现乘加级联功能,无需额外使用逻辑资源搭建加法器。如果将输入的图像和权重信息量化为 Int8 格式,则可以使用权重移位的方式将某一权重左移 18 位后与另一权重相加,作为乘法器 27 bit 的输入数据,从而使用一个 DSP 实现两次乘加计算,可以做到算力加倍。

(2) 多点滑窗卷积计算

在传统的二维卷积计算中,每个时钟沿参与卷积的因子为 M 个输入通道图像中的某个单点与对应输出通道的卷积核的对应单点相乘加,得到对应 N 个输出通道中对应位置的单点部分和。使用多点滑窗的卷积方式,引入 PIX 变量,表示每个时钟沿里,每个输入通道的图像参与卷积的像素点数,传统卷积中的 PIX 通常为 1,即每个输入通道只有一个像素点参与乘加计算,而现在 PIX 可以取大于 1 的其他值,这样就使每个时钟沿参与卷积的图像像素点得到了伸展。

(3) 多算子融合技术

推理时的 BatchNorm 算子运算非常耗时,但由于其是线性运算,可以在建模初期将 BatchNorm+Scale 的线性变换参数融合到卷积层,替换原来的 Weights 和 Bias。与单独计算 BatchNorm 相比,这种算子融合大大减少了内存的读写操作,有效提高了处理帧率[4]。

(4) 流水线计算和乒乓加载技术

流水线技术和乒乓加载技术是 FPGA 设计中常用的手段。使用流水线技术可以使串行计算并行化,大大减少运算时间;乒乓加载技术则利用片上存储器实现了输入图像和权重的预加载,使输入图像由 FPGA

片外 DDR 传至片上的时间被卷积计算时间覆盖，同样起到了减少图像处理时间的作用。

6 结束语

在深度学习计算这类计算密集型的应用场景下，通用硬件设计中的某些特点，如复杂任务调度等，很可能会成为计算瓶颈，导致计算效率降低，所以针对应用部署硬件的特征完成异构硬件上的计算优化，在当前已经成为研究的热点。模型优化首先需要在架构上支持多类异构硬件和计算引擎，即需要在异构计算上做一些技术实践，同时也需要特定硬件上的计算优化技术的研究，来加速推动 AI 模型的工程应用。

参考文献

[1] 中国人工智能产业发展联盟. 聚焦“新基建”|人工智能支撑新冠肺炎疫情防控信息平台正当其时 [Z/OL]. (2020-05-15) [2020-08-09]. http://aiiaorg.cn/index.php?m=content&c=index&a=show&catid=8&id=122.

[2] ITU-T SG13-C-0917-R2. Architecture framework for serving ML models in future networks including IMT-2020 [S], 2020.

[3] GitHub. Tao Liu. adlik tutorials [R]. [2020-08-09]. https://github.com/Adlik/Adlik.

[4] Benoit Jacob, Skirmantas Kligys, Bo Chen, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference [N]. arXiv: 1712.05877, 2017-12-15.

作者简介：

孟伟　中兴通讯标准规划总工程师、开源总监，ITU FG-ML5G WG3 主席，Linux AI 基金会董事会董事，中国人工智能产业发展联盟总体组副组长，主要从事 IP 网络、人工智能、5G 网络自动化/智能化技术方面的研究工作，累计获得相关领域专利授权 20 余件

袁丽雅　中兴通讯标准工程师、Linux 基金会 Adlik 项目 TSC 主席，主要从事人工智能模型训练、算法框架、开源生态等方面的研究工作

韩炳涛　中兴通讯 AI 技术委员会主席、人工智能平台总工程师，主要从事 AI 系统架构、模型训练、推理加速及网络智能化应用方面的研究工作

刘涛　中兴通讯人工智能平台系统工程师，主要从事人工智能开源平台、算法、模型训练等方向的技术研究工作

Exploration of model optimization architecture on the inference side of deep learning

MENG Wei[1], YUAN Liya[1], HAN Bingtao[2], LIU Tao[2]

(1. Nanjing R&D Center of ZTE Corporation, Nanjing 210012, China;
2. Tianjin R&D Center of ZTE Corporation, Tianjin 300308, China)

Abstract: This paper discusses the origin of model optimization on the inference side, describes the overall architecture and innovative applications of model optimization acceleration, and puts forward the suggestions for the model optimization on the inference side.

Key words: model optimization; artificial intelligence; model acceleration

(收稿日期: 2020-08-09)