一种面向分布式深度学习系统的资源及批尺寸协同配置方法

梁 毅[1] 丁振兴[1] 赵 昱[1] 刘明洁[1),2)] 潘 勇[3] 金 翊[3]

[1)](北京工业大学信息学部 北京 100124)

[2)](北京机电工程研究所 北京 100074)

[3)](北京市计算中心 北京 100094)

摘 要 如何在受限时间内满足深度学习模型的训练精度需求并最小化资源成本是分布式深度学习系统面临的一大挑战.资源和批尺寸超参数配置是优化模型训练精度及资源成本的主要方法.既有工作分别从计算效率和训练精度的角度,对资源及批尺寸超参数进行独立配置.然而,两类配置对于模型训练精度及资源成本的影响具有复杂的依赖关系,既有独立配置方法难以同时达到满足模型训练精度需求及资源成本最小化的目标.针对上述问题,本文提出分布式深度学习系统资源-批尺寸协同优化配置方法.该方法首先依据资源配置和批尺寸超参数配置与模型训练时间和训练精度间的单调函数关系,选取保序回归理论工具,分别建立模型单轮完整训练时间和训练最终精度预测模型;然后协同使用上述模型,以资源成本最小化为目标,求解满足模型训练精度需求的资源和批尺寸优化配置解.本文基于典型分布式深度学习系统 TensorFlow 对所提出方法进行性能评测.实验结果表明,与既有基于自动化的资源或批尺寸独立配置方法相比,本文提出的协同配置方法最大节约资源成本 26.89%.

关键词 分布式深度学习系统;模型训练;批尺寸;资源配置;资源成本

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2022.00302

A Collaborative Method for Resource Allocation and Batch Sizing on Distributed Deep Learning System

LIANG Yi[1] DING Zhen-Xing[1] ZHAO Yu[1] LIU Ming-Jie[1),2)] PAN Yong[3] JIN Yi[3]

[1)](*Faculty of Information, Beijing University of Technology, Beijing* 100124)

[2)](*Institute of Beijing Electro-Mechanical Engineering, Beijing* 100074)

[3)](*Beijing Computing Center, Beijing* 100094)

Abstract Distributed deep learning systems are the engine of large-scale deep learning model training. As both of the volume of training datasets and the complexity of training models go up, the resource cost for the deep learning model training increases significantly, and hence, becomes a new concern of the distributed deep learning systems. In the distributed deep learning systems, the resource allocation refers to the number of computing nodes allocated to a parallel model training job and the batch sizing determines the training data size processed by a single training task. The empirical studies demonstrate that, from the perspective of the resource cost optimization, there is complex interdependence between the configurations of resource allocation and batch sizing. However, extant works ignore such interdependence and only take these two configuration

收稿日期:2021-01-19;在线发布日期:2021-09-26.本课题得到北京市自然科学基金面上项目(4192007)、国家重点研发计划(2017YFC0803300)资助.**梁 毅**,博士,副教授,主要研究方向为人工智能系统、云计算系统.E-mail:yliang@bjut.edu.cn.**丁振兴**,硕士研究生,主要研究方向为人工智能系统.**赵 昱**,硕士研究生,主要研究方向为人工智能系统.**刘明洁**,硕士,工程师,主要研究方向为云计算、高性能计算.**潘 勇**,硕士,高级工程师,主要研究方向为云计算系统.**金 翊**(通信作者),博士,工程师,主要研究方向为虚拟化技术、云计算系统.E-mail:jinyi@bcc.ac.cn.

methods as the independent ways to optimize the accuracy and computational efficiency of distributed deep learning model training respectively, and hence, are difficult to meet both goals of maximizing training accuracy with the training time constraint and minimizing resource cost. Aiming at this issue, a collaborative configuration method of resource allocation and batch sizing is proposed for distributed deep learning systems in this paper. Here, the resource cost is defined as the product of resource allocation and the training time. The proposed collaborative method is designed based on the observation that both function relationships of the resource allocation to the training time, and the batch sizing to the training accuracy, are monotonic. In the proposed method, the training accuracy prediction model and training time prediction model are first established with the isotonic regression technique. The training time prediction model is established as a function of the resource allocation and batch sizing and it can predict the elapsed time of one training epoch. The training accuracy prediction model is a function of the bath sizing and the total number of training epochs and it can predict the convergent accuracy of the model training. Then, with the given training time, the total amount of training epoch under the different configurations of resource allocation and batch sizing can be calculated with the training time model, and correspondingly, the convergent training accuracy can be predicted with the training accuracy prediction model. Finally, based on the above predictions, the optimized configurations of resource allocation and batch sizing, that satisfy the requirements of training accuracy, time constraint and resource cost, can be found by using Tabu search heuristics. We implement our proposed method in TensorFlow and verify its efficiency with representative deep learning models and training datasets. Experimental results demonstrate that the mean relative error of the proposed training time prediction model and training accuracy prediction model is 7.5% and 1.65% by maximum, respectively. Compared to the independent configuration methods, the proposed collaborative method can reduce the resource cost of deep learning model training by the maximum of 26.89%.

Keywords　distributed deep learning system; model training; batch sizing; resource allocation; resource cost

1　引　言

深度学习是计算机科学与人工智能的重要分支领域，它是神经网络的进一步延伸，通过从数据中自动学习有效的特征表示，提升预测模型的准确度，已被广泛应用于语音识别、图像识别、目标检测等领域[1]. 深度学习系统是支撑深度学习计算的基础设施. 随着深度学习数据规模的急剧扩增，单机的计算性能十分受限，分布式深度学习系统应运而生. 分布式深度学习系统通过将具有庞大计算量和数据量的深度学习任务部署于多个工作节点并行执行，提升深度学习的计算效率[2].

深度学习模型训练是指从海量样本数据中学习数据的内在关联和分布，是分布式深度学习系统支撑的核心计算[3]. 训练精度及计算效率是既有针对深度学习模型训练研究的关注重点. 然而，深度学习模型训练具有计算密度大，资源需求高的特征. 随着模型复杂度的增加，训练数据规模的急剧膨胀，用于模型训练的基础设施带来的资源成本开销日渐被重视. 因此，如何在受限时间内满足深度学习模型的训练精度需求并最小化资源成本是分布式深度学习系统面临的新挑战.

在单工作节点软硬件资源配置确定的前提下，分布式深度学习模型训练的资源配置是指单位时间内模型训练可使用的工作节点数量. 相应地，资源成本被定义为模型训练执行过程中配置的工作节点总量，可表示为资源配置与训练时间的乘积[4]. 资源配置对模型训练资源成本的影响体现于两方面：一是决定了模型训练单位时间内可使用的节点总量；二是决定了可并行执行的训练任务数量，进而影响训练时间. 然而，既有研究工作主要从计算效率的角度，通过扩大节点规模，加速模型训练执行，并未从资源成本的角度优化配置方案[5-7]. 分布式深度学习

中，批尺寸超参数是指为单个并行训练任务分配的训练样本数据规模.与其他深度学习超参数不同，批尺寸同时从模型训练的精度收敛效率和计算效率的角度影响训练时间并最终对资源成本产生影响.一方面，不同的批尺寸配置使得给定样本数据集划分为不同的训练样本子集，改变迭代训练中参数更新的趋势，进而影响训练的精度收敛速度；另一方面，在数据并行的模式下，对于一定规模的样本数据集，批尺寸配置决定训练的并行任务切割粒度，形成不同的任务启动和管理时间开销[8]，进而影响训练的计算效率.然而，既有研究主要从精度收敛效率的角度，对批尺寸的自动优化配置开展研究，提升训练可获得的训练精度，缺乏从资源成本的角度对批尺寸超参数开展优化配置[8-9].

