

# 面向航天异构平台的深度学习编译器 加速技术优化

刘功晗<sup>1,2</sup> 李 悦<sup>1,2</sup> 王晓玲<sup>2</sup>

1. 宇航智能控制技术国家级重点实验室, 北京 100854

2. 北京航天自动控制研究所, 北京 100854



**摘 要** 为了解决 TVM 编译器在航天应用过程中优化时间过长的问题, 针对 TVM 的自动搜索框架 Ansor, 提出一种新的基于 LightGBM 算法的成本模型来预测代码的运行时间。LightGBM 算法由于减少了分裂点数量、样本数量、特征数量, 使得新成本模型有效地降低了计算量, 在不损害准确率的条件下加快了模型的训练速度, 缩短了优化时间。实验结果表明, 在相同的优化任务下, 具有新成本模型的 Ansor 在优化时间上提高了 1.6 倍, 整体推理时间提高了 6.1%。

**关键词** 深度学习; 编译器; 时间优化; 成本模型; LightGBM 算法

**中图分类号** V448.21 **文献标识码**: A

**文章编号**: 1006-3242(2022)02-0060-06

## Optimization of Deep Learning Compiler Acceleration Technology for Aerospace Heterogeneous Platforms

Liu Gonghan<sup>1,2</sup>, Li Yue<sup>1,2</sup>, Wang Xiaoling<sup>2</sup>

1. National Key Laboratory of Science and Technology on Aerospace

Intelligence Control, Beijing 100854, China

2. Beijing Aerospace Automatic Control Institute, Beijing 100854, China

**Abstract** In order to solve the problem of too long optimization time of TVM compiler in the aerospace application process, a new cost model based on LightGBM algorithm is proposed to predict the running time of the code for the automatic search framework of TVM Ansor. The number of split points, the number of samples and the number of features are reduced by using the LightGBM algorithm, so that the new cost model can effectively reduce the amount of calculation, speeds up the training speed of the model without compromising accuracy and shortens the optimization time. The experimental results show that under the same optimization task, the optimization time and the overall reasoning time of Ansor with the new cost model can be improved by 1.6 times and 6.1% respectively.

**Key words** Deep learning; Compiler; Time optimization; Cost model; LightGBM algorithm

收稿日期: 2021-11-04

**作者简介**: 刘功晗(1997-), 男, 硕士, 主要从事人工智能、深度学习编译器方向研究; 李 悦(1984-), 男, 高级工程师, 主要从事控制科学与软件系统工程方向的研究; 王晓玲(1970-), 女, 研究员, 主要从事宇航领域软件开发与测试方向技术研究。本文通信作者。E-mail: wxl.amy@sohu.com

# 0 引言

目前,航天系统正向智能化方向迈进,各种智能算法的应用在航天装备中取得了优异的表现。比如,在导弹突防控制中有学者提出应用深度强化学习技术,设计了一种深度神经网络(Deep Neural Network,DNN)模型架构,实现了对弹道中段突防最优控制模型的逼近<sup>[1]</sup>。在目标检测中,通过 Faster RCNN 及其改进算法为遥感任务图像中复杂地理环境背景下的目标检测方法提供新思路<sup>[2]</sup>,在故障检测中,通过基于双向长短记忆神经网络的故障诊断模型,提高了对于发动机故障诊断的准确率<sup>[3]</sup>。然而,以 DNN 模型为例,其优异表现是以高计算复杂性为代价的,尤其是对实时性有着高要求的航天领域,更需要通过调用底层加速库来提高 DNN 模型的实现效率,如 PyTorch 和 TensorFlow 采用人工优化内核如 Intel MKL-DNN 或是 Nvidia cuDNN 作为后端。然而,随着新兴算子的不断增加、网络模型的不断扩大,人工优化的加速库很难适配不断增加的 DNN 模型,进而导致其可能无法满足航天装备对于实时性的要求,也极大阻碍了 DNN 模型在航天装备异构平台上的部署,限制了航天智能技术的进一步发展。

为了解决这一难题,我们通过引入深度学习编译器 TVM<sup>[4]</sup>来保证航天装备对于实时性的高要求,提高部署的灵活性。Ansor<sup>[5]</sup>是一个被集成到 TVM 中的自动搜索框架,为深层神经网络生成高性能的张量程序。编程人员只需要提供网络模型,Ansor 会自动搜索出一套调度策略,再通过 TVM 的代码生成,就可以得到优化后的代码,实现与人工优化库相似甚至更好的性能。

