计算机科学
COMPUTER SCIENCE

基于特征重要性的深度学习自动调度优化研究

杨恒, 刘勤让, 范旺, 裴雪, 魏帅, 王轩

引用本文

杨恒, 刘勤让, 范旺, 裴雪, 魏帅, 王轩. 基于特征重要性的深度学习自动调度优化研究[J]. 计算机科学, 2024, 51(7): 22-28.

YANG Heng, LIU Qinrang, FAN Wang, PEI Xue, WEI Shuai, WANG Xuan. Study on Deep Learning Automatic Scheduling Optimization Based on Feature Importance [J]. Computer Science, 2024, 51(7): 22-28.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230500220

基于特征重要性的深度学习自动调度优化研究

杨 恒[1,2] 刘勤让[2] 范 旺[2] 裴 雪[2] 魏 帅[2] 王 轩[1,2]
1 郑州大学网络空间安全学院 郑州 450003
2 信息工程大学信息技术研究所 郑州 450002
(yh1423828145@163.com)

摘 要 随着深度学习和硬件架构的快速发展,模型和硬件架构的多样性导致采用手工优化方式实现深度学习模型的高性能部署面临严峻的挑战,因此现有的AI编译器框架通常采用自动调度的方法来实现这一过程。但是已有的TVM自动调度优化方法中存在着代价模型数据集不平衡以及调度时间过长的问题,为了解决这些问题,提出了一种基于特征重要性的自动调度优化方法。首先采用xgboost算法对特征重要性进行分析,然后基于重要性系数降低特征维度并对数据标签值进行重分配,以实现提高代价模型精度和优化自动调度效率的目的。实验结果表明,应用所提优化方法,使3种深度学习模型的自动调度时间缩短了9.7%~17.2%,推理时间最多缩短了15%。

关键词:AI编译器;自动调度;xgboost;特征重要性;深度学习
中图分类号 TP302

Study on Deep Learning Automatic Scheduling Optimization Based on Feature Importance

YANG Heng[1,2],LIU Qinrang[2],FAN Wang[2],PEI Xue[2],WEI Shuai[2] and WANG Xuan[1,2]
1 College of Cyberspace Security,Zhengzhou University,Zhengzhou 450003,China
2 Institute of Information Technology,Information Engineering University,Zhengzhou 450002,China

Abstract With the rapid development of deep learning and hardware architectures,the diversity of models and hardware architectures make the deployment for deep learning models with high performance manually become increasingly challenging. So current AI compiler framework often adopts automatic scheduling. Since the existing optimization to TVM automatic scheduling has such issues as unbalanced data sets in cost model and overlong scheduling time,an automatic scheduling optimization strategy based on feature importance is designed in this paper. First,the feature importance is analyzed through the xgboost algorithm. Then a strategy that reduce the data feature dimensions based on the importance coefficient and reassign the data labels is adopted to improve the precision of the cost model and optimize the efficiency of the automatic scheduling. Experiment results show that the proposed optimization method can reduce the automatic scheduling time of three kinds of deep learning models by 9.7%~17.2%,and reduce the inference time by up to 15%.

Keywords AI compiler,Automatic scheduling,xgboost,Feature importance,Deep learning

1 引言

深度学习的不断发展促进了各个领域研究的深入,为了简化深度学习模型的实现,已经推出Caffe[1],Tensorflow[2],PyTorch[3],MaxNet[4]等众多深度学习框架。与此同时,底层的硬件也在迅速发展,因此将深度学习模型高效地部署到不同的硬件后端变得极为困难。针对这一问题,目前已经有众多的硬件厂商设计了专用的深度学习优化库,如NVIDIA的cuDNN[5]和TensorRT[6]。Intel针对自身CPU和GPU设计了深度学习MKL-DNN加速库,同时已有一些专用库[7]以实现加速部署。虽然已经有众多的高性能优化加速库,但是这些加速优化库只针对专有硬件提供优化部署。深度学习编译器的发展是解决模型部署困难的关键,其主要目标是自动实现能与手工优化库相媲美的高性能优化。目前已出现众多深度学习编译器,如TVM[8],FlexTensor[9],Glow[10],nGraph[11]等,在这些深度学习编译器中,TVM凭借端到端的高性能自动优化成为当前主流的深度学习编译器。

尽管当前TVM已经实现从最初的半自动优化AutoTVM[12]到全自动代码优化Ansor[13]的转变,并且可以与部分优化库相媲美,但是由于自动调度需要搜索巨大的空间,

到稿日期:2023-05-30 返修日期:2023-10-13
基金项目:国家重点研发计划重点专项(2022YFB4401401);嵩山实验室项目(纳入河南省重大科技专项管理系)(221100211100-01)
This work was supported by the Major Project of National Key R&D Program of China(2022YFB4401401) and Program of Songshan Laboratory (included in the management of Major Science and Technology Program of Henan Province)(221100211100-01).
通信作者:刘勤让(tsaliuqr@163.com)