总结而言，既有分布式深度学习系统资源和批尺寸配置均缺乏在保障训练精度的前提下面向最小化资源成本开展研究.另一方面，通过量化分析实验可知，从精度保障和资源成本双目标优化的角度，资源与批尺寸优化配置间存在互补性和复杂的依赖性.其中，互补性是指在给定资源配置的情况下，可通过改变批尺寸配置，优化精度收敛效率，缩短训练时间，进而实现最小化资源成本的目标；依赖性是指对于不同的资源配置，使资源成本最小化的批尺寸配置是不同的，因此无法通过资源和批尺寸各自独立的配置优化，实现资源成本最小化目标.

针对上述问题，本文提出一种面向分布式深度学习系统的资源-批尺寸协同优化配置方法.该方法面向分布式深度学习同步训练模式，首先以资源配置、批尺寸配置为参数，建立模型单轮训练时间和模型训练精度的预测模型；然后协同利用上述预测模型选取满足模型训练精度需求的资源和批尺寸优化配置解，实现资源成本最小化目标.具体而言，本文的主要工作如下：

(1)以典型分布式深度学习系统 TensorFlow 为例，通过量化实验分析，揭示了以最小化资源成本为目标，资源与批尺寸的优化配置间存在互补性和依赖性[10].

(2)提出了一种面向分布式深度学习同步训练模式的资源及批尺寸协同配置方法.该方法根据批尺寸配置与单轮完整训练时间及模型精度之间具有分段单调数学关系的特性，选取保序回归理论工具，以资源、批尺寸配置为参数，构建单轮完整训练时间预测模型；以批尺寸配置和完整训练轮次为参数，构建训练精度预测模型.然后，采用禁忌搜索算法，在上述两种预测模型的基础上启发式地搜索，求解以资源成本最小化为目标，保障训练精度需求的资源和批尺寸优化配置解.

(3)采用典型深度学习模型及数据集，对本文提出的资源-批尺寸协同配置方法进行了性能评测.实验结果表明，本文所提出的资源-批尺寸协同配置方法可较好地预测训练时间和训练精度；与基于人工经验的配置方法比较，可最大节约资源成本 34.83%；与既有基于自动化的配置方法比较，可最大节约资源成本 26.89%.

2 背景与问题提出

2.1 背景知识

分布式深度学习中，模型训练按照并行模式可分为数据并行和模型并行[11].本文主要针对数据并行模式.数据并行是指将海量训练样本数据划分为多个小批次数据(batch)，将每个批次数据映射至一个并行任务执行.所有并行任务每完成一个批次数据处理后进行同步并更新参数，称为一次迭代(Iteration).经过多轮批次数据并行处理迭代，当所有训练样本数据处理结束，称为完成了一次完整训练(Epoch).经过多轮完整训练后，模型被应用于测试集，评估质量.如此反复，直至达到训练轮次阈值或达到训练精度阈值[12].

针对数据并行模式的分布式深度学习模型训练，通常采用基于参数服务器的系统架构模型[2].如图 1 所示，在该模型中，服务器被分为参数服务器和工作节点.其中，参数服务器主要负责参数的聚合更新以及训练任务调度.实际执行模型训练的并行任务则部署于工作节点运行.因此，本文中分布式深度学习模型训练的资源配置主要关注模型训练的工作节点规模.批尺寸是分布式深度学习模型训练最重要的超参数之一.分布式环境下，批尺寸是指单个训练任务的样本数据输入规模.如前所述，与其他深度

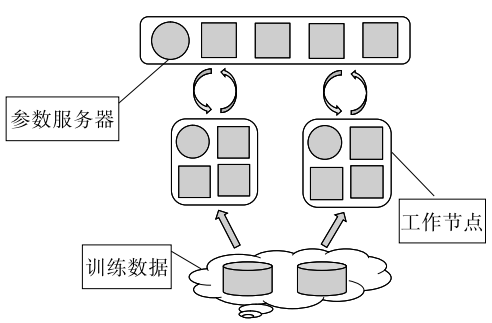


图 1 分布式深度学习系统-参数服务器架构

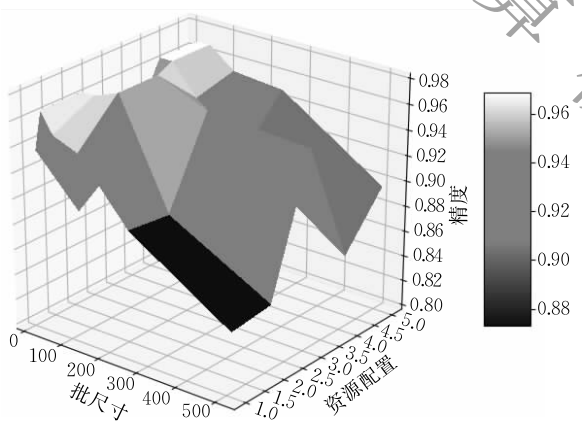
学习超参数不同，批尺寸超参数不仅从精度收敛效率的角度，还从计算效率的角度与资源配置协同对训练的资源成本产生影响. 因此，本文以其他超参数配置确定为前提，重点探讨资源配置与批尺寸配置的协同优化.

2.2　问题提出

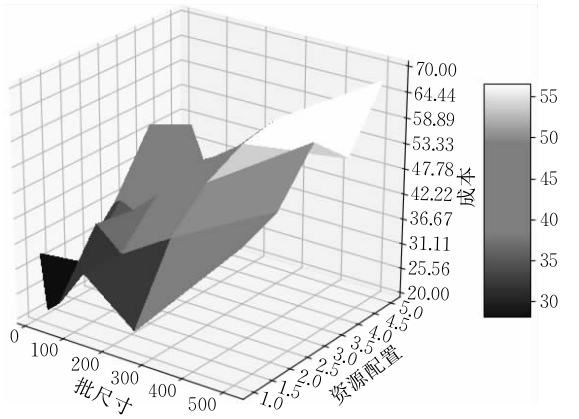
本节基于典型分布式深度学习系统 TensorFlow 说明资源与批尺寸协同配置的必要性和可行性. 实验选取 LeNet-5 卷积神经网络模型，并使用 CIFAR-10 开源数据集进行训练.

通过分析图 2，可以得到以下结论：

结论 1. 在相同的资源配置和时间要求条件下，可以通过调优批尺寸，加速模型训练精度收敛，减小资源成本. 本文将这种特性称为互补性. 以图 2(a)为例，当训练精度要求为 0.975，资源配置为 2 个工作节点时，若将批尺寸设置为 16，则 6000 s 内可达到的训练精度仅为 0.935，无法满足要求. 然而，不改变资源配置，将批尺寸调整至 64，则可将精度提升至 0.976. 同样，当资源配置为 3 个工作节点时，将批尺寸配置从 256 调整至 64，也可在 6000 s 内，将训练精度从 0.950 提升至 0.975，满足精度需求. 在上述例子中，不改变资源配置规模，然而，通过调优批尺寸则可使训练精度在限定时间内满足要求，降低了资源成本. 这是因为，在同一资源配置下，批尺寸配置的调整影响参数的更新效率，进而加速模型训练精度收敛，从而在限定时间内达到要求的训练精度.