虽然 Ansor 可以生成高度优化的 DNN 模型代码,但这个过程可能要花费很长的优化时间。例如,在 ResNet-18 中,通过 Ansor 生成的代码性能优于 PyTorch,但是高性能代码的生成可能需要几个小时甚至十几个小时。随着当前深度学习的研究不

断深入,网络模型的参数也到达了数百万甚至数十亿的规模,这使得减少 TVM 的优化时间的问题变得十分突出。此外,随着新的神经网络结构<sup>[6-7]</sup>的快速出现,以及在不同的航天场景下硬件平台的选择,使得相关人员被迫更频繁地优化网络。过长的优化时间给优化过程带来了极大的不便,甚至使当前基于编译器的解决方案受到了质疑。

通过研究我们发现,TVM 使用成本模型来预测每个程序在真实硬件上的运行时间,再通过搜索算法选择其中的最优解。以 Ansor 为例,它采用 XGBoost<sup>[8]</sup>作为成本模型进行预测,因此,XGBoost 预测的速度,对整个优化时间会产生重要影响。本文首先对相关的技术原理进行介绍,然后设计了以 LightGBM (Light Gradient Boosting Machine)<sup>[9]</sup>为预测方法的成本模型,替代原始的 XGBoost 模型,这是一种旨在得到类似或更好的优化质量但是优化时间更短的方法。最后通过实验,证明我们所提出的方法是更优的,它使我们能够以最高快 1.6 倍的速度优化 DNN 模型,而且得到了类似甚至更优的推理时间,打破了 TVM 在航天装备中的应用瓶颈。

# 1 相关技术

## 1.1 DL 编译器

目前,像 TVM 这样的 DL 编译器因为可以自动优化 DL 程序而变得流行。与传统编译器类似,典型的 DL 编译器就是将 DNN 模型抽象看成各种编程语言,并对其进行优化,最后生成后端可执行的高性能代码的过程。如图 1 所示,DL 编译器的结构主要包含 2 部分:编译器前端和编译器后端,中间表示(IR)横贯前端和后端,IR 是程序的抽象,用于程序的优化。DNN 模型在 DL 编译器中转换为多级 IR (Intermediate Representation),分为高级 IR 和低级 IR。基于高级 IR,编译器前端负责独立于硬件的转化和优化;基于低级 IR,编译器后端负责特定于硬件的优化、代码生成和编译。

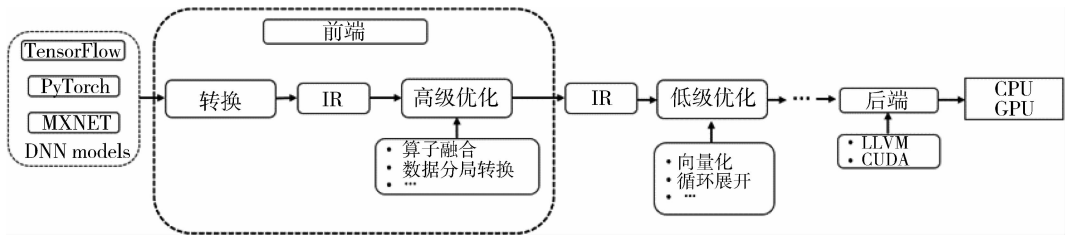


图 1 DL 编译器架构图

1.2 Ansor

TVM 最初需要编程人员手动或半手动的指定调度策略,然后通过 TVM 的代码 Pipeline 进行代码生成,但是通过手写或采用模板的优化方式可能会陷入局部最优,同时,想要写出一个好的调度策略,编程人员需要对硬件有深入的理解。

为了解决上述问题,TVM 引入了 Ansor,这样,编程人员只需要提供模型,Ansor 会自动选择出一套调度策略,再通过 TVM 进行代码生成,就可以得到优化后的代码,如图 2 所示。

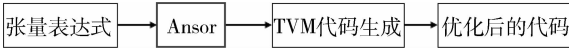


图2 Ansor 在 TVM 中的环节

Ansor 的代码生成流程可以分为以下几个步骤,如图 3 所示:

- 1) 任务调度:将完整的计算图划分为多个子图,对有潜力的子图进行重点优化。
- 2) 草图构建:提取算子中的高层次特征,对算子进行粗粒度的优化,确定代码的基本结构。
- 3) 随机注释:随机初始化 Tiling Size 和一些 for 循环的策略,获得计算图的完整表示。
- 4) 搜索评估:训练成本模型,根据成本模型对代码的性能进行评估,选取评估中高分数的一组实现,获得其在真实硬件测量中的正确结果,选取实际

