计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210700199

GSO:基于图神经网络的深度学习计算图子图替换优化框架

苗旭鹏[1]　周　跃[1]　邵蓥侠[2]　崔　斌[1]
1 北京大学信息科学技术学院　北京 100871
2 北京邮电大学计算机学院　北京 100871
(xupeng.miao@pku.edu.cn)

摘　要　深度学习在各种实际应用中取得了巨大成功,如何有效提高各种复杂的深度学习模型在硬件设备上的执行效率是该领域重要的研究内容之一。深度学习框架通常将深度学习模型表达为由基础算子构成的计算图,为了提高计算图的执行效率,传统的深度学习系统通常基于一些专家设计的子图替换规则,采用启发式搜索算法来优化计算图。它们的不足主要有:1)搜索空间大,效率低下;2)缺乏可拓展性;3)难以利用历史优化结果。为了解决上述问题,文中提出了GSO,即一个基于图神经网络的深度学习计算图子图替换优化框架。该框架将计算图的子图优化建模成经典的子图匹配问题,基于计算图中算子的特征信息和计算图的拓扑结构信息,通过图神经网络模型来估计每种子图替换规则的匹配可行性和位置。基于与主流深度学习系统兼容的Python接口实现了GSO,实验结果表明:1)相比全量的子图替换规则,基于图神经网络的子图匹配预测可以最多减少92%的搜索空间;2)相比现有的启发式搜索算法,GSO可以更快地完成计算图子图替换优化(2倍以上),并使优化后的子图最多得到34%的加速。

关键词:计算图优化;子图替换;深度学习;图神经网络
中图法分类号　TP311

GSO:A GNN-based Deep Learning Computation Graph Substitutions Optimization Framework

MIAO Xu-peng[1],ZHOU Yue[1],SHAO Ying-xia[2] and CUI Bin[1]
1 School of Electronics Engineering and Computer Science,Peking University,Beijing 100871,China
2 School of Computer Science,Beijing University of Posts and Telecommunications,Beijing 100871,China

Abstract　Deep learning has achieved great success in various practical applications. How to effectively improve the model execution efficiency is one of the important research issues in this field. The existing deep learning frameworks usually model deep learning in the form of computational graphs,try to optimize computational graphs through subgraph substitution rules designed by experts and mainly use heuristic algorithms to search substitution sequences. Their shortcomings mainly include:1)the existing subgraph substitution rules result in a large search space and the heuristic algorithms are not efficient;2)these algorithms are not scalable for large computation graphs;3)cannot utilize the history optimization results. In order to solve the above problem,we propose GSO,a graph neural network-based deep learning computation graph optimization framework. We transfer the graph substitution optimization problem as the subgraph matching problem. Based on the feature information from the operators and the computation graph topology,we utilize the graph neural network to predict the subgraph matching feasibility and positions. We implement the framework using Python,which is compatible with the mainstream deep learning systems. The experimental results show that:1)compared to the total graph substitution rules,the proposed rule can reduce the search space by up to 92%;2)compared to the existing heuristic algorithms,GSO can complete the subgraph replacement process of the computational graph 2 times faster. The optimized computation graph is up to 34% faster the original graph.

Keywords　Optimizing DNN computation,Graph substitutions,Deep learning,Graph neural network

到稿日期:2021-07-19　返修日期:2021-08-16
基金项目:国家重点研发计划(2018YFB1004403);国家自然科学基金(61832001);北京大学—腾讯协同创新实验室项目
This work was supported by the National Key Research and Development Program of China(2018YFB1004403),National Natural Science Foundation of China(61832001) and PKU-Tencent Joint Research Lab.
通信作者:崔斌(bin.cui@pku.edu.cn)

1 引言

深度学习模型的执行效率是深度学习系统领域重要的研究问题之一。现有的深度学习系统框架通常将深度学习模型建模为计算图形式，图中的节点是基于硬件加速的预定义基础算子。随着深度神经网络模型规模越来越大，结构越来越复杂，深度学习模型的训练和推理代价也大幅增加[1]。为进一步提升深度学习模型的执行效率，对计算图进行优化很有必要。