(a) 训练精度



(b) 资源成本

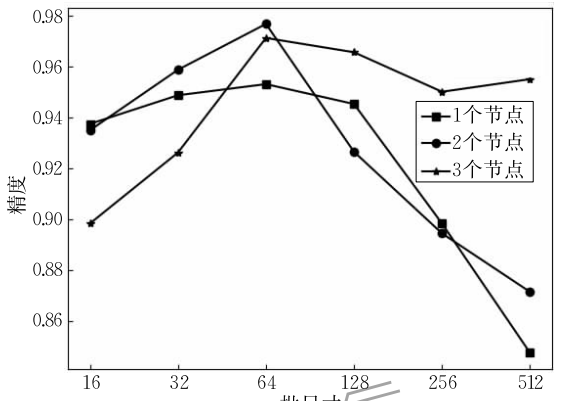
图 2　不同批尺寸及资源配置组合下的模型训练精度和资源成本

结论 2. 在保证训练精度前提下从最小化资源成本的角度上，资源及批尺寸优化配置相互关联. 本文将这种特性称为依赖性，即在不同的资源配置下，达到资源成本最小化的批尺寸优化配置不同. 以图 2(b)为例，当资源配置为 1 个工作节点，批尺寸设置为 32 以及资源配置为 3 个工作节点，批尺寸配置为 64 时，均达到了同一资源配置下的最小化资源成本. 对于不同资源配置，最小化资源成本的批尺寸优化配置不同的主要原因在于，模型训练可达到的精度与完整训练轮次息息相关[8]. 不同资源配置下相同的批尺寸配置使得训练并行任务的总量相同. 较大的资源配置虽然可以缩短单轮完整训练的时间，但由于不同资源配置下，并行任务的并发启动开销不同，单轮完整训练时间并未随着资源规模的增加而线性下降. 如图 3(b)中，批尺寸为 32 时，当资源配置从 1 个节点增加至 3 个节点时，单轮完整时间缩短了约 1/2 而非 1/3. 由此可知，相同资源成本、不同资源配置下，训练可完成的完整训练轮次不同. 这就导致将针对资源配置方案 A 的批尺寸优化配置值应用于资源配置方案 B，可获得的模型训练精度发生变化，无法保证其满足精度要求. 相应地，无法保证该批尺寸配置仍然是面向资源成本最小化的优化解.

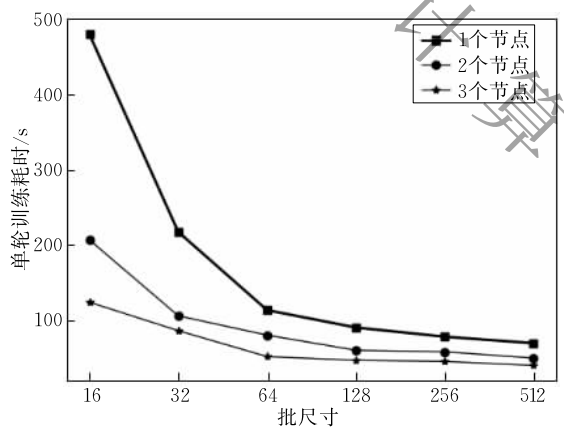
为进一步考察批尺寸配置与单轮完整训练时间以及模型精度之间的关系，本节通过实验进行说明.

结论 3. 批尺寸配置与单轮完整训练时间及模型精度之间的数学关系具有分段单调性质. 分析图 3(a)可知，随着批尺寸的增加，训练精度呈现先增加后减少的趋势. 这是因为，依据深度学习模型训练原理，采用梯度下降等方法进行模型参数迭代更新过程中，批尺寸配置与单位训练任务处理的样本数据规模成正比，数据规模过小导致参数更新趋势具有较强的不确定性，规模过大则导致参数更新易陷入局部最优，上述情况均导致了模型训练精度难以收敛[8,13]. 另一方面，分布式深度学习模型训练采用并行计算模式，批尺寸配置决定了每个并行训练

任务的数据处理规模，进而决定了一次完整训练包含的并行任务总量.在给定资源配置的前提下，过多的并行任务导致额外的任务启动开销，过少的并行任务则未能充分利用所分配的计算资源.上述两种情况均降低了训练的计算效率.



(a) 训练精度



(b) 单轮完整训练时间

图 3 不同的批尺寸设置下模型训练精度和单轮完整训练时间的变化

3 资源-批尺寸协同配置方法

以满足模型训练精度要求且最小化资源成本为目标，本文提出资源-批尺寸协同优化配置方法.该方法的基本思想是首先构建关于训练精度和单轮完整训练时间的性能预测模型，然后结合上述预测模型，采用启发式算法搜索满足时间要求的资源和批尺寸优化配置组合.

3.1 性能预测模型的定义

从训练时间的角度，结合 2.3 节结论 3，资源配置(r_{num})及批尺寸配置(b_{size})均直接影响单轮完整训练时间.从训练精度的角度，本文着重关注批尺寸与训练精度之间的量化关系.同时，相同批尺寸配置下，不同的完整训练次数(e_{num})获得不同的精度.因此，本文所构建的单轮完整训练时间预测模型和精度预测模型分别形如式(1)和式(2).

$$f_1(b_{\text{size}}, r_{\text{num}}) = t_{\text{predict}} \tag{1}$$

$$f_2(b_{\text{size}}, e_{\text{num}}) = acc_{\text{predict}} \tag{2}$$

值得注意的是，式(1)中的训练时间是指单轮完整训练时间.在时间要求内，通过确定单轮完整训练时间，可以确定训练可执行的完整训练次数，从而可利用精度预测模型计算最终可达到的精度.

从资源配置的角度，增加资源配置可使模型训练获得更多计算资源，缩短训练时间.同时，依据 2.2 节分析得到的结论可知，批尺寸配置与单轮完整训练时间及模型精度之间的数学关系具有分段单调性质.由此，启发本文使用保序回归进行模型函数拟合.

保序回归是回归分析的一种，用于寻找一组非递减的片段连续线性函数，使其与样本尽可能地接近[14].在本文提出的方法中使用保序回归的优势在于，一是充分利用了模型各参数与目标变量间的单调关系；二是由于不同的训练模型及数据集其性能预测模型的具体形态各异，采用保序回归可在缺乏预测模型函数先验知识的前提下准确建模.

3.2 性能预测模型的构建与求解

本文所需构建的两个性能预测模型均属于多元函数.然而，既有多维保序回归存在计算复杂度高，缺乏完备实现的问题[15].因此本文采用如下设计思路：首先分析不同参数与目标变量间的相关性，以此为权重，将多维保序回归问题转换为多个一维保序回归问题进行求解，并将一维保序回归结果依据权重进行叠加作为模型的预测结果.

对于任一训练模型及数据集，首先进行性能建模样本数据收集，构建样本数据集 $SampleSet = \{ss_1, ss_2, \cdots, ss_n\}$.任一样本数据 ss_i 可表示为一个五元组，形式如下：

$$ss_i = (b_{\text{size}}, r_{\text{num}}, e_{\text{num}}, t_{\text{act}}, acc_{\text{act}}) \tag{3}$$

其中，b_{size}，r_{num}，e_{num} 分别表示样本数据收集时，实际配置的批尺寸、资源数量和完整训练次数，t_{act} 和 acc_{act} 则表示实测获取的单轮完整训练时间和获得的训练精度.在两个性能模型构建时，则根据模型参数选取样本数据中对应项使用.

在样本数据收集的基础上，本文利用信息熵、条件熵和互信息原理，根据模型各参数的信息量多少来分配权值.对于任意一个随机变量 X，它的信息熵定义如下：

$$H(X)=-\sum_{x}P(x)\log_2 P(x) \tag{4}$$

变量的不确定性越大，熵也就越大，解释其所需要的信息量也就越大. 条件熵表示在已知随机变量 Y 的条件下，随机变量 X 的信息量，可以表示为

$$H(X|Y)=-\sum_{x,y}P(x,y)\log_2 P(x|y) \tag{5}$$

互信息可以由上述信息熵和条件熵得到，表示一个随机变量中包含的关于另一个随机变量的信息量，X 和 Y 的互信息定义如下：

$$MI(X;Y)=H(X)-H(X|Y) \tag{6}$$

本文提出一种基于互信息的多参数权重分配方法，具体流程如算法 1 所示.

算法 1.　基于互信息的多参数权重分配方法.