性能最优的实现作为 Ansor 的输出。

总结来说,任务调度模块划分 DNN 模型的计算子图,确定各个子图的优化次数;草图构建模块搭建代码的整体框架;随机注释模块对细节进行填充;搜索评估模块训练一个模型,对生成的代码进行评价和更新,获得最后的代码生成结果。

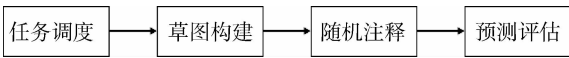


图3 Ansor 编译流程

2 成本模型设计

2.1 数据集

如 1.2 节所述,随机注释模块随机生成代码,但不考虑代码的性能。因此,待优化的计算可能十分复杂,遍历所有的随机注释模块来获取最优实现是不现实的。用成本模型预测随机注释模块生成代码的性能,需要通过提取完整程序的上下文中的特征为程序最里面的非循环语句构建特征向量,提取的特征包括算术特征和内存访问特征,这些特征可以通过遍历 TVM 的 TIR 获取,使用独热码对其进行编码,包含一个语句的所有列出特征的特征向量的长度是 164。将这些特征向量作为成本模型的输入数据,提取的部分特征如表 1 所示。

表 1 Ansor 提取的部分特征

指标	子类	特征集合
计算类	总操作数	float_mad, float_addsub, float_mul, float_divmod, float_cmp, float_match_func, float_other_func, int_mad, int_addsub, int_float_mul, int_divmod, int_cmp, int_match_func, int_other_func, ...
	关键字	vec_num, vec_prod, vec_len, vec_type
		unroll_num, unroll_prod, unroll_len, unroll_type
		parallel_num, parallel_prod, parallel_len, parallel_type ...
	线程块	blockldx_x_len, blockldx_y_len, blockldx_z_len threadldx_x_len, threadldx_y_len, threadldx_z_len vthread_len, ...
访存相关	内存访问量	Acc_type, Bytes, unique_bytes Lines, unique_lines Stride, ...

2.2 模型设计

XGBoost 是 GBDT<sup>[10]</sup> 的一种高效实现,与 XGBoost 相比,LightGBM 模型采用三大思想加速模型

的预测过程。首先采用直方图算法解决分裂点数量过多的问题;其次采用单边梯度抽样算法解决样本数量过多的问题,最后采用特征互斥捆绑算法解

决特征数量过多的问题。模型整体设计如图 4 所示。首先,模型将目标程序  $P$  通过随机注释模块提取特征  $\tau$ ,运行时模块获取对应样本的正确结果  $D_\tau$ ,构成初始训练集作为输入,对其进行指定次数的迭代训练,提高模型的预测精度。对于新特征  $\tau^*$ ,预测其输出  $D_{\tau^*}$ 。新特征  $\tau^*$  被发送到真实硬件中用于运行时测量,得到真实值  $D_{\tau^*}$ ,更新成本

模型以增强对后续迭代的探索。经过指定次数的迭代,选择出具有最短耗时的  $P(\tau)$  作为本层的输出。在优化 DNN 网络时,用于测量的特征通常少于 30000 个,在这样的数据集上训练一个 LightGBM 模型非常快,所以我们每次都训练一个新的模型,而不是进行增量更新。