需要代价模型对庞大的搜索样本进行模型训练及预测，同时需要对预测的部分样本进行编译测试，对一个深度学习模型的优化往往需要几小时甚至十几个小时。针对调度时间过长的问题，目前大部分优化工作集中在优化搜索空间及替换代价模型以加速模型的训练，进而减少自动调度产生的巨大时间开销。对机器学习模型而言，一个数据集的好坏往往决定了这些模型的上限，目前 TVM 自动调度中存在着大量不平衡数据，这会影响调度性能。现有的研究聚焦于模型而非数据，仅通过更换代价模型并不能真正解决 TVM 当前存在的问题。

本文的贡献如下：

1）将优化聚焦于代价模型转移到聚焦于数据，提出了一种新的数据标签优化方法。

2）拟合了新的代价模型参数集成到 Ansor。

3）对 3 类主流的深度学习模型在优化后的系统上进行测试分析，验证了优化方法的有效性。

本文第 2 章介绍了研究的系统框架及研究动机；第 3 章介绍了本文的优化方法及新代价模型的训练；第 4 章对优化方法的有效性进行了验证；最后总结全文。

2　研究背景

2.1　系统框架

TVM 是一种开源的深度学习编译器框架，提供端到端的高性能编译优化，其整体架构如图 1 所示。编译器的输入是深度学习模型或 Relay DSL 定义的算子，在编译器的前端，编译器将模型转化为计算图的中间表示 Relay IR[14-15]，按照一定的规则进行目标硬件无关的计算图优化，输出优化后计算图。在编译器后端使用自动调度进行目标硬件相关的低级优化，优化后的低级 IR 根据不同的硬件后端生成相应的目标文件，完成模型部署。

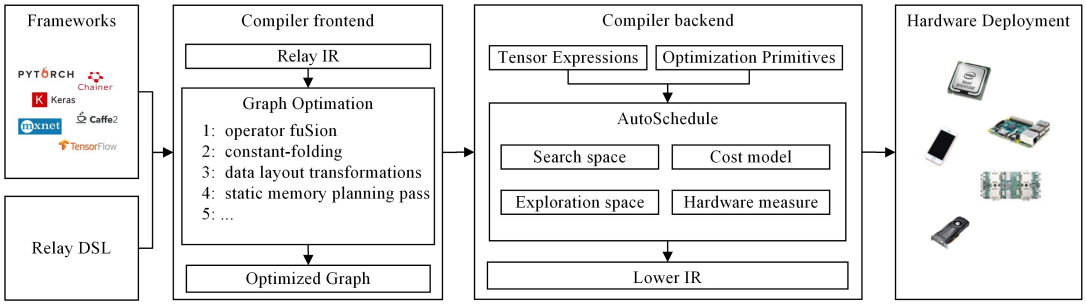


图 1　TVM 的架构

Fig. 1　Architecture of TVM

自动调度 Ansor 是 TVM 的一个子模块，用于实现代码的全自动优化，Ansor 的基本架构图如图 2 所示。

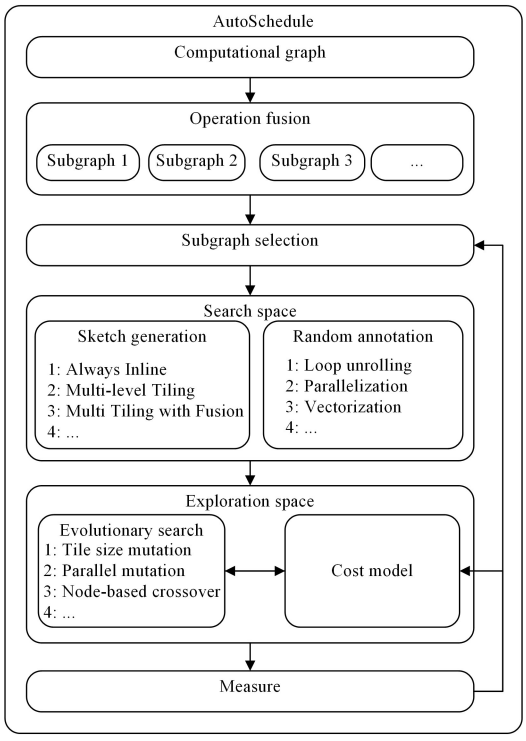


图 2　自动调度流程

Fig. 2　Flow of automatic scheduling

Ansor 的输入是经过优化后的计算流图。为了加快调度、减少搜索开销，算子融合模块会将网络模型划分为多个子图，每轮迭代子图选择模块会基于梯度选择一个最具有潜力的子图作为待优化任务。搜索空间模块会根据调度规则生成大量样本，使用遗传算法[16]生成候选样本，通过代价模型来对大量的样本进行预测，以加速调度过程。为了保证测试样本的准确性，硬件测试模块会对优秀样本结果进行硬件测试以获得真实性能，将测量的真实性能用于训练代价模型，同时用于指导下一轮迭代的优化。