输入：样本集合 *SampleSet*，参数集合 *ss*

输出：权重集合 *W*

1. *M*←NULL，*W*←NULL //初始化因素的互信息集合、权重集合
2. FOREACH *ss* in *SampleSet* DO
3. *normalize*(*ss*) //对每个参数的样本集合进行归一化处理
4. END FOR
5. *H*=*getEntropy*(*SampleSet*) //计算样本信息熵
6. FOREACH *s* in *ss* DO
7. *H_p*=*getConditionalEntropy*(*SampleSet*,*s*) //计算针对当前参数，样本的条件熵
8. *m*=*H*−*H_p* //计算样本去除当前参数与否的互信息
9. *m*=1−1/(*m*+1) //互信息倒数计算参数信息量
10. *M.append*(*m*)
11. END FOR
12. FOREACH *m* in *M*
13. *w*=*m*/*sum*(*M*) //计算每个参数的权重
14. *w.append*(*W*)
15. END FOR
16. RETURN *W*

首先初始化中间过程保留的参数信息量集合以及输出的权重集合(第 1 行)，为消除特征指标单位的差别和特征指标量级不同的影响，对每个参数的样本作归一化处理(第 2～4 行)；数据预处理之后计算每个参数的信息量(第 5～9 行)，该信息量计算是基于互信息度量. 以单轮完整训练时间 t_{predict} 预测为例，首先计算样本中需预测变量所对应样本数据，即 t_{act} 的信息熵(第 5 行)，然后依次计算针对参数 b_{size} 和 r_{num} 的条件熵(第 7 行)，再次根据样本信息熵和条件熵，计算参数与预测变量的互信息度量(第 8 行). 从原理上看，互信息度量值越大的参数，应赋予更大的权重. 本文将每个参数的信息相关度量约束在[0,1]范围内，以便比较(第 9 行)；最终根据各参数信息量确定其权重(第 10～13 行)，参数 p 的权重可表示为

$$w_p=M_p\Big/\sum_{l=1}^{k}M_l \tag{7}$$

其中，M 是信息量，k 是参数数量；最后返回权重集合 W，算法结束.

在确定权重的基础上，本文选取一元保序回归算法对预测模型中各参数与目标变量间数学关系进行求解. 依据观测实验，样本数据中各参数的取值可构成简单半序关系，本文选取 PAVA 算法进行模型求解[16]. 为进行模型求解，首先应进行样本数据的抽取，对于每一个一元保序回归求解，其样本数据集可表示如下：

$$SampleSubset=\{ssb_1,ssb_2,\cdots,ssb_m\} \tag{8}$$

其中，任一 ssb_i 可表示为如下二元组：

$$ssb_i=(pa_i,tar_i) \tag{9}$$

pa_i 和 tar_i 分别表示在模型参数和目标变量上的取值. 同时，pa_i 满足如下半序关系：

$$pa_1\prec pa_2\prec\cdots\prec pa_i \tag{10}$$

PAVA 算法的原理是在原数据序列的基础上，寻找一个单调序列，使得该单调序列满足与原始数据序列的具有最小平方差. PAVA 算法对样本数据中出现非单调的情况，通过回溯前继样本数据求取平均来消除非单调性. 本节提出的保序回归算法中，基于互信息的多参数权重分配方法的时间复杂度为 $O(N)$，PAVA 算法求解一元保序回归方法的时间复杂度为 $O(N)$，且当完成权重分配后，对于每一维度保序回归问题，可使用 PAVA 算法并发执行，最终的计算复杂度为 $O(kN)$，$k<N$. 因此，随着数据规模的增大，本文提出的预测方法具有较好的计算效率.

3.3　优化配置解的搜索

在性能预测模型构建的基础上，根据前述分析，优化的资源-批尺寸协同配置求解可抽象为带约束的目标优化问题，形式如式(11).

$$\begin{aligned}&\min(r_{\text{num}}\times t_{\text{predict}}\times e_{\text{num}})\\&\max(acc_{\text{predict}})\\&\text{subject to:}\begin{cases}acc_{\text{predict}}\geq ACC\\\sum\limits_{j=1}^{e_{\text{num}}}t_{\text{predict}}\leq T_{\text{limit}}\end{cases}\end{aligned} \tag{11}$$

其中，$r_{num} \times t_{predict} \times e_{num}$ 表示模型训练的资源成本，ACC 表示应达到的精度要求，T_{limit} 表示要求的训练时间，e_{num} 表示单轮完整训练时间.

本文使用禁忌搜索方法选取出优化协同配置解[17]. 禁忌搜索方法是一种亚启发式随机搜索方法，可通过维护禁忌表对当前的最优配置解进行记录，并指导下次搜索，防止陷入局部最优.

算法 2 给出资源-批尺寸协同配置优化解选取算法. 该算法依据各参数的取值范围，确定组合配置候选集 D 作为算法输入. 在搜索算法中，首先需要初始化禁忌表为空(行 1)；并从候选集中选取有效的资源和批尺寸配置(行 2). 所谓有效配置满足利用所构建的性能预测模型计算，得出在该配置下，训练时间小于要求时长 T_{limit}，且可达到精度满足精度要求 ACC. 然后，算法进入迭代，每次迭代计算当前选择的配置组合的收益. 在该算法中，定义配置的收益为资源成本的反比(行 9). 为了计算可获得精度和资源成本，该算法利用所构建的单轮完整训练时间预测模型，计算当前配置下单轮完整训练时间，然后根据该时间信息，计算当前配置下可执行的最大训练次数，并利用所构建的训练精度模型，从该最大次数开始逐步递减，选取可满足精度要求的最小完整训练次数；将该次数对应的训练精度和资源成本分别作为当前资源和批尺寸配置组合可获得的精度和资源成本(行 6～9). 本算法基于禁忌搜索，因此对于每一个有效配置组合，需查找禁忌表，若该配置组合已在禁忌表中，则分别按照步长为 1 增减批尺寸和资源配置数量，从而计算当前配置组合的邻域配置(行 10～12). 若该配置组合不在禁忌表中，则选取收益最大者替换当前配置，若出现收益相同的配置，则选取精度最大者替换(行 13～21). 该算法迭代的终止条件是满足足够的迭代次数，最终返回所有搜索配置解中收益最大者(行 22～24).

算法 2. 参数-资源协同配置优化方案选取算法.

输入：候选集 D，精度要求 ACC，要求训练时间 T_{limit}，固定数据规模 d_{size}，最大迭代次数 NG

输出：优化方案 sol_{opt}

1. INITIALIZE T=NULL //初始 sol_{opt} 化禁忌表为空
2. $g=getRandomEff(D)$ //随机选取有效的初始解
3. $sol_{opt} \leftarrow$NULL, $sol_c \leftarrow$NULL
4. $getSolInfo(sol_c, g)$
5. REPEAT
6. $t_{predict} \leftarrow calRuntime(sol_c.b_{size}, sol_c.r_{num})$ //计算当前配置下单轮完整时间
7. $sol_c.e_{num} \leftarrow getOptIter(sol_c, T_{limit}, ACC)$ //计算当前配置下可执行的最大训练次数
8. $sol_c.acc \leftarrow getACC(sol_c.e_{num})$ //计算当前迭代轮次下可达到的精度
9. $sol_c.rew \leftarrow 1/(sol_c.r_{num} \times t_{predict} \times sol_c.e_{num})$ //计算当前配置的收益
10. IF sol_c in T DO //若当前配置解在禁忌表中
11. $sol_c \leftarrow move(sol_c, step)$ //以步长为 $step$ 移动，产生新的批尺寸和资源配置
12. END IF
13. ELSE DO //若当前配置解不在禁忌表中
14. IF $sol_c.rew > sol_{opt}.rew$ DO
15. $sol_{opt} \leftarrow sol_c$ //选取邻域内收益最大的配置解替换当前配置
16. END IF
17. IF $sol_c.rew == sol_{opt}.rew$ DO
18. $sol_{opt} \leftarrow$MAX$(sol_c.acc, sol_{opt}.acc)$
19. END IF
20. $addTabu(sol_c, T)$ //将当前的最优解加入到禁忌表中
21. END ELSE
22. $update(T)$ //更新禁忌表
23. until iteration reaches NG//迭代直到满足最大迭代轮次
24. RETURN sol_{opt}

4 性能评估

4.1 实验方法概述

为了说明该方法具有较好的性能表现，本文选取四个有代表性的优化方法作为对比对象：

(1) 基于人工经验的配置方法. 对于每组实验，选取具有不同调参经验背景的调试人员尝试 60 组以上配置组合，选取最优解作为配置解.