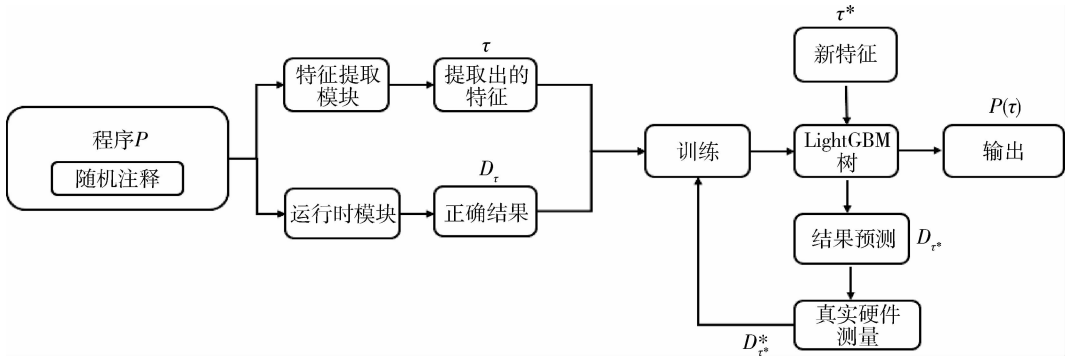


图 4 LightGBM 成本模型的整体设计

LightGBM 算法使用带深度限制的 leaf-wise 的叶子生长策略,与深度增长算法相比,叶向增长算法收敛速度更快。然而,如果模型参数设置不当,叶向生长可能会出现过拟合现象,导致预测精度降低。本文采用 Scikit-Learn 的 GridSearchCV 函数实现参数搜索工作。在指定的参数范围内,按照步长依次调整参数,遍历所有可能的参数组合,挑选验证误差最小的超参数组合返回,最终结果如表 2 所示。

Ansor 的搜索评估模块使得优化能够更快地收敛,减少了遍历时间,在采用 LightGBM 后的搜索评估模块整体流程如图 5 所示。其中,可以看到特征提取的输入有 3 个来源:1) 随机注释模块的样本,保证了样本的丰富性;2) 之前迭代中获取的注释模块的优秀样本,有助于训练得到准确度更高的 LightGBM 模型;3) 优秀样本的变种,根据进化算法,好的样本“突变”往往会更容易得到好的样本,因此采取对好样本的参数进行微调的方式扩充样

本集。提取的特征用来训练 LightGBM 模型,得到程序  $p$  及其对应的性能  $Perf(p)$ ,同时该模型又能对注释模块产生的样本进行一次初筛选,将评分高的样本通过 TVM 运行时模块获得相应的正确结果 Runtime ( $Perf(p)$ ),根据正确值可以选择出性能更优的代码,即整个 Ansor 的输出,整个算法至此结束。该输出可以无缝转换成调度策略,省去了手写调度策略的麻烦,并直接通过 TVM 代码生成模块 (TVM CodeGen) 进行代码的生成。

表 2 模型重要参数及优化结果

参数	含义	终值
max_depth	树的最大深度	6
feature_fraction	每次迭代时使用的特征比例	0.41
bagging_fraction	每次迭代时用的数据比例	0.48
num_leaves	叶子节点数量	10
min_data_in_leaf	叶子可能具有的最小记录数	10
Lambda_l1/lambda_l2	正则化参数	0.001/1.0

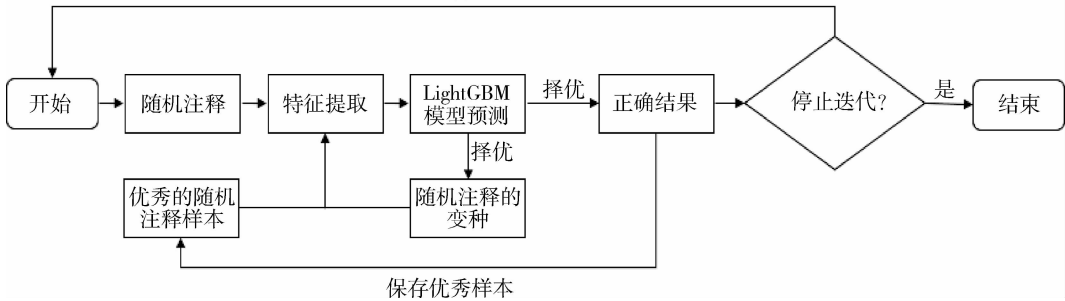


图 5 搜索评估模块整体流程

3 实验评估

在本节中,通过实验评估 LightGBM 模型,寻找它能够有效加速优化时间的答案。用 Python 编写 LightGBM 模型,利用 scikit-learn 实现模型的参数优化。从优化时间和推理时间 2 个方面评估。在优化时间方面,分别比较了 Ansor 使用 XGBoost 和 LightGBM 两种成本模型在搜索高性能代码时花费的时间。在推理时间方面,从局部的代码生成到整体的网络推理进行比较。在下面的内容中,把 Ansor 利用 XGBoost 算法的成本模型简称为 xgb,利用 LightGBM 算法的成本模型简称为 lgb。

3.1 优化时间

表 3 展示了 Ansor 使用 2 种不同的成本模型在 GPU 上的优化时间。分别比较了 xgb 和 lgb 在 ResNet-18, VGG16, 和 SqueezeNet1\_1 上端到端的优化时间。在表 3 中,“num\_trials”是在调优过程中的迭代次数,适当增加这个值有利于搜索的充分收敛。对于 ResNet-18, Ansor 将其总共划分成 18 个任务进行调优,图 6 展示了分别使用 lgb 和 xgb 调优该网络所花费的时间。由表 3 和图 6,总体而言, lgb 实现了 1.6 倍的提速。这是因为通过使用 LightGBM 模型,减少了特征的数量,过滤掉了小梯度的样本,并将遍历方式从遍历样本变为遍历直方图的形式,这些方法有效降低了时间复杂度,减少了大量不必要的计算。