主流的深度学习系统（如 TensorFlow[2]）采用类似贪心搜索的方式来逐个尝试专家设计的子图替换规则，试图减少冗余/低效的算子，提高深度学习模型的执行效率。这类方法每次进行替换时往往只以性能的严格提升为评价指标，无法保证全局最优解。MetaFlow[3]在此基础上提出了松弛的子图替换方法，允许搜索过程中一定程度上的性能下降，通过回溯寻找全局最优解。TASO[4]提出了一种在特定结构上自动探索子图替换规则的搜索框架，并拓展了对数据布局搜索的联合优化。OCGGS[5]对上述计算图中的子图替换问题进行了形式化建模和理论分析，证明了该问题是 NP-hard 和 Poly-APX-complete 问题，并提出了基于剪枝和动态规划的搜索算法来提高该问题的求解效率。

这些工作和研究大多采用启发式的算法来搜索子图替换序列，以优化计算图，在面对复杂模型时容易影响其有效性或效率。事实上，现有的子图替换优化方法仍存在以下挑战：1）基于启发式搜索的算法，搜索空间大，时间复杂度高，难以得到全局最优解；2）缺乏可拓展性，随着计算图规模增大，搜索效率急剧降低；3）难以利用历史的优化结果。

针对上述问题，本文提出了 GSO，即一个基于图神经网络的深度学习计算图子图替换优化框架。

GSO 将计算图子图优化建模成经典的子图匹配问题，把计算图中算子的属性作为特征信息，结合计算图本身的拓扑结构信息，对每种子图替换规则分别建立相应的图神经网络模型。训练好的图神经网络可用于识别和预测子图规则匹配的可行性和位置，高效过滤无效的子图替换规则，极大地缩小了搜索空间。为了验证此优化方案的有效性，本文基于与主流深度学习系统兼容的 Python 接口实现了该方案，实现框架如图 1 所示。

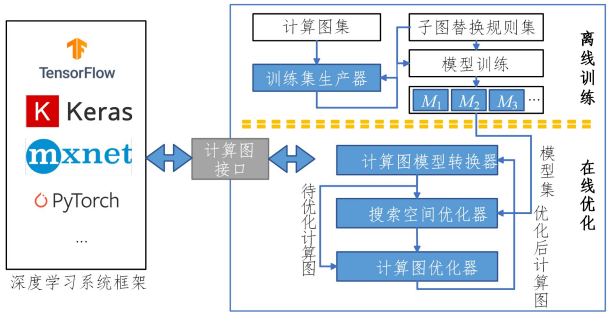


图 1 GSO 框架

Fig. 1 Overview for GSO

本文在几个常用的深度学习网络上进行了充分的实验，实验结果表明：1）基于图神经网络的子图匹配预测可以最多减少 92%的搜索空间；2）与基于启发式搜索方法相比，能够更快完成计算图优化过程，搜索优化效率达到不小于 2 倍的加速。

总的来说，本文的主要贡献如下：

（1）提出了一种基于图神经网络的深度学习计算图子图替换优化方法，提升了子图的替换效率。

（2）设计了一套针对计算图优化的训练数据集生成算法，通过在现有模型结构上进行改动来实现数据增强，用于指导机器学习模型的训练。

（3）实现了一个兼容主流深度学习系统的计算图优化框架，并在常用的模型上进行验证，实验结果表明，GSO 能够学习到计算图图结构特征和子图替换信息，对子图集进行筛选，进而更快更好地进行深度学习计算图优化。

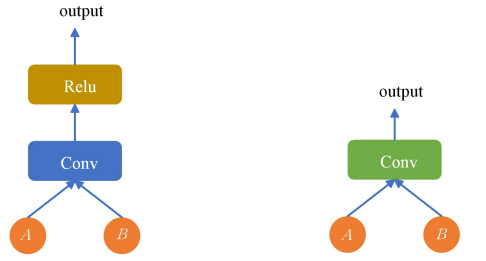
2 相关工作

2.1 子图替换优化

2.1.1 子图替换规则

子图替换规则一般包含一个源图和一个目标图，其中源图G_s定义了待替换子图的结构，其可以映射到计算图中的特定子图；目标图G_t定义了如何创建新的子图来替换映射的子图。源图和目标图一般应满足一定的约束，如类型约束、参数约束、输入输出约束等，以确保对于任意的输入均能得到相同的输出，即$\forall I: G_s(I)=G_t(I)$。

一个典型的子图替换规则如图 2 所示。



（a）源图 Relu(*Conv*(*A*,*B*))　（b）目标图(*Conv*(*A*,*B*))