2.2　研究现状及动机

目前对于自动调度时间过长的问题，其相关的优化工作集中在模型替换及搜索空间的优化。

文献[17]认为现有自动调度时间过长的问题是代价模型带来的巨大开销，使用 lightgbm 算法以替代 xgboost[18]算法，使得自动调度的时间有一定程度的缩短。文献[19]通过分析 TVM 自动调度优化空间的搜索过程发现，优化空间中存在大量无用的信息，并且在搜索过程中存在选择较优而不是更优节点的情况，提出了基于蒙特卡洛树的搜索空间优化方法，对 TVM 自动调度搜索空间进行优化。文献[20]对当前调度时间过长的问题提出了一种基于元学习的代价模型 Meta-Tune，通过这种代价模型能够以更少的参数及迭代次数达到更准确和高效的性能。文献[21]提出自动调度多轮迭代会带来大量冗余的开销，因此提出基于历史调度记录的调度框架来减少冗余数据的使用。文献[22]提供了巨大的自动调度测试记录用于训练统一的静态代价模型，以减少冗余数据带来

的开销，但对于通用的自动调度框架而言，网络模型和硬件架构的不断发展会带来数据漂移，难以保证模型的通用性。

本文主要是对 Ansor 中的 Cost model 模块进行优化。为了分析自动调度带来的时间开销，本文对自动调度期间的各个模块进行了测试，选取了 3 种深度学习模型作为测试网络模型，分别对重要的模块时间进行了相关测试，测试结果如图 3 所示。对于自动调度 Ansor 来说，模型训练和特征提取带来的时间开销占比约为 40%，而硬件测试模块带来的时间开销占比约为 55%，其他时间开销包括子图划分、搜索空间、模型预测所带来的时间开销仅占约 5%左右。

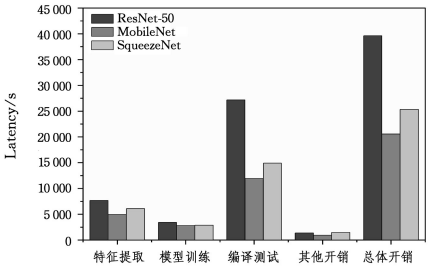


图 3 自动调度时间开销分布

Fig. 3 Time overhead distribution of automatic scheduling

机器学习数据集通常由特征和标签组成，特征是描述每个数据点的属性或特性的变量，而标签用于区分每一个数据点，标签应该具有单一性和可区分性，以便于模型的预测和泛化，数据集的构建能够直接影响模型的性能和准确性。代价模型作为自动调度的关键部分，对整个调度结果以及调度时间都会产生巨大的影响，但是目前的 Ansor 的数据集存在着标签值分配不均衡的问题。如图 4 所示，一个包含矩阵乘和计算最大值的子图调度的标签值分配方案，经过自动调度搜索空间搜索完成之后，会生成大量待测试调度，需要通过有监督学习模型进行性能评估，对自动调度而言其标签值用吞吐量表示。由于自动调度每轮迭代是对整个子图进行调度以生成调度方案，在硬件上测量真实值是对整个调度方案进行真实性能测量。然而，在样本特征提取的过程中需要对调度方案中的每个内层计算进行特征提取，现有的标签分配是对每一个样本共用总体的标签值，这种方式并不能给每一个样本分配有效的标签值，而且会使数据集存在大量不平衡数据，特别是当一个调度方案中存在多个内层计算时，会对代价模型产生较大的影响，进而影响调度结果。