(2) LB-BSP 方法. 从计算效率的角度上，针对云数据中心单工作节点可用资源动态变化的场景，通过对模型训练时间量化建模，进行分布式深度学习批尺寸的动态调优[9]. 本文基于 GPU 集群与该方法进行性能比较.

(3) 贝叶斯优化配置方法. 以训练精度为优化目标，采用贝叶斯优化方法通过在定义的批尺寸解空间中自动化搜索，求解出最优的批尺寸配置[18].

(4) Cynthia 方法. 以资源成本最小化为目标，单纯进行资源配置的优化. 为具备可比性，在本文实验中，将 Cynthia 中的训练损失预测建模替换为本文的训练精度预测建模，重点比较单纯资源配置与

资源-批尺寸协同配置对资源成本的优化能力[19].

综合以上对比对象的实验评测系统，本文最终选取了共性系统：TensorFlow 系统进行实现和评测. 在实验的数据集和模型选取上，本文选取了 CIFAR-10 数据集训练 CIFAR-10 DNN 模型和 VGG-19 模型；MNIST 数据集训练 LeNet-5 模型；SVHN 数据集训练 ResNet-18 模型[19].

实验环境由 11 台服务器组成，选取 1 台服务器作为参数服务器，其余 10 台服务器作为工作节点. 每台服务器的配置如表 1. 实验过程中，其他超参数依据 TensorFlow 官网示例程序推荐配置，主要设置包括学习率为 0.01，学习率衰减率为 0.9，优化器为基础的梯度下降优化器，损失函数为交叉熵函数.

表 1　测试环境配置

资源名称	资源配置
CPU	Intel　Xeon　SkyLake 6151 v5 3.00GHz
GPU	1×NVIDIA T4/1×16GB
内存	32GB
磁盘	1TB
网络带宽	PS：5Gbit/s Worker：2Gbit/s

4.2　性能模型准确度评测

本节首先评估本文提出的两个性能预测模型的准确度. 实验对于每一个数据集和模型，分别设置资源配置值为 1 到 8 个工作节点、批尺寸配置以均匀分布的模式在[16，256]区间内随机选取 10 个值[20]，完整训练次数以均匀分布的模式在[1，40]区间内随机选取 3 个值，将上述配置值进行组合形成 240 组配置组合. 实验对于每一个评测对象，分别收集 240 组配置下的训练时间和最终训练精度，与使用模型获得的预测值进行比较，求取平均相对误差（*MRE*），计算如式(12)所示.

$$MRE=\frac{\sum_{i=1}^{n}|y_i-y_i'|}{\sum_{i=1}^{n}y_i'} \tag{12}$$

其中，y_i'为第 i 组实验中的真实值，n 为实验组数，y_i 为预测模型输出的预测值.

图 4 给出四个评测对象的测试结果. 由图可知，四个评测对象的训练精度平均相对误差均小于 1.65%，训练时间的平均相对误差均小于 7.5%，具有较好的预测准确率，这是由于在保序回归中引入了数据间单调变化的约束关系，能够更准确发现样本数据的内在规律. 其中，MNIST＋LeNet-5 的训练精度平均相对误差最大，训练时间平均相对误差最小. 这是因为 MNIST＋LeNet-5 在较小的批尺寸增加范围内精度即到达峰值，而后处于较为平缓的趋势. 样本数据中均匀分布采样，未能较好的匹配该变化模式，形成一定的噪音数据. 另一方面，从训练时间的角度，MNIST 数据规模较小，可选取的批尺寸和资源配置区间较小，相对而言样本数据则较为充足，具有更好的预测准确度. 其余评测对象的训练精度误差在 1.41%～1.5%，训练时间误差在 5.7%～6.6%.

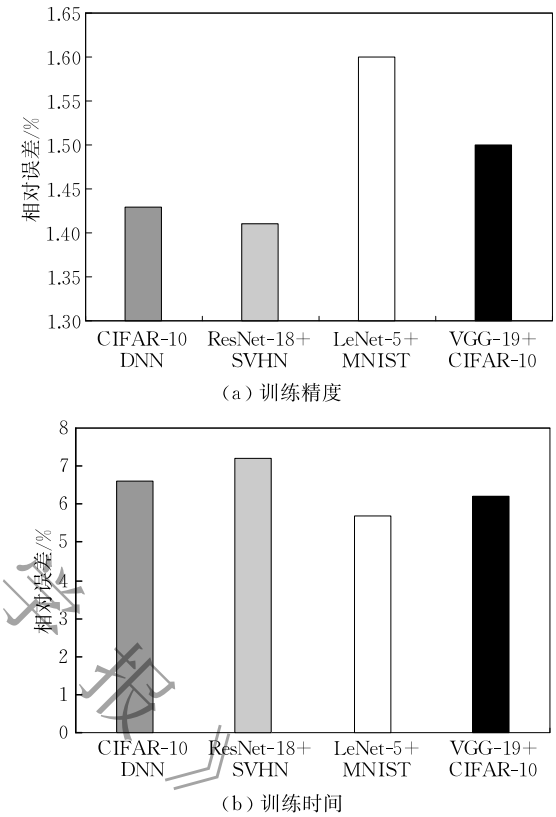


图 4　性能预测模型准确度

4.3　限定训练时间条件下的性能评测

本节实验在限定训练时间的条件下，比较本文所提出的协同配置方法与 LB-BSP 方法及贝叶斯优化配置方法可获得的训练精度①. 实验根据每个模型训练的总时长，按比例选取不同的时间限制，测算了在时间要求内，训练模型可以达到的精度. 由于 LB-BSP 方法和基于贝叶斯优化配置方法仅以确定的资源配置规模为前提进行批尺寸配置优化. 因此，本节实验参照协同配置方法在不同时间条件下形成的优化资源配置，采用 LB-BSP 方法和贝叶斯优化配置方法求算相应的批尺寸优化配置值，并对三种

① 由于 Cynthia 方法中需要依靠精度模型进行资源配置的优化选取，而在限定时间条件下，没有精度的约束，无法进行资源配置的优化，因此本节实验不考虑 Cynthia 作为对标对象.

方法最终可达到的训练精度进行比较. 表 2、表 3、表 4 和表 5 分别给出三个评测对象对于不同训练模型及数据集的批尺寸和资源优化配置.

表 2 CIFAR-10 DNN 参数设置

时间限定/s	LB-BSP		贝叶斯优化		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
2000	512	10	256	10	128	10
3000	512	7	256	7	64	7
4000	512	6	128	6	128	6
6000	512	4	64	4	64	4

表 3 ResNet-18+SVHN 参数设置

时间限定/s	LB-BSP		贝叶斯优化		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
1000	512	8	128	8	256	8
2000	512	6	128	6	256	6
3000	512	5	64	5	256	5
4000	512	4	32	4	64	4

表 4 LeNet-5+MNIST 参数设置

时间限定/s	LB-BSP		贝叶斯优化		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
50	256	4	64	4	128	4
70	256	3	64	3	64	3
100	256	2	64	2	32	2
200	256	1	64	1	16	1

表 5 VGG-19+CIFAR-10 参数设置

时间限定/s	LB-BSP		贝叶斯优化		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
13000	256	3	64	3	32	3
15000	256	4	64	4	32	4
18000	256	3	256	3	16	3
20000	256	7	128	7	64	7

图 5 给出三个评测对象在限定时间条件下的训练精度比较. 由图可知,与 LB-BSP 比较,协同配置精度提升最大为 2.81%;与贝叶斯方法相比,精度提升最大为 1.4%. 这是由于 LB-BSP 针对模型训练过程中根据工作节点可用资源变化进行批尺寸的实时调优. 本文针对的场景是模型训练前的静态资源、批尺寸配置场景,因此,LB-BSP 方法的"动态"特征未发挥优势,在不同的资源配置下批尺寸的优化值相同. 贝叶斯优化仅针对于批尺寸,由于资源配置不同,每次参数更新时梯度规模不同,因此批尺寸略有差异. 既有贝叶斯优化方法在训练时间不受限的前提下,以优化训练最终收敛精度为目标进行批尺寸调优. 相较于本文提出的协同配置方法,该方法忽略了批尺寸调整对训练计算效率的影响,因此未能在限定时间内达到较高的训练精度.