图 2 子图替换示例

Fig. 2 Graph substitution example

现有的 DNN 框架通过专家手工设计的图替换规则来优化计算图，如 TensorFlow，TVM[6]，PyTorch[7]和 TensorRT[8]。TASO[4]在此基础上提出了一种自动生成图替换的方法，拓展了联合数据布局搜索的计算图优化算法，使用给定的操作符作为基本构建块生成候选子图替换规则，根据操作符规范进行验证，以确保生成的规则可行有效[4]。

2.1.2 子图替换优化搜索方法

现有的深度学习框架通常只考虑严格提高运行性能的变换，使用了基于贪婪规则的穷举优化策略，直接在输入计算图上执行所有适应的替换规则。MetaFlow 提出了通过放松严格的改进约束来探索复杂图的优化，基于回溯搜索算法来寻找最优计算图，该算法中允许保留在阈值范围内的替换增加了可探索的优化空间，这些“降级”的图替换作为转换图架构和最终发现具有更好运行时性能的新图的中间步骤非常有用。OCGGS 重新定义了子图替换问题，并基于 MetaFlow

提出了两种改进的搜索算法：1）基于剪枝（prune）的算法消除了对冗余图替换序列的检查，提高了子图替换搜索效率；2）基于剪枝的动态规划算法（DPP）重用了其所探索的图替换信息，使得探索无需在整个搜索空间上进行，进一步提高了效率。

2.2 图神经网络

图神经网络（GNN）将传统的深度神经网络从欧氏空间下的数据拓展到了图这种非欧氏空间下的数据。其中，图卷积神经网络（GCN）[9-12]是众多复杂图神经网络的基础，它把卷积操作从传统数据（如图像）拓展到图数据上。通过该操作，每个节点可以聚合自身特征和邻居节点的特征，并生成节点的新表示。基于学习到的图节点表示，可以用于图节点分类、链接预测、图分类等各种下游任务。

3 GSO：基于 GNN 的子图替换优化框架

3.1 问题定义

3.1.1 计算图优化问题

深度学习计算图优化可以看成一个在计算图上不断进行子图匹配的过程，为了更好地描述问题，我们将子图替换问题形式化定义为如下形式。

定义 1（计算图优化问题） 假设 $G=(V,E)$ 代表等待优化的计算图，C 代表代价模型，Φ 代表一系列预定义的子图替换规则，那么计算图优化在于找到一个子图替换序列 $P(\phi_1,\phi_2,\cdots,\phi_k)$，使得应用替换序列 P 后生成的新的计算图 $G'=G(P)$ 与原计算图 G 等价，并且总代价最小：

$$\min_{P(\phi_1,\phi_2,\cdots,\phi_k)} C_{G(P)} \quad \text{s.t. } G\equiv G(P) \tag{1}$$

其中，C_G 是对计算图 G 的执行代价估计，由框架设计的在线代价模型进行计算，$G(P)$ 表示在计算图 G 上应用子图替换序列 $P(\phi_1,\phi_2,\cdots,\phi_k)$ 后生成的新的计算图。

子图替换规则集 $\Phi=(\phi_1,\phi_2,\cdots,\phi_m)$ 表示 m 个子图替换规则。其中，$\phi_i=(G_s,G_t,f)$ 表示第 i 个替换规则元组，满足 $G_s \overset{f}{\equiv} G_t$。$G_s$，$G_t$ 分别代表等价的源子计算图和目标子计算图，f 代表源子计算图和目标子计算图之间的输入输出映射。

3.1.2 子图替换规则集优化问题

传统的计算图优化会按照回溯搜索等方法不断地在子图替换规则集中探索最好的子图替换序列 P，以使得计算图总代价最小（见定义 1）。在此方法下，所有可能的图替换序列的探索空间均会随着 Φ 的规模的增长而呈爆炸性增长。为了解决搜索空间巨大的问题，寄希望于借助机器学习过滤掉大部分不可行的子图替换规则（具体见 3.4 节），尽可能使搜索空间最小，为此我们将子图替换规则集优化问题定义为如下形式。