为了解决这些问题，本文使用 xgboost 的 3 种特征重要性度量方式，基于特征重要性对数据标签重分配优化数据集，优化代价模型，进而提高自动调度的效率。

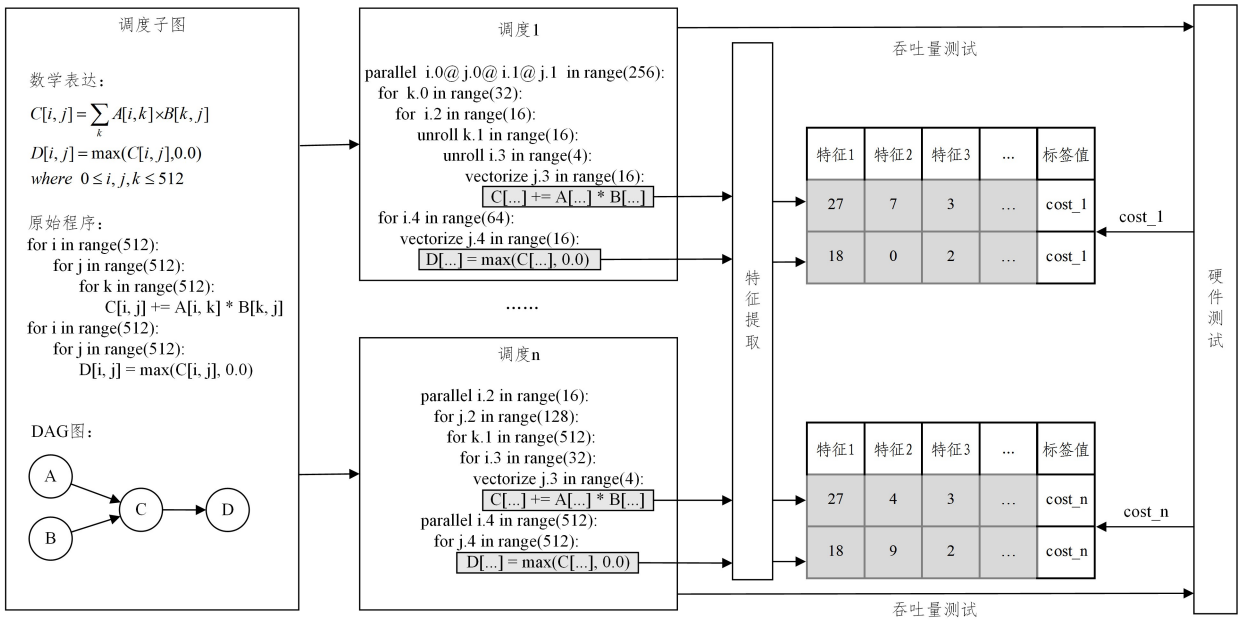


图 4 自动调度标签分配

Fig. 4 Label allocation of automatic scheduling

3 优化策略

3.1 标签重分配

本节重点介绍优化策略，整体的优化策略设计流程如图 5 所示。首先从 Tenset[22] 数据记录中选择一定量的样本测试记录进行特征提取，然后用具有大量冗余标签的非平衡数据集进行模型训练，并基于嵌入法特征重要性为每个特征设计相应的特征重要性系数，再基于特征重要性系数去除无用特征以及重新设计数据标签，使用平衡后的数据集重新拟合新的通用模型参数以集成到自动调度中进行实验分析。

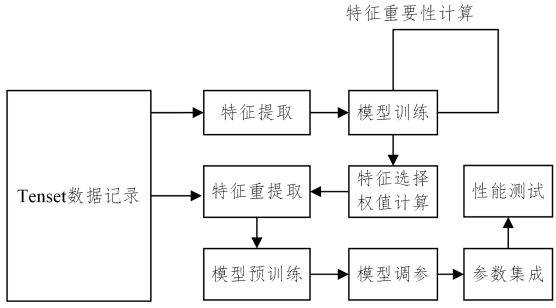


图 5 优化策略的流程

Fig. 5 Flow of optimization strategy

在自动调度样本评估过程中，在进行样本的特征提取时会提取到大量无用特征，并且数百万的样本特征提取所带来的时间开销也是巨大的。此外，由于每一轮迭代都会加入上一轮迭代真实测量的数据重新训练代价模型，剔除无用的特征会减少特征提取及模型训练的开销。而且基于特征重要性对标签重分配能有效解决数据不平衡的问题，提高代价模型的精度。

本文将 xgboost 的特征重要性的 3 种度量方法作为度量指标。以 Tenset 数据记录训练代价模型，然后遍历代价模型，分别用 3 种度量方式进行评估求均值，得到最终的特征重要性得分。3 种度量方法如下：

1）基于权重（Weight）的特征选择，遍历决策树中的每一个特征，将其作为分裂节点的次数，分裂次数越多的特征越重要，因此将分裂次数作为特征重要度量。基于权重的特征重要性的计算式如下：

$$H_i=\frac{1}{H_i}\sum_{k=1}^{T}\sum_{j=1}^{n_k}\delta(x_i,x_{k^j}) \tag{1}$$

其中，H_i 表示第 i 个特征基于权重的特征重要性；T 表示决策树的数量；n_k 表示第 k 棵树的节点数量；x_i 表示第 i 个特征；$\delta(x_i,x_{k^j})$ 表示遍历的特征与当前特征是否相等，若相等则返回 1，否则返回 0。

2）基于增益（Gain）的特征选择，遍历已训练的决策树结构中的每个特征，得到所有特征分裂后带来的增益，并将其作为每个特征的重要性。基于增益的特征重要性的计算式如下：

$$G_i=\frac{1}{H_i}\sum_{k=1}^{T}\sum_{j=1}^{n_k}\delta(x_i,x_k^j)\cdot(T_{k^j}-T_{k_j^{\text{left}}}-T_{k_j^{\text{right}}}) \tag{2}$$

其中，G_i 表示第 i 个特征基于基尼的特征重要性，T_{k^j}，$T_{k_j^{\text{left}}}$ $T_{k_j^{\text{right}}}$ 分别表示遍历的树节点和树分裂后的左右节点的损失。

3）基于覆盖（Coverage）的特征选择，对已完成训练的决策树进行遍历，覆盖度量方法关注特征在分裂节点中涉及的样本数量。覆盖值较大意味着特征涉及的样本较多，可能对整体预测有更大的贡献。基于覆盖的特征重要性的计算式如下：

$$C_i=\frac{1}{H_i}\sum_{k=1}^{T}\sum_{j=1}^{n_k}\delta(x_i,x_{k^j})\cdot O_{k^j} \tag{3}$$

其中，O_{k^j} 表示 k^j 覆盖的样本数量，C_i 表示基于覆盖的特征重要性。

为了避免不同度量方式间的差异性，对每种度量的特征重要性进行线性归一化后求均值，并将其作为标签重分配的权重系数。由于不同的特征会对调度的时间产生不同的影响，如 float_mul 表示浮点乘的运算量，会增加调度的时间；parallel_len 表示并行的粒度，会缩短调度的时间。因此，对每一个特征进行分析，赋予正负系数 α_i。