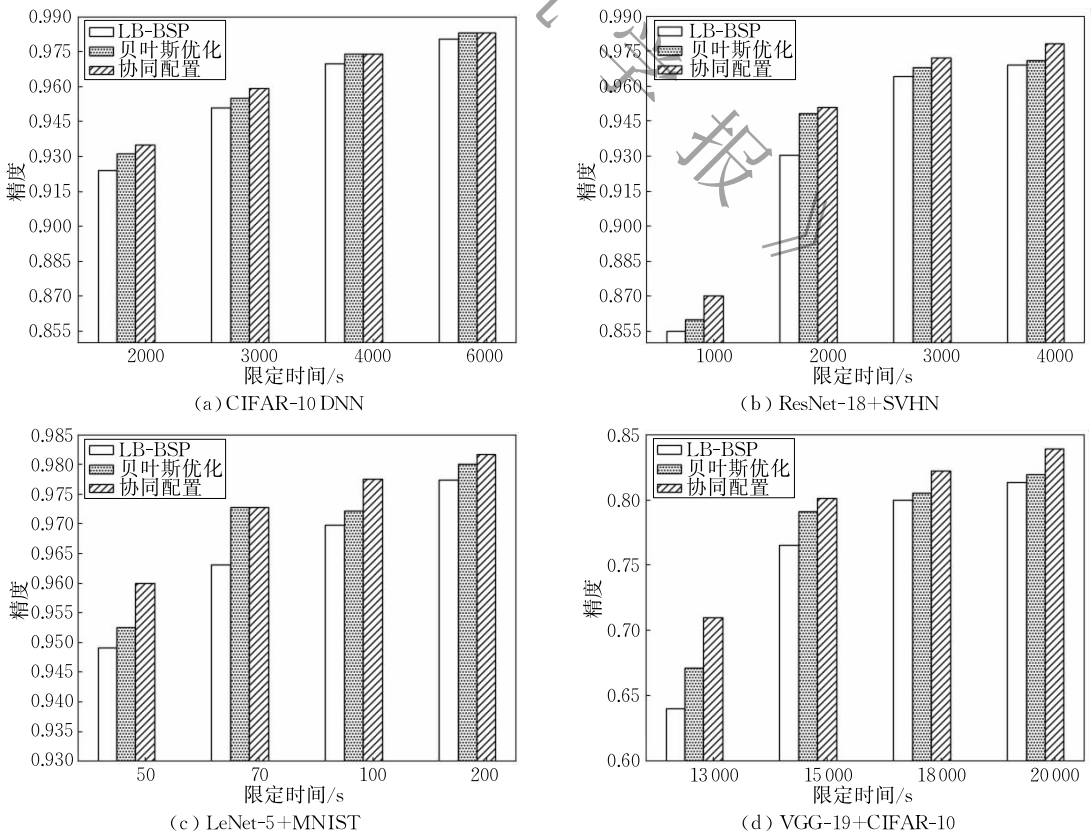


图 5 限定时间条件下模型训练精度对比

4.4　限定精度条件下的性能评测

本节实验在限定训练精度目标的条件下，比较本文所提出的协同配置方法与 LB-BSP 方法、贝叶斯优化配置方法及 Cynthia 方法的模型训练时间. 实验选取各模型训练普遍的精度需求作为限定精度，测算了模型训练达到限定精度需求的最终时间[19]. 针对不同的精度限定，实验对象的参数设置如表 6、表 7、表 8 和表 9 所示，由于 LB-BSP 方法和贝叶斯优化方法仅对批尺寸配置进行优化，因此与 4.3 节相似，在本节实验中上述两类方法的资源配置规模也采用参考协同配置中资源配置优化值的方法.

表 6　CIFAR-10 DNN 参数设置

精度要求	LB-BSP		贝叶斯优化		Cynthia		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
0.945	512	3	16	3	32	2	32	3
0.955	512	3	64	3	32	3	32	3
0.975	512	5	128	5	32	4	64	5
0.985	512	9	256	9	32	8	64	9

表 7　ResNet-18＋SVHN 参数设置

精度要求	LB-BSP		贝叶斯优化		Cynthia		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
0.945	512	3	64	3	32	3	128	3
0.965	512	5	128	5	32	4	64	5
0.975	512	6	128	6	32	6	64	6
0.985	512	7	256	7	32	6	32	7

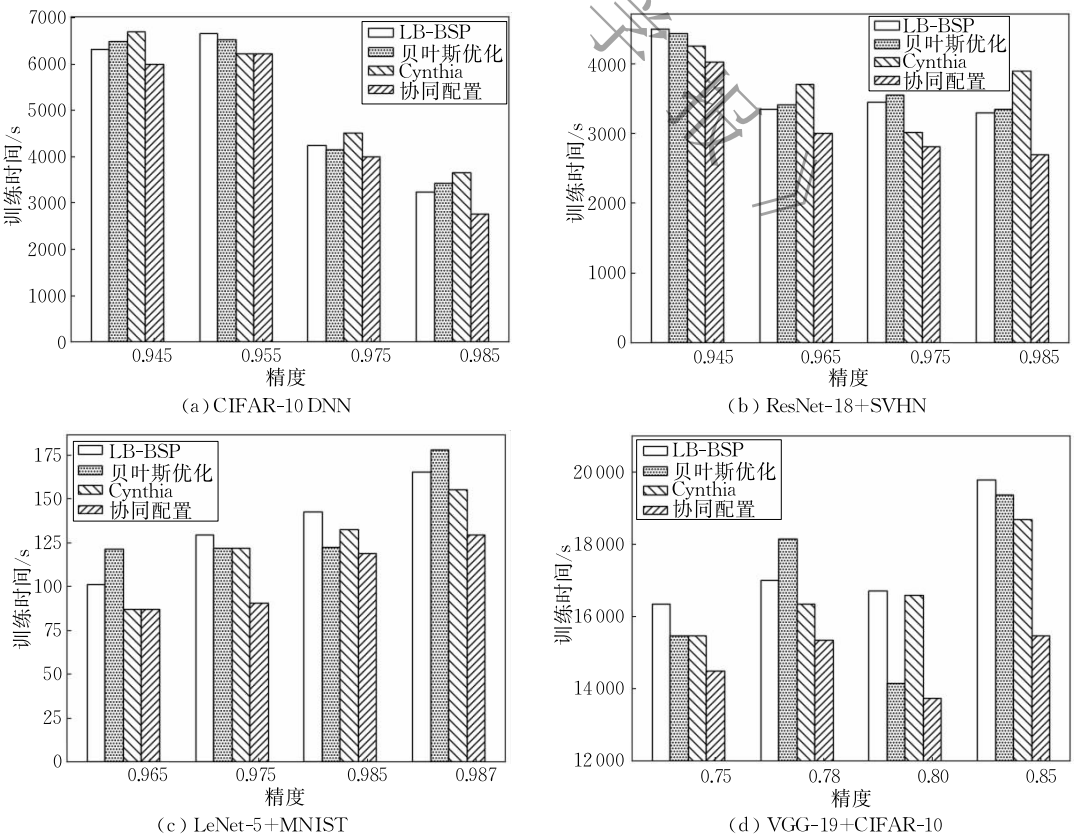
表 8　LeNet-5＋MNIST 参数设置

精度要求	LB-BSP		贝叶斯优化		Cynthia		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
0.965	256	2	16	2	32	2	32	2
0.975	256	2	32	2	32	2	64	2
0.985	256	3	64	3	32	3	128	3
0.987	256	4	64	4	32	4	128	4