定义 2（子图替换规则集优化问题） 假设 $M(M_1,M_2,\cdots,M_m)$ 代表经过训练生成的模型集，Φ_t 代表在计算图 G 上使用的真实替换规则子集（$\Phi_t\subset\Phi$），Φ_p 代表使用模型集 M 预测的替换规则子集（$\Phi_p\subset\Phi$），那么子图替换规则集的优化目标在于训练生成一个模型集 M，使得应用 M 后，Φ_t 和 Φ_p 之间的差集最小。引用集合的 Jaccard similarity 相似性系数建模该问题：

$$\min_{M(M_1,M_2,\cdots,M_m)} J(\Phi_t,\Phi_p)=1-\frac{|\Phi_t\cap\Phi_p|}{|\Phi_t\cup\Phi_p|} \tag{2}$$

3.2 计算图优化框架

为了解决上述问题，本文实现了一个深度学习计算图优化框架。以下步骤展示了该框架的执行流程（其中，步骤 1、步骤 2 为框架的离线训练阶段，后续步骤为对给定计算图进行在线预测和优化阶段）：

（1）离线训练阶段

1）训练数据集生成，在现有深度学习模型的计算图上，基于本文提出的数据增强的方法，针对不同子图替换规则，获得带有标签的训练数据集；

2）将步骤 1）中的训练集和子图替换规则集进行迭代训练，生成 GCN 模型集。

（2）在线优化阶段

1）将现有深度学习框架中待优化的计算图转换为 GSO 中待优化的计算图；

2）使用模型集逐个推理得到在该计算图下优化后的可行子图替换规则集；

3）在过滤后的可行子图替换规则集上使用现有的搜索算法（如剪枝、动态规划等）对待优化计算图进行优化；

4）将优化后的计算图转换回框架计算图。

在训练模型的设计上，可以设计一个统一的模型对所有子图替换规则同时进行训练，学习到一个模型，用于预测优化计算图时可行的子图替换规则；也可以针对每一个子图替换规则分别进行训练，进而使每一个子图替换规则均学习到了一个独立的模型。为了对这两种模型进行验证和取舍，本文对 TASO 自动生成的子图替换规则集进行了分析，通过观察所有子图替换规则集中每种算子的数量占比发现，数量较多的 conv 算子在所有算子中的占比达到 14%，而数量最少的 split 算子在所有算子中的占比只有 1.3%。另外，每种算子的参数属性也不尽相同，如 stride 属性只在 conv 和 pool 算子中存在，relu 和 add 算子并没有任何参数属性。这种算子数量和算子参数属性的分布不均匀性导致子图替换规则之间千差万别，无法通过训练学习到一个统一的模型来适应所有子图替换规则。本文在实验过程中也发现，采用一个模型同时进行训练，试图得到一个统一的模型时，训练往往无法正常达到收敛。为此，本文采用针对每一个子图替换规则均进行训练得出一个独立的模型的方式进行后续的在线推理。

3.3 数据集生成

为了进行计算图的子图替换搜索空间优化的训练，需要一个足够的训练数据集 $X_G=(X_1,X_2,\cdots,X_n)$，其中 X_g（$1\leqslant g\leqslant n$）表示一个计算图。然而，现实中并没有类似公开的可用数据集，一种很自然的想法是将目前的深度学习模型集合起来组成数据集。Minar 等[13]对深度学习模型进行了综述，详细分类整理了常见的深度学习模型共 21 小类，个别小类又有一些变种类型，显然这个规模还无法达到机器学习训练的要求。其次，深度学习模型能够被优化的地方有限，将目前的深度学习模型集合起来的想法对于训练来说正样例不足，很容易造成泛化能力不强。

为了解决上述问题，作者利用常见的深度学习模型先

建立一个小的数据集作为初始数据，然后使用数据增强的方法进行增广，进而增加模型的泛化性。本文开发了两种算法进行数据增强。

(1)保存子图替换的中间结果的算法。使用剪枝、动态规划等搜索算法逐步优化一个原始计算图数据 X，将运行过程中生成的新图保存下来添加到 X_G 中。在原始图的基础上每进行一次子图替换的过程都可以看成一次数据增强的过程。当 X_G 达到所需数量要求时，停止该算法。

(2)随机添加抖动和扰乱的算法。保存子图替换的中间结果的算法的优化结果不足以反映未知计算图的复杂性，为此我们提出了一种基于扰动的增强方法。首先定义了扰动的几种操作：1)随机修改常见节点属性，如 conv. padding，conv. stride，pool. type 等；2)添加节点，在合适的位置添加常见节点，如 conv，relu，split 等；3)随机删除节点，我们随机选取一个 X_g，应用一种上述定义的扰动方法，将生成的新图加入 X_G 中，当 X_G 达到所需数量要求时，停止该算法。