$H=\{\alpha_1H_1,\alpha_2H_2,\cdots,\alpha_mH_m\}$

H 为 i 个特征基于权重的重要性分数集合。

$G=\{\alpha_1G_1,\alpha_2G_2,\cdots,\alpha_mG_m\}$

G 为 i 个特征基于增益的重要性分数集合。

$C=\{\alpha_1C_1,\alpha_2C_2,\cdots,\alpha_mC_m\}$

C 为 i 个特征基于覆盖的重要性分数集合。

求得的重要性系数的计算式如下：

$$W_i=\sum_{S\in(H,G,C)}\frac{1}{3}\frac{S_i-\max(S)}{\max(S)-\min(S)} \tag{4}$$

其中，W_i 为得到的第 i 个特征的重要系数。本文基于特征重要度剔除不重要的特征，通过线性回归，以特征重要性为权重，样本相应特征值为真实数据，为每一个样本进行近似打分，对属于相同调度下的样本按分数比值重新进行标签划分，可以有效平衡数据集。划分后的标签的计算式如下：

$$label_k^n=\frac{\cos t_n\sum_{i=1}^{m}W_iD_{n^k}^i}{\sum_{j\in T_n}\sum_{j=1}^{m}W_iD_j^i} \tag{5}$$

其中，$label_k^n$ 表示第 n 个调度中的第 k 个样本的标签值，$\cos t_n$ 表示第 n 个调度的开销，T_n 表示第 n 个调度中的样本集合，$D_{n^k}^i$ 表示第 n 个调度下的第 k 个样本的第 i 个特征值，m 表示特征数量。

3.2　模型预训练

为了对巨大的搜索空间的样本进行测试，代价模型是不可或缺的，本文使用 xgboost 梯度提升树训练代价模型。为了保证代价的预测精度，对于每一个需要调度的网络模型，Ansor 都会根据当前调度的网络模型和目标硬件组成的调度样本形成当前代价模型的训练数据集，动态地重新训练相应代价模型。由于每一轮重新对模型调参的开销是巨大的，因此模型预训练的目的是拟合出通用的模型参数，而非训练出最终的静态模型。

使用的特征是基于原有特征优化后的，通过遍历 TVM 低级 TIR 语法树提取到的特征，主要包括计算量相关的特征及内存访问量相关的特征，部分数据特征如表 1 所列。使用的训练数据集是对 Tenset 公开测试记录进行特征提取优化后的数据集，Tenset 是来源于在 6 个硬件平台上对 120 个神经网络模型自动调度优化的数据测试记录，足以保证数据的多样性。

表 1　自动调度的部分特征

Table 1　Part features of automatic scheduling

特征类型	特征指标
计算量	float_addsub，float_cmp，int_cmp，float_mathfunc，int_addsub，int_mul，int_divmod …
优化粒度	vec_num，vec_prod，parallel_num，parallel_prod，alloc_inner_prod，num_loops，auto_unroll_max_step，alloc_size，alloc_prod，alloc_outer_prod …
内存访问量	B0.bytes，B0.unique_bytes，B0.lines，B0.reuse_dis_bytes B0.reuse_ct，B0.bytes_d_reuse_ct，B0.lines_d_reuse_ct B0.unique_lines_d_reuse_ct …

由于自动调度对一个神经网络模型的优化往往在（1000 * 子图数量）次硬件测试达到性能收敛，大部分神经网络模型经过子图划分后都会有约 25 个子图。此外，在自动调度中，由于不同的神经网络优化提取的数据集会有较大差异，每次对于一个新的需要优化的神经网络模型，都需要重新训练代价模型，自动调度大多在 25000 次测试时收敛。因此，为了避免数据量对模型产生影响，随机选取 Tenset 数据记录中的 25000 条测试记录进行特征提取，并将其作为模型的训练集

进行模型参数的拟合。数据的提取过程如算法 1 所示。

算法 1 数据提取

```
输入:(ts,w)
输出:(features,labels,ids)
1./*注册 Tenset 测试记录集 ts*/
2. samples=visit(ts)/*遍历语法树*/
3. normalization(samples)/*标准化数据*/
4. for(xs,y,id)∈ samples do/*遍历调度方案*/
5.     count ← 0,temp←[]
6.     for row∈x do
7.       add row to features∪ temp
8.       add id to ids
9.       count+=1
10.      if count=len(x) then
11.        /*对样本进行重要性计算*/
12.        scores=compute(temp,w)
13.        for score∈ scores do
14.        /*重新分配其标签值*/
15.          label=by_score(score)
16.          add label to labels
17.        end for
18.    end for
19. end for
```

4 实验评估

本实验使用的硬件平台为 AMD R7-5800H，软件环境为 ubuntu20.04，编译框架为 tvm0.8 及 python3.9。实验的相关参数信息如表 2 所列。选取了 3 种主流的深度学习模型进行优化，包括 ResNet-50 以及两种轻量级模型 MobileNet 和 SqueezeNet，对自动调度前后的时间、推理时间以及自动调度的收敛速度分别进行对比，同时将优化前和优化后的代价模型的 3 种评估指标进行性能对比。用 xgb 表示 TVM 原生的代价模型，用 opti-xgb 表示经过本文优化策略优化后的模型。