表 9　VGG-19＋CIFAR-10 参数设置

精度要求	LB-BSP		贝叶斯优化		Cynthia		协同配置	
	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点	批尺寸	工作节点
0.75	256	2	32	2	32	2	16	2
0.78	256	4	64	4	32	3	16	4
0.80	256	5	64	5	32	4	16	5
0.85	256	6	128	6	32	5	64	6

图 6 给出四个评测对象在限定精度条件下训练总时间比较结果. 图中可以看出，针对不同的训练精度需求、训练模型及数据集，使用本文所提出的协同配置方法可使模型训练总时间较采用 LB-BSP 和贝叶斯优化有不同程度的减少，最大减少了 19.86%，相比 Cynthia 的训练时间最大减少了 16.36%. 这是由于，与本文所提出的协同配置方法相比，LB-BSP 方



(a) CIFAR-10 DNN　(b) ResNet-18+SVHN

(c) LeNet-5+MNIST　(d) VGG-19+CIFAR-10

图 6　限定精度条件下模型训练时间对比

法主要针对工作节点间资源供给非对称的场景，从计算效率而非训练精度收敛效率的角度，进行批尺寸调优.因此，虽然达到各工作节点间的计算负载堆成，但达到限定训练精度的时间较长；贝叶斯优化配置方法关注模型训练的最终收敛精度，但未考虑达到收敛精度的时间和资源成本，因此达到限定精度的时间较长.相比于 Cynthia 方法，本文提出的协同配置方法，充分利用了资源配置与批尺寸配置之间的互补特性，通过批尺寸调优加速训练达到限定精度，缩短了训练时间，提升计算效率.另外，从上述四组模型训练结果可知，协同配置的优势在 CIFAR-10＋VGG-19 测试组中最为明显.这是因为相对于其他训练模型，VGG-19 较为复杂，单次完整训练时间较长，通过扩展资源配置可获得的计算效率提升潜力较大，同时协同配置方法通过优化批尺寸设置确保模型训练提高计算效率的同时具有较好的精度收敛效率.

4.5 面向资源成本的性能评测

本节实验在不同的限定时间和精度要求下，综合从资源成本和精度双目标，评估所提出的协同配置方法.本节将所提出的协同配置方法分别与基于人工经验的配置方法、LB-BSP 方法、贝叶斯优化配置方法及 Cynthia 方法进行比较.

本节实验对于每一个评测对象设置两组受限时间和精度要求.实验配置如表 10 所示.

表 10 协同配置实验参数设置

数据集	模型	时间要求/s	精度要求	最大资源数/个
CIFAR-10	CIFAR-10 DNN	4000	0.963	8
		5000	0.983	
SVHN	ResNet-18	2000	0.947	8
		3000	0.970	
MNIST	LeNet-5	70	0.973	4
		140	0.985	
CIFAR-10	VGG-19	14000	0.800	8
		18000	0.850	

表 11 给出不同配置方法下的训练精度比较结果.其中，对于人工配置，本文选取具有不同经验背景的人员进行测试，以调参次数为单位计算时间；对于 LB-BSP，根据选取的优化批尺寸值，选取可以满足其精度时限要求的最小资源配置；对于贝叶斯优化配置，通过自动搜索，选取达到最优精度的批尺寸配置；对于 Cynthia，通过自动优化得到最优资源配置，并根据文献[18]对批尺寸进行配置.由表可知，在不同的精度时限要求下，本文所提出的协同配置方法均在限定时间内满足训练精度要求，并在资源成本上取得了最优的性能，最大节省资源成本 34.83%.此外，与 LB-BSP 和贝叶斯优化方法比较，本文所提出的协同配置方法可获得的最终训练精度略低，不超过 0.62%.这是由于本文所提出的方法，在确保精度和时间要求的前提下，减少完整训练次数以获得较低的资源成本.反之，LB-BSP 和贝叶斯优化方法虽然获得了较高的训练精度，但耗费了较多的训练时间，最大增加了 24.9%.人工配置法对用户经验要求较高，虽然获得了较好的精度和资源成本，但对于一般用户需通过大量尝试获得最优解，因此配置的时间和人力成本高.Cynthia 在精

表 11 协同配置实验结果

(a) CIFAR-10＋CIFAR-10 DNN 第一组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	256	4642.87	4	0.967	18571.48～22816.15
	64	2529.91	8	0.973	
	256	3259.45	7	0.972	
LB-BSP	512	3612.77	5	0.966	18063.85
贝叶斯优化	256	3349.422	6	0.968	20096.532
Cynthia	32	4095.21	4	0.965	16380.84
协同配置	64	3773.01	4	0.967	15092.04

(b) CIFAR-10＋CIFAR-10 DNN 第二组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	512	6427.96	4	0.982	25711.84～30209.98
	256	3741.62	8	0.985	
	64	3887.14	7	0.985	
LB-BSP	512	5315.81	5	0.983	26597.5
贝叶斯优化	256	3788.224	7	0.986	26517.568
Cynthia	32	4582.73	6	0.984	27496.38
协同配置	128	4064.32	6	0.984	24385.92

(c) SVHN＋ResNet-18 第一组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	32	1739.40	7	0.945	12175.8～14295.2
	512	2135.71	6	0.953	
	128	1661.90	8	0.951	
LB-BSP	512	3417.63	4	0.949	13670.52
贝叶斯优化	128	1969.113	7	0.950	13783.791
Cynthia	32	1939.23	6	0.947	11635.38
协同配置	256	1775.2	6	0.948	10651.2

(d) SVHN＋ResNet-18 第二组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	32	2459.62	7	0.971	15826.86～17217.34
	64	2637.81	6	0.973	
	128	3421.27	5	0.972	
LB-BSP	512	3669.54	4	0.971	14678.16
贝叶斯优化	128	2563.488	6	0.976	15380.928
Cynthia	32	3067.45	5	0.972	15337.25
协同配置	256	2671.23	5	0.970	13356.15

(e) MNIST+LeNet-5 第一组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	32	86.80	2	0.978	173.6～253.4
	64	98.53	2	0.975	
	128	58.85	4	0.977	
LB-BSP	256	68.71	3	0.973	206.13
贝叶斯优化	64	60.58	4	0.977	242.32
Cynthia	32	70.47	3	0.975	211.41
协同配置	64	64.01	3	0.973	192.03

(f) MNIST+LeNet-5 第二组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	32	142.34	3	0.986	364.94～487.2
	256	182.47	2	0.982	
	64	121.80	4	0.987	
LB-BSP	256	157.41	3	0.985	472.23
贝叶斯优化	64	121.80	4	0.987	487.2
Cynthia	32	142.34	3	0.986	427.02
协同配置	128	118.73	3	0.985	356.19

(g) CIFAR-10+VGG-19 第一组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	128	16156.73	4	0.803	64626.92～93390.48
	32	14849.55	5	0.803	
	64	15565.08	6	0.824	
LB-BSP	256	14159.25	6	0.825	84955.5
贝叶斯优化	64	14142.91	5	0.825	70714.55
Cynthia	32	15588.96	5	0.8125	77944.8
协同配置	16	13742.46	5	0.809	68712.3

(h) CIFAR-10+VGG-19 第二组实验对比

优化方法	批尺寸数值	训练时间/s	工作节点资源数/个	训练精度	资源成本
人工配置	128	22914.90	5	0.859	114574.5～142461.62
	256	19765.48	6	0.875	
	64	20351.66	7	0.859	
LB-BSP	256	23039.414	5	0.861	115197.07
贝叶斯优化	128	19351.344	6	0.891	116108.064
Cynthia	32	18686.384	6	0.859	112118.304
协同配置	64	15473.249	6	0.891	92839.494

度和资源成本的结果上最接近协同配置，这是因为该方法的优化目标同时考虑了限定时间和精度，但由于未考虑批尺寸在计算效率和精度收敛效率上的影响，因此该方法得到的配置解不是最优的. 与资源配置相比，协同配置的资源成本最大减少了 16.6%.