针对每一个特定的子图替换规则 ϕ_j，将其建模为一个单独的二分类问题。通过预处理的方式将每个计算图的子图替换规则匹配情况都标注成 label。对于共有 k 个节点的训练计算图 X，其 label 表示为 $l(X,\phi_j)=(l_1,l_2,\cdots,l_k)$，$l_i\in(0,1)$。其中，$l_i=1$ 表示计算图 $X(V_1,V_2,\cdots,V_k)$ 的第 i 个节点 V_i 可以匹配规则 ϕ_j 进行替换。

3.4 模型集离线训练

在 GSO 中，为每种子图替换规则分别训练一个单独的 GCN 模型，该模型的输入不仅包括计算图的拓扑结构信息，生成的节点 label 信息还包括节点特征信息。将计算图节点类型、节点参数等信息作为该节点的特征，将节点之间的输入输出关系作为节点之间的边，将这些特征信息整合之后，使用 GCN 方法进行迭代训练，从而得到能够预测各个结点是否采用某子图替换规则的模型。本文使用了双层的 GCN 模型、一个 fc 层，为防止过拟合，在层之间使用了 Dropout 方法，输出 Dense 层使用的激活函数为 Sigmoid。GCN 模型结构如图 3 所示。

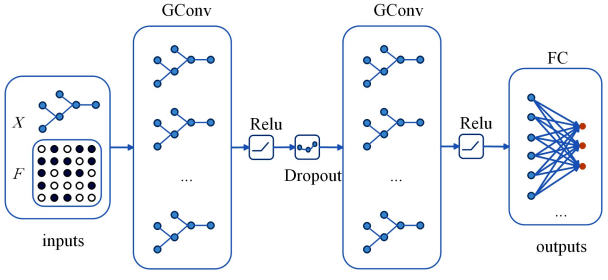


图 3 模型结构

Fig. 3 Model structure

图 3 中，X 表示输入训练集，F 表示 X 中节点边的连接关系，隐藏层的传递使用经典的 GCN 层传播方式。

$$H^{l+1}=\sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^lW^l)$$

通过迭代训练得到的子图替换规则模型，将被应用于 3.5 节中的计算图在线优化。

与一般的深度学习模型类似，生成的子图替换规则模型在子图替换规则集未变化的情况下，将无需在推理前重新进行训练，训练一次便可以重复使用。因此，在子图替换规则集未变化的情况下离线训练代价可以忽略不计。

3.5 计算图在线优化

首先调用 3.4 节生成的模型集 $M(M_1,M_2,\cdots,M_m)$，逐个对待优化计算图 G 进行推理，得出 G 中的各个节点对是否使用某个子图替换规则 M_j 的预测 $p_j(l_1,l_2,\cdots,l_k)$，$l_i\in(0,1)$。当 p_j 中存在为 1 的预测结果时，将 M_j 加入待用子图替换规则 M'，这样就从一个很大的搜索空间中预测并筛选出了最有可能使用到的子图替换规则集，然后利用常规的计算图优化算法进行优化。与 MetaFlow 类似，针对每个配置和数据布局测量 DNN 操作符的执行时间，并通过汇总测量的操作符执行时间来评估图的性能。

3.6 基于逻辑回归的优化方法

3.4 节介绍了一种使用 GNN 进行子图替换搜索空间的训练方法，为了进行对比实验，我们设计了一个利用逻辑回归(LR)[14]作为基线模型预测子图替换匹配的方法，生成了一个基于 LR 的模型集。该方法使用与 GNN 相同的节点类型及节点参数特征作为输入，网络结构为一个单一的 LR 层，但是没有利用到计算图的结构信息。

4 实验评测

4.1 实验配置

实验中采用的基本配置如下。

(1)深度学习模型集。本文实验使用了 4 种常用的网络结构：ResNet[15]，AlexNet[16]，VGG[17]，RNN[18]。为分析 GSO 和 LR 方法在搜索时间上的表现，又重点对部分网络的 blocks 进行了测试。

(2)数据集。本文中的训练数据集采用 3.3 节中的数据集生成方法生成，选择 ResNet，AlexNet，ResNet_blocks 组合以及部分常用替换组合等网络结构，通过数据增强最终生成 2000 个数据集，其中 1500 个用于训练，500 个用于测试。