表 2 调度相关参数

Table 2 Scheduling related parameters

参数	参数值	参数说明
num_measure_trials	25000	自动调度分配的资源
num_measures_per_round	128	每轮迭代编译测试的样本数
evolutionary_search_num_iters	2	每轮迭代的变异次数
evolutionary_search_population	1024	每次变异的种群数

4.1 调度时间

表 3 列出了在集成 xgb 和 opti-xgb 代价模型的自动调度对 ResNet-50，MobileNet，SqueezeNet 进行端到端的调度优化的调度时间。为了使网络模型充分收敛优化，为每个网络模型分配了 25 000 次调度资源。通过实验数据可以看到，相比 xg-b，opti-xgb 为 TVM 自动调度带来了 9.7%～17.2%的性能提升，这是因为 opti-xgb 减少了无用的特征提取带来的时间开销，同时更低维度的特征加快了模型的训练。

表 3 调度时间开销

Table 3 Time overhead of scheduling

网络模型	实验次数	Xgb/s	opti-xgb/s	加速比/%
ResNet-50	25000	39618	35828	9.7
MobileNet	25000	20563	17208	16.3
SqueezeNet	25000	25345	20973	17.2

4.2 推理时间

图 6 给出了未优化的 MobileNet，ResNet-50，SqueezeNet 网络模型与在以 xgb 和 opti-xgb 为代价模型优化后的推理性能的提升对比，可以看到，经过优化后的模型都带来了较高性能提升，同时可以发现 opti-xgb 相比 xgb 为网络模型带来了最高 15%的性能提升。这是因为 opti-xgb 使用的是经过特征重要分析后的新特征，同时对标签值进行了重分配，使用的是更为平衡的数据集，为模型带来了一定的精度提升。

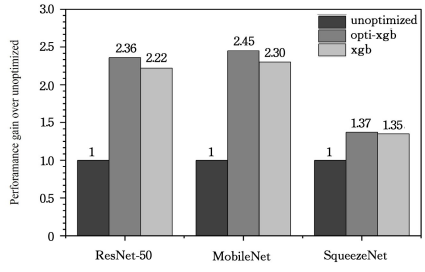
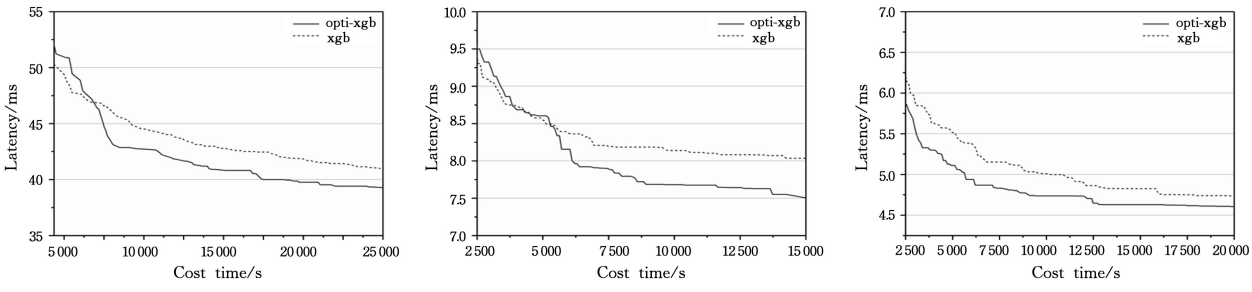


图 6 推理性能对比

Fig. 6 Comparison of inferring performance

图 7 给出了 ResNet-50，MobileNet 以及 Squee-zeNet 网络模型在相同优化时间内的性能收敛情况，以所有〔子图都完成了一次预热优化为初始阶段，可以看到，随着时间的增加，在调度资源充足的情况下，opti-xgb 相比 xgb 会带来更优的调度方案，并且可以更快速地到达收敛。这主要也是因为 opti-xgb 相比 xgb 模型带来了更高的性能。



(a) ResNet-50 (b) MobileNet (c) SqueezeNet

图 7 收敛速度对比

Fig. 7 Comparison of convergence rate

4.3　模型评估

由于本文是对数据集进行优化，对模型的评估需要保证数据集的一致性。为了准确地对代价模型进行评估，使用一种新的评估方式，本文对数据集的优化是对每一个组成调度任务的相应样本进行数据标签优化，并未改变每个调度的整体标签值，因此实验评估的粒度是每个调度整体而非单个样本。

TOP-K 是一种新的专门用于评估 Ansor 自动调度的评估指标。由于自动调度期间每轮迭代会对一批样本进行预测，对预测性能最好的前 K 个样本进行目标硬件性能测试，同时用于指导下一轮迭代，因此自动调度的模型本质上是一个排序模型，关注的不是模型对样本准确性的预测，而是能否预测出性能最好的前 K 个样本。使用新的评估指标 TOP-K 更能评估 Ansor 代价模型的精度。