实验结果表明，对比基于人工经验为主的手动配置方法，本文所提出的资源-批尺寸参数协同配置方法可使模型训练的资源成本最大节约 34.83%，对比自动化配置的 LB-BSP 方法、贝叶斯优化和 Cynthia，协同配置分别最大降低成本 24.9%、26.89%和 16.6%.

5　相关工作

资源配置和批尺寸超参数优化是提升分布式深度学习模型训练性能的重要途径，但既有工作往往针对两类优化配置开展独立研究. 本节对既有研究工作进行总结分析阐述.

5.1　分布式深度学习系统资源配置研究

面向分布式深度学习的资源配置优化主要分为两种方法：一种是在运行前进行资源需求评估，将评估得到的资源分配给深度学习任务；另一种是在多租户平台上，从作业吞吐的角度上，在运行过程中进行资源的再分配.

对于深度学习单个应用的资源配置优化往往采取第一种方法进行，文献[19]通过预运行阶段采集参数信息，根据模型训练的系统行为建立白盒预测模型对资源需求进行准确预测，并根据预测结果一次性配置资源. 文献[20]也是通过预运行采集的应用信息进行资源配置评估，但是由于其依靠得到的运行数据在配置空间中进行评估，导致时间开销过大. 由于本文所采用的方法是运行前的资源和批尺寸优化配置且需要尽量满足用户的时间效率，因此文献[19]提出的工作与本文最相近.

在多租户平台上，一部分工作依靠统计规律进行资源的动态配置. 文献[21]在训练任务的损失值达到阈值时，通过资源的重新分配来提高训练效率. 文献[22]为每个 Worker 和集群构建一个吞吐量分布，根据分布做出细粒度的资源收缩/扩展决策. 另一部分工作通过性能建模的方法进行资源再分配. 文献[23]和文献[24]在运行时收集作业进度以及资源使用信息，并通过函数拟合的方法预测资源需求以及损失值变化，对资源进行调整. 文献[25]通过监控运行时每个容器内作业的执行进度和资源使用情况，通过建立优化模型进行资源调整决策. 文献[26]通过运行时监控，使用函数拟合的方式为训练任务构建训练时间和训练精度的预测模型，并在进程级对资源再分配. 文献[27]利用集群历史数据建立 LSTM 模型对负载到达强度进行预测，运行阶段以资源成本和资源利用率为目标，使用启发式方法求解最优资源配置. 文献[28]在运行时监控训练损失值，通过函数拟合的方法预测收敛到阈值的时间，使用完成时间公平性指标进行资源再分配，提高整体计算效率. 文献[29]首先对用户提交的应用进行特征分析，并在运行过程中，采集历史数据对所有作业

的资源调度优先级进行调整，加速整体作业的完成速度.综上所述，资源的动态调整依赖于实时监控的运行数据，但是运行过程中的监控将造成运行开销，且本文主要关注运行前资源的一次性分配，因此上述方法不适用于本文.

5.2 分布式深度学习超参数设置研究

针对批尺寸配置优化的工作主要分为四类：第一类工作是自动化超参数优化技术，这类工作首先定义了批尺寸等超参数的搜索空间，然后通过一定的搜索策略搜索到空间内的最优配置解，如黑盒方法中的网格搜索、随机搜索和贝叶斯优化.其中，贝叶斯优化依赖概率分布的数学原理，可以更高效更准确地搜索到最优超参数配置，因此被广泛应用[18,30].这类工作以最终训练精度为目标，无法解决用户面临的最小化资源成本问题.

第二类工作对批尺寸的优化配置进行了定性分析，这类工作通过观测实验以及理论推导，分析了批尺寸配置对深度学习模型的精度收敛影响，并最终给出批尺寸配置的建议.文献[31-32]分析了批尺寸配置对精度收敛效率的影响，并指出对于大多数据集，最优的批尺寸配置为 32.文献[8,13]对批尺寸的泛化性能影响进行了分析，分析结果显示泛化性能与更新次数有关，不同的批尺寸配置会导致相同时间内的更新次数不同.文献[33]指出，通常的优化手段是降低学习率，而通过在训练期间提高批尺寸配置，可在训练集和测试集上取得相同的学习曲线.文献[34]认为虽然将批尺寸设置为小数值可保证大多数用户的模型训练性能，但这样需要更高的通信带宽.

第三类工作从理论和实验两方面，分析批尺寸与学习率对于训练的协同影响，发现最优的批尺寸配置与学习率配置有关.文献[35]认为神经网络的泛化能力与批尺寸和学习率的比值呈负相关.文献[36]认为学习率和批尺寸之间存在较高的相关性，为了确定最佳批处理配置，应该首先尝试较小的批尺寸.

第四类工作优化了批尺寸配置，在运行过程中监控作业的运行信息，通过性能建模的方法决策批尺寸配置的调整值.文献[37]提出一种基于停止准则的动态批尺寸调整方法，当损失值满足停止准则时，通过改变批尺寸大小提高泛化能力.文献[38]根据上一轮训练中每个 Worker 的运行情况动态调整批尺寸的配置，从而提高资源利用率.文献[39]提出一种自适应的批尺寸动态调整方法，在完成一定轮次的完整数据集训练后，通过增加批尺寸配置，提高了收敛效率.然而这类工作仅关注批尺寸对精度的影响，另一方面，对于批尺寸的动态调优依赖运行时的数据监控，批尺寸动态调整造成训练任务的频繁启停，影响训练性能.因此上述方法不适用于本文的批尺寸静态配置优化.

总结而言，目前工作对资源配置和批尺寸配置优化是独立进行的，同时资源的配置优化工作多是面向计算效率，批尺寸的配置优化工作仅关注训练精度.然而，如前所述，资源和批尺寸配置对分布式深度学习训练的计算效率和精度收敛效率均有影响，且存在较强的相关性，独立配置无法实现在限定训练时间内将深度学习模型训练到目标精度并且最小化资源成本的目标.因此，本文将资源和批尺寸配置相结合，探索云数据中心多目标优化问题的解决方案.

6 结 语

针对分布式深度学习系统中，模型训练需同时满足时间受限的精度要求和资源成本最小化需求，本文提出了一种资源-批尺寸协同配置方法.首先量化分析了分布式深度学习系统中资源配置与批尺寸配置对模型训练精度及资源成本的影响；然后提出基于保序回归的模型单轮完整训练时间和训练精度预测模型；接着采用禁忌搜索算法计算资源-批尺寸协同优化配置解；最终采用有代表性的神经网络模型和训练数据集对本文所提出的协同配置方法进行了性能评测.实验结果表明，对比基于人工经验为主的手动配置方法，本文所提出的资源-批尺寸参数协同配置方法可使模型训练的资源成本最大节约 34.83%，对比自动化配置的 LB-BSP 方法、贝叶斯优化和 Cynthia，协同配置分别最大降低成本 24.9%、26.89%和 16.6%.

参 考 文 献

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436-444

[2] Shu Na, Liu Bo, Lin Wei-Wei, Li Peng-Fei. Survey of distributed machine learning platforms and algorithms. Computer Science, 2019, 46(3): 9-18(in Chinese)
(舒娜，刘波，林伟伟，李鹏飞. 分布式机器学习平台与算法综述. 计算机科学，2019，46(3)：9-18)

[3] Chen Xian-Chang. Research on Algorithm and Application of Deep Learning Based on Convolutional Neural Network [M. S. dissertation]. Zhejiang Gongshang University, Hangzhou, 2014(in Chinese)
(陈先昌. 基于卷积神经网络的深度学习算法与应用研究[硕士学位论文]. 浙江工商大学，杭州，2014)

[4] Peng S, Wen Y, Ta N