在子图集上，使用自行设计的 20 个替换规则以及 TASO 自动生成的 133 个规则，共计 153 个规则。

(3)硬件配置。本文实验中采用的 CPU 为 Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz 共 20 核；GPU 为 Tesla P100-PCIE，CUDA Version：10.1。

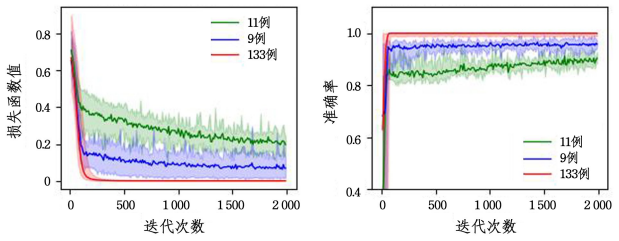
(4)衡量指标。本文首先在不同网络上直观地比较了两种子图搜索空间优化算法训练时的准确性表现。为了衡量优化之后计算图的性能，分别比较了 GSO 和 GSO-LR 方法优化后执行时间的变化。最后，本文还与 OCGGS 等相关方法就子图替换搜索时间进行了比较，展示了本文方法在子图替换上的优越性。

4.2 子图替换搜索优化训练表现

在 GSO 的配置上，使用两层 GCN 加上一层全连接层，节点特征为节点类别加上节点属性集合，共 30 个特征，隐藏层分别为 40，10 个特征，优化器使用 Adam。在 LR 的配置上，使用传统的一层全连接层，节点特征和优化器与 GSO 配置一样。分别训练 20 个 epoch，每个 epoch 将所有训练集均训练一遍，在每个 epoch 之后设置了早停机制。

针对每个子图替换规则训练生成一个 GCN 模型，共计

153 个模型。首先观察所有模型在训练过程中的表现，GSO 方法在训练过程中第一个 epoch 期间的 loss 和 accuracy 如图 4 所示。



(a)所有模型 loss 表现（$epoch=1$） (b)所有模型准确率表现（$epoch=1$）

图 4 GSO 方法的训练表现(电子版为彩色)

Fig. 4 Performance of training GSO

从图 4 中观察到：

(1)所有模型在训练过程中均能够正常收敛，在第 50 次迭代后准确率均能达到 80%以上，迭代 100 次后有 129(占 84%)个模型的准确率均达到 99%以上。我们在实验中还观察到，所有模型在第 5 个 epoch 内准确率均能达到 95%以上。

(2)在图 4(a)所示的 loss 表现图中，模型 loss 显示出了较明显的聚集情况，按一定的阈值将其分为 3 类，分别使用绿线、蓝线、红线表示(即类别 1、类别 2、类别 3)，条数分别为 11:9:133，子图替换规则分类如表 1 所列。

表 1 子图替换规则分类

Table 1 Subgraph substitution rules classfications

	条数	对应节点类型
类别 1	11	Conv，relu，matmul 等
类别 2	9	Conv，split，concat 等
类别 3	133	Matmul，add，transpose 等

可以发现，较为常见的替换规则(类别 1、类别 2)收敛较慢，而大多数替换规则(类别 3)能很快收敛，在 LR 训练过程中也能得到类似的结果。

4.3 子图筛选结果集的召回率表现

为了更准确地观察子图筛选模型的表现，首先分别验证 153 个模型在验证集上的表现情况。随机选取 4.1 节中验证集中的 200 个网络结构进行实验，将计算图中存在某个子图规则的源图作为真值，召回率表现如图 5 所示。可以看出，在 4 种网络结构上 GSO 方法都优于 GSO-LR 方法，这主要得益于 GNN 方法在训练模型时考虑了计算图结构边之间的关系，这会将其邻居节点的特征加入训练中，GNN 相比不考虑边关系的 LR 方法更为准确。

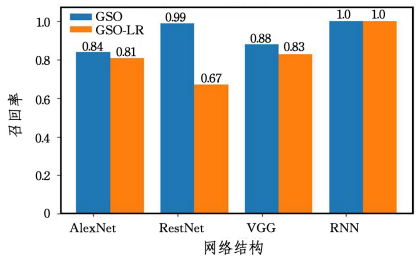


图 5 GSO 和 GSO-LR 的召回率表现

Fig. 5 Recall performance of GSO and GSO-LR

进一步分析使用 GSO 后搜索空间的变化发现，对于表现最好的 ResNet 网络来说，GSO 将原生的 153 个子图集缩减为 12 个(减少了 92%)。

4.4 优化前后执行时间的比较

在 GSO 框架计算图优化部分复用了穷举(ENU)、剪枝(prune)、动态规划(DPP)3 种方法，并分别比较了各种优化方法在使用 GSO 和 LR 前后生成的计算图执行时间。由于离线训练的 GCN 模型集可以被不同的输入计算图反复利用，因此在效率对比中，将其作为预处理，不考虑其时间代价。