本文使用的评估指标如下：

1）MAE 真实值与预测值的平均绝对误差，计算式如下：

$$MAE=\frac{1}{m}\sum_{n=1}^{m}\left|\sum_{i\in T_n}(y_i-\hat{y}_i)\right| \tag{6}$$

2）RMSE 真实值与预测值的均方误差，计算式如下：

$$RMSE=\sqrt{\frac{1}{m}\sum_{n=1}^{m}\left(\sum_{i\in T_n}(y_i-\hat{y}_i)\right)^2} \tag{7}$$

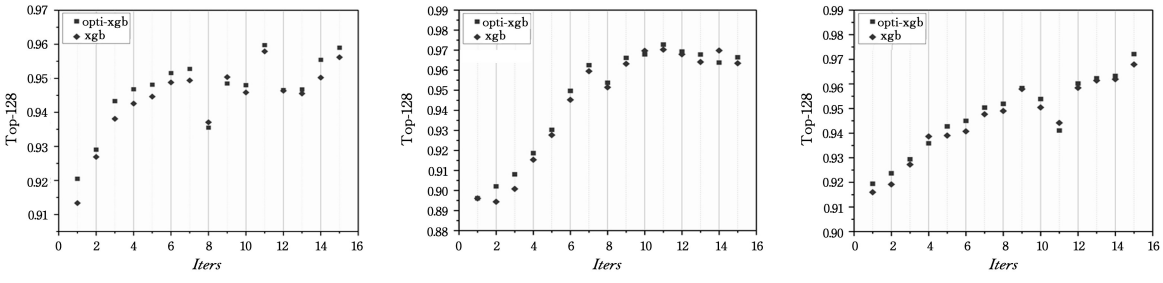
3）TOP-K 预测出的最优的前 K 个调度的真实值与测试数据最优真实值的接近程度，计算式如下：

$$TOP\text{-}K=\frac{1}{K}\frac{curve(P)}{max(Y)} \tag{8}$$

其中，m 表示预测的调度总数，T_n 表示组成第 n 个调度的样本集合，y_i 表示样本的真实值，$\hat{y}_i$ 表示样本的预测值，Y 表示真实标签集，P 表示预测最优的 K 个调度真实值排序序列，$max(Y)$ 表示最优调度标签值。对于 $curve(P)$ 中的第 n 个元素，都有：

$$f(n)=\max(P[i]\text{ for }i<n) \tag{9}$$

由于自动调度对模型的调优过程是多轮迭代的，为了保证模型的精度，代价模型并不是静态的模型，而是在每一轮迭代时都会重新训练，因此分别以 ResNet-50，MobileNet，SqueezeNet 的 25000 条测试记录为数据集。由于每一轮会挑选出最优的 128 个样本用于编译测试，子图预热一般需要 30 轮左右，因此初始数据集设为 3840 个调度数据，每一轮增加 128 个调度数据，并且以当前迭代下数据集的后 20%数据为测试集，剩下数据为训练集，进行多轮模型评估，总共 165 轮迭代。以 10 为间隔选取了其中的 15 次迭代下模型的 TOP-128 指标进行可视化分析，图 8 给出了其中的 15 轮迭代中 TOP-128 的对比情况。可以看到，在 TOP-128 评估指标下，opti-xgb 会带来比 xgb 更优的性能，只有在部分情况下略逊于 xgb，而且随着迭代的不断进行，模型的精度整体也呈上升的趋势。



(a) ResNet-50　(b) MobileNet　(c) SqueezeNet

图 8　模型 TOP-128 的对比

Fig. 8　Model comparison on TOP-128

表 4 列出了优化前后的调度代价模型在对 ResNet-50，MobileNet，SqueezeNet 神经网络进行优化时，TOP-32，TOP-64，TOP-128，MAE，RMSE 这 5 种评估指标在 165 轮迭代下的均值。通过图表可以看到 opti-xgb 在各种评估指标上均优于 xgb，并且在以不同模型为训练集时模型的精度基本一致，可以说明此代价模型保证了模型的通用性。通过本文的方案有效提高了代价模型的精度。

表 4　代价模型指标对比

Table 4　Comparison of cost model indicators

Network	Method	*TOP*-32	*TOP*-64	*TOP*-128	*MAE*	*RMSE*
Resnet-50	xgb	0.879	0.914	0.946	0.859	1.642
Resnet-50	opti-xgb	0.883	0.917	0.948	0.485	0.716
Squeezenet	xgb	0.870	0.912	0.943	0.569	0.732
Squeezenet	opti-xgb	0.878	0.916	0.947	0.459	0.568
Mobilenet	xgb	0.876	0.916	0.945	0.508	0.661
Mobilenet	opti-xgb	0.882	0.920	0.948	0.455	0.561

结束语　随着深度学习的快速发展，高性能部署成为当前深度学习面临的难题，需要大量人工优化，TVM 自动调度的发展一定程度上缓解了这一问题，但是当前 TVM 自动调度存在调度时间过长及优化效率不高的问题。基于此，本文设计实现了基于特征重要性的标签重分配方法，不仅降低了特征维度，减少了特征提取的时间开销，同时平衡了数据集，重新训练了代价模型，带来了更优的性能，使得 TVM 自动调度在优化时间及推理时间上都有一定的提升。