表 3 优化时间比较				
网络模型	num_trials	xgb	lgb	加速比
Squeezenet	20,000	34608.55s	27910.12s	1.24 ×
VGG16	25,000	37743.53s	32813.11s	1.15 ×
ResNet18	30,000	56181.63s	35113.52s	1.60 ×

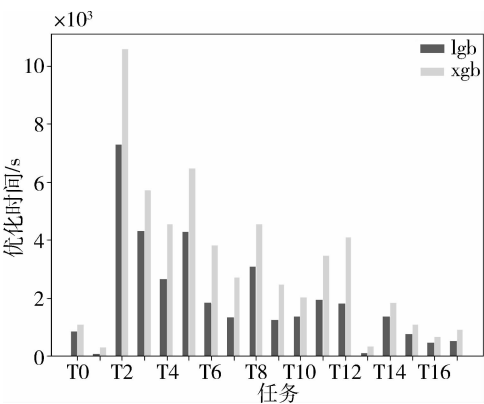


图 6 RestNet-18 网络优化时间比较

3.2 推理时间

首先,分别对使用 xgb 和 lgb 生成代码的性能进行比较,如图 7 所示,其中 y 轴表示调优后的代码在真实硬件中测量得到的 GFLOPS, x 轴表示迭代次数。分别选取 2 个任务:图 7(a)是在 CPU 上对矩阵乘法进行调优,图 7(b)是在 GPU 上,选取了 ResNet-18 中最后一层的卷积进行调优,为使 2 个任务均能充分收敛,取 num\_trials = 1000. 最终结果如图 7 所示。可以看出:首先,使用 lgb 调优后的代码性能与 xgb 类似,有时甚至可以超过 xgb,证明了 lgb 在缩短优化时间的同时保证了生成代码的质量。其次,通过 lgb 进行预测时,收敛速度更快。更快的收敛速度对于调优 DNN 模型以获得更好的性能至关重要。

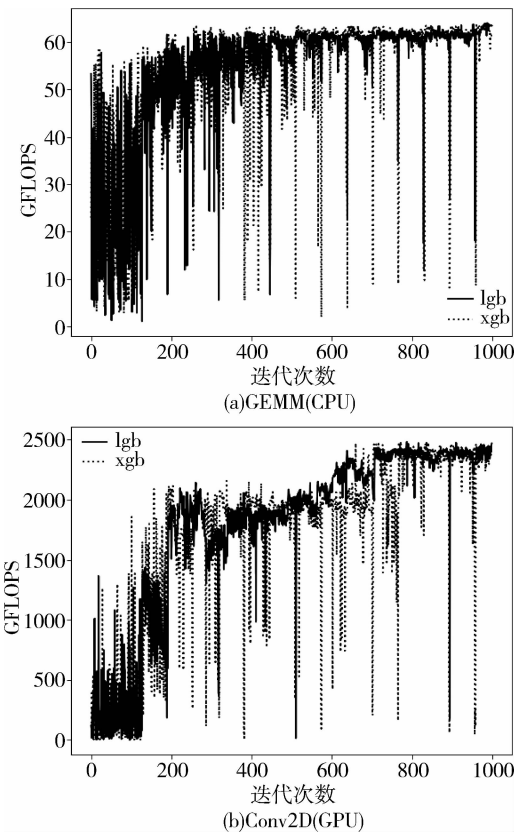


图 7 代码性能比较

图 8 比较了在 GPU 上 ResNet-18, VGG16 和 SqueezeNet1\_1 三种网络的推理时间,用目前最流行的深度学习框架 PyTorch 作为测量的基线。总的来说, lgb 的推理速度比 xgb 快了 6.1%, 比 PyTorch 快 90.1%。这一方面是由于两者决策树的生成方式不同, XGBoost 算法采用按层生长的策略( level-wise ), 不加区分的对待同一层叶子,使得部分收益很低的