由于 ENU，prune 以及 DPP 均为精确解，因此这里使用搜索时间较短的 DPP 方法生成的最优计算图执行时间作为最优优化图，并在此基础上分别加上 GNN 和 LR 方法进行比较。图 6 给出了 4 种网络结构分别在原计算图、DPP 方法生成的计算图以及加上 GNN 和 LR 之后生成的计算图的执行时间。其中，VGG 由于网络结构太大，无法在规定时间内执行完成，因此我们选取 7 个 blocks 进行测试，RNN 选取 5 个 steps。对于深层网络结构，部分研究工作采取图划分的方法分开执行计算图优化，能够有效缩短搜索时间。

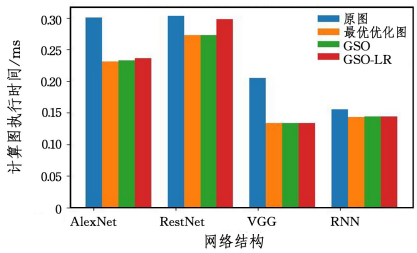


图 6 计算图优化前后执行时间的对比

Fig. 6 Runtime comparison of graph optimization

从图 6 中可以看出：

首先，使用加入了 GNN 和 LR 的方法来完成计算图的优化任务，这样能找出潜在的优化子图，实现计算图优化，证明了该方法的可行性。

然后，在 4 种网络结构中 GSO 方法生成的优化计算图执行时间与最优优化图相近，证明该方法能够将计算图所需的所有子图替换规则推理出来，而 GSO-LR 在 ResNet 网络结构上表现不佳，有部分需要的子图替换规则没有被推理出来，证明了 GNN 方法在计算图优化任务上优于 LR 方法。

最后，对于优化效果最好的 VGG(7 个 blocks)来说，使用 GSO 方法将原图执行时间由 0.205 ms 提速到 0.134 ms(提速达到了 34%)。

4.5 子图替换搜索时间比较

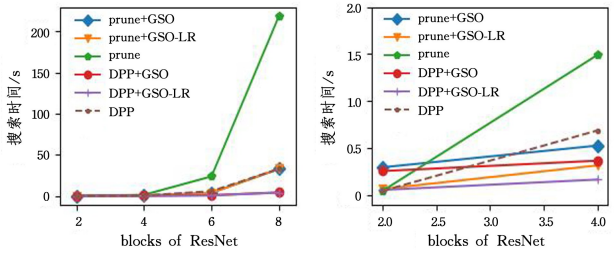
为了较好地比较子图替换搜索时间，我们选取 ResNet-blocks，分别在 2，4，6，8 个 blocks 上进行实验。图 7(a)给出了 ResNet 在 2～8 个 blocks 上的优化效率，图 7(b)更详细地给出了在 2 个 blocks 和 4 个 blocks 上的效率变化情况。

首先，对于节点数较多的深层网络结构，在回溯搜索的基础上，使用 GSO 或 GSO-LR 的方法对搜索空间进行优化，根据模型学习到的计算图结构特征，推理出可替换子图的位置范围，这样能够缩小子图替换规则搜索空间，有效降低时间复杂度，进而提升计算图的优化效率(搜索速度提速至少 2 倍)。

然后，随着 blocks 的增大，原生的 prune 和 DPP 方法的搜索时间显著增长，特别是 prune 方法的执行时间呈指数

增长，而使用 GSO 和 GSO-LR 方法后减小了搜索空间，搜索时间增长较为缓慢，其主要原因是优化了子图替换规则搜索空间后能够有效减少回溯搜索轮数，从而提高搜索效率。

最后，通过实验发现，在搜索时间上 GSO-LR 略估于 GSO，主要是由于 LR 的模型结构较为简单，推理时间会比 GNN 的模型短 0.2 s 左右，因此从搜索时间来看 GSO-LR 略优于 GSO。