目前采用的方法主要是针对自动调度的代价模型进行相关优化，减少了模型训练及特征提取的资源开销。但 TVM 调度优化中的编译测试带来的时间开销依旧是巨大的，是整个自动调度资源开销的核心部分，下一步工作将对调度的架构进行改进，通过减少编译次数来进行相关优化。

参 考 文 献

[1]　JIA Y，SHELHAMER E，DONAHUE J，et al. Caffe：Convolutional architecture for fast feature embedding[C]// Proceedings of the 22nd ACM International Conference on Multimedia. New-York：Association for Computing Machinery，2014：675-678.

[2] ABADI M, BARHAM P, CHEN J, et al. Tensorflow: A system for large-scale machine learning[C]// 12th {USENIX} Symposium on Operating Systems Design and Implementation ({Osdi} 16). Sacannah, GA, USA: USENIX Association, 2016: 265-283.

[3] PASZKE A, GROSS S, MASSA F, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library[C]// Advances in Neural Information Processing Systems 32 (NeurIPS 2019). Vancouver, Canada, 2019: 8024-8035.

[4] CHEN T, LI M, LI Y, et al. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems[J]. arXiv: 1512.01274, 2015.

[5] CHETLUR S, WOOLLEY C, VANDERMERSCH P, et al. cuDNN: Efficient Primitives for Deep Learning[J]. arXiv: 1410.0759, 2014.

[6] NVIDIA. TensorRT Github repository [EB/OL]. [2020-02-04]. https://github.com/NVIDIA/TensorRT.

[7] GAO J, LIU S, HUANG Z Q, et al. Deep Neural Network Operator Acceleration Library Optimization Based on Domestic Many-core Processor [J]. Computer Science, 2022, 49(5): 355-362.

[8] CHEN T, MOREAU T, JIANG Z, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning [C]// 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18). Berkeley: {USENIX} Association, 2018: 578-594.

[9] ZHENG S, LIANG Y, WANG S, et al. FlexTensor: An Automatic Schedule Exploration and Optimization Framework for Tensor Computation on Heterogeneous System[C]// ASPLOS' 20: Architectural Support for Programming Languages and Operating Systems. NewYork: Association for Computing Machi-nery, 2020: 859-873.

[10] ROTEM N, FIX J, ABDULRASOOL S, et al. Glow: Graph Lowering Compiler Techniques for Neural Networks [J]. arXiv: 1805.00907, 2018.

[11] CYPHERS S, BANSAL A K, BHIWANDIWALLA A, et al. Intel nGraph: An Intermediate Representation, Compiler, and Executor for Deep Learning[J]. arXiv: 1801.08058, 2018.

[12] CHEN T, ZHENG L, YAN E, et al. Learning to Optimize Tensor Programs[J]. arXiv: 1805.08166, 2018.

[13] ZHENG L, JIA C, SUN M, et al. Ansor: Generating High-Performance Tensor Programs for Deep Learning[J]. arXiv: 2006.06762, 2020.

[14] ROESCH J, LYUBOMIRSKY S, KIRISAME M, et al. Relay: A High-Level IR for Deep Learning[J]. arXiv: 1904.08368, 2019.

[15] ROESCH J, LYUBOMIRSKY S, WEBER L, et al. Relay: a new IR for machine learning frameworks[C]// Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL 2018). New York: Association for Computing Machinery, 2018: 58-68.

[16] VIKHAR P. A Evolutionary algorithms : A critical review and its future prospects[C]// International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). IEEE, 2016: 261-265.

[17] LIU G H, LI Y, WANG X L. Optimization of Deep Learning Compiler Acceleration Technology for Aerospace Heterogeneous Platforms[J]. Aerospace Control, 2022, 40(2): 60-65.

[18] CHEN T, GUESTRIN C. Xgboost: A scalable tree boosting system[C]// Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD' 16). New York: Association for Computing Machinery, 2016: 785-794.

[19] ZHAO J Q. Research on Compiler auto-tuning Method Based on Deep Reinforce Learning[D]. Xi'an: Northwest University, 2022.

[20] RYU J, SUNG H. MetaTune: Meta-Learning Based Cost Model for Fast and Efficient Auto-tuning Frameworks[J]. arXiv: 2102.04199, 2021.

[21] MU J, WANG M, LI L, et al. A history - based auto-tuning framework for fast and high-performance DNN design on GPU [C]// 57th ACM/IEEE Design Automation Conference (DAC). IEEE Press, 2020: 1-6.

[22] ZHENG L, LIU R, SHAO J, et al. TenSet: A Large-scale Program Performance Dataset for Learned Tensor Compilers[C]// Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). 2021.

YANG Heng, born in 1998, postgraduate. His main research interest is deep learning compiler.

LIU Qinrang, born in 1975, Ph.D, professor, Ph.D supervisor. His main research interests include cyberspace security and chip design.

(责任编辑:喻藜)