叶子也进行了搜索和分裂。LightGBM 使用带有深度限制的叶子生长算法(leaf-wise)。因此在分裂次数相同的情况下,LightGBM 可以降低更多的误差,得到更好的精度。另一方面,lgb 和 xgb 在 SqueezeNet 上的推理时间显著降低,这或许是因为通过 Ansor 自动搜索的策略与 PyTorch 相应的加速库相比,更适配底层硬件。

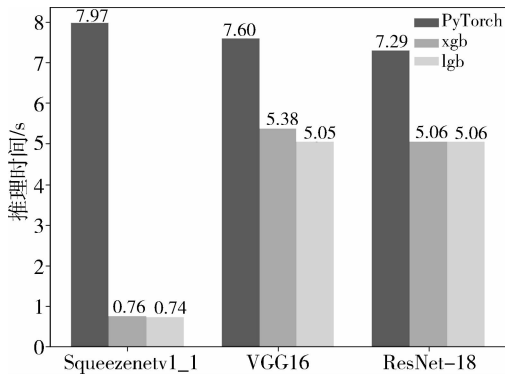


图 8 推理时间比较

4 结束语

随着航天智能技术的不断发展,对于 DNN 模型的需求也不断增加,TVM 的出现满足了航天装备对实时性的要求,有效降低了 DNN 模型在航天异构平台上部署的难度。然而,虽然高度优化的代码可以通过 TVM 实现,但它们需要生成不同的张量程序,并在真实硬件上测量,才能得到最优的结果,导致优化时间过长的问題。在本文中,我们建议将 LightGBM 模型集成到 Ansor 中,该模型在保证精度的情况下,减少了优化过程的时间成本。因此,对于在不同的 DNN 模型中搜索高性能的张量程序,使用带有 LightGBM 的 Ansor 花费时间更少,优于之前基于 XGBoost 的成本模型,是一种更加高效的方法。该方法使得在航天异构平台中,基于 TVM 解决实时性和灵活性的问题成为一种可能。

参 考 文 献

[ 1 ] 马子杰,高杰,武沛羽,等. 用于巡航导弹突防航迹规划的改进深度强化学习算法[J]. 电子技术与应用,

2021,47 ( 8 ): 11-14, 19. ( Ma Zijie, Gao Jie, Wu Peiyu. An improved deep reinforcement learning algorithm for cruise missile penetration path planning [ J ]. Application of Electronic Technique, 2021, 47 ( 8 ): 11-14, 19. )

[ 2 ] 王露荻,解月江. 基于域适应 Faster RCNN 的复杂背景目标检测 [ J ]. 航天控制, 2020, 38 ( 1 ): 63-69. ( Wang Ludi, Xie Yuejiang. Complex background object detection based on domain adaptive Faster-RCNN [ J ]. Aerospace Control, 2020, 38 ( 1 ): 63-69. )

[ 3 ] 刘利军,雷宇,余臻. 双向 LSTM 模型在航空发动机气路故障诊断的应用 [ J ]. 航天控制, 2020, 38 ( 5 ): 67-72. ( Liu Lijun, Lei Yu, Yu Zhen. The Application of bidirectional lstm model in aero-engine gas path fault diagnosis [ J ]. Aerospace Control, 2020, 38 ( 5 ): 67-72. )

[ 4 ] Chen T, Moreau T, Jiang Z, et al. TVM: An automated end-to-end optimizing compiler for deep learning [ C ] // In Advances in Neural Information Processing Systems, Montreal, Canada, October 8-10, 2018.

[ 5 ] Zheng L, Jia C, Sun M, et al. Ansor: Generating high-performance tensor programs for deep learning [ C ] // Proceedings of the 14th Symposium on Operating Systems Design and Implementation, November 4-6, 2020.

[ 6 ] Gao S, Cheng M-M, Zhao K, et al. Res2net: A new multi-scale backbone architecture [ J ]. IEEE transactions on pattern analysis and machine intelligence, 2019, 43 ( 2 ): 652-662.

[ 7 ] Wu Z, Shen C, Van Den Hengel A J P R. Wider or deeper: Revisiting the resnet model for visual recognition [ J ]. Pattern Recognit, 2019, 90: 119-133.

[ 8 ] Chen T, Guestrin C. Xgboost: A scalable tree boosting system [ C ] // proceedings of the Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, San Francisco, California, USA, August 13-17, 2016.

[ 9 ] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree [ C ] // Advances in Neural Information Processing Systems, Vancouver, Canada, December 6-12, 2017.

[ 10 ] Friedman JH. Greedy function approximation: a gradient boosting machine [ J ]. The Annals of Statistics, 2001, 29 ( 5 ): 1189-1232.