(a)blocks 数量从 2 到 8 变化的结果　(b)blocks 数量从 2 到 4 变化的结果

图 7　ResNet-blocks 在不同 Blocks 下搜索时间的对比

Fig. 7　Search time costs comparison on ResNet-blocks, varying the block number

结束语　本文结合机器学习方法，致力于通过缩小搜索空间来提升深度学习计算图优化效率，因此提出了 GSO。本文实现了一套兼容主流深度学习系统的计算图优化框架。在该框架上的几个常用网络结构实验中，与现有研究人员提出的方法相比，GSO 能够获得更好的计算图优化性能。

下一步将继续探索如何将机器学习更好地应用到计算图优化领域，通过挖掘子图替换规则间的关系和模式来提升模型的泛化能力、召回率以及计算图的优化效率。

参 考 文 献

[1] CANZIANI A, PASZKE A, CULURCIELLO E. An analysis of deep neural network models for practical applications[J]. arXiv: 1605.07678v4, 2016.

[2] MARTÍN A, PAUL B, JIANMIN C, et al. TensorFlow: a system for large-scale machine learning[C]// 12th USENIX Symposium on Operating Systems Design and Implementation. 2016: 265-283.

[3] JIA Z H, JAMES T, TODD W, et al. Optimizing DNN Computation with Relaxed Graph Substitutions[C]// Proceedings of the 2nd SysML Conference. 2019.

[4] JIA Z H, ODED P, JAMES T, et al. TASO: optimizing deep learning computation with automatic generation of graph substitutions[C]// Proceedings of the 27th ACM Symposium on Operating Systems Principles. 2019: 47-62.

[5] FANG J Z, SHEN Y Y, WANG Y, et al. Optimizing DNN computation graph using graph substitutions[C]// Proceedings of the VLDB Endowment. 2020: 2734-2746.

[6] CHEN T Q, THIERRY M, JIANG Z H, et al. TVM: an automated end-to-end optimizing compiler for deep learning[C]// 13th USENIX Symposium on Operating Systems Design and Implementation. 2018: 578-594.

[7] PASZKE A, GROSS S, MASSA F, et al. Pytorch: An imperative style, high-performance deep learning library[C]// Advances in Neural Information Processing Systems. 2019, 32: 8026-8037.

[8] Nvidia tensorrt: Programmable inference accelerator[Z/OL]. https://developer.nvidia.com/tensorrt.

[9] THOMAS N, MAX W. Semi-supervised classification with graph convolutional networks[J]. arXiv: 1609.02907, 2016.

[10] MIAO X P, ZHANG W T, SHAO Y X, et al. Lasagne: A multi-layer graph convolutional network framework via node-aware deep architecture[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2021. https://ieeexplore.ieee.org/document/9513581.

[11] MIAO X P, GÜREL N M, ZHANG W T, et al. Degnn: Improving graph neural networks with graph decomposition[C]// Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021: 1223-1233.

[12] ZHANG W T, MIAO X P, SHAO Y X, et al. Reliable Data Distillation on Graph Convolutional Network [C]// Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020: 1399-1414.

[13] MINAR M R, NAHER J. Recent advances in deep learning: An overview[J]. arXiv: 1807.08169, 2018.

[14] MIAO X P, MA L X, YANG Z, et al. Cuwide: towards efficient flow-based training for sparse wide models on gpus[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2020. https://ieeexplore.ieee.org/document/9261124.

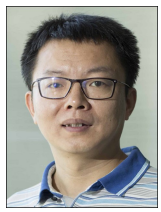
[15] HE K, ZHANG X Y, REN S Q, et al. Deep Residual Learning for Image Recognition [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

[16] ALEX K, ILYA S, GEOFFREY E. ImageNet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems. 2012, 25: 1097-1105.

[17] KAREN S, ANDREW Z. Very deep convolutional networks for large-scale image recognition[J]. arXiv: 1409.1556v4, 2014.

[18] ZAREMBA W, ILYA S, ORIOL V. Recurrent neural network regularization[J]. arXiv: 1409.2329, 2014.

MIAO Xu-peng, born in 1995, Ph.D candidate, is a student member of China Computer Federation. His main research interests include deep learning system and distributed optimization.



CUI Bin, born in 1975, Ph.D, professor, Ph.D supervisor, is a distinguished member of China Computer Federation. His main research interests include database system architectures, query and index techniques, big data management and mining.

（责任编辑：喻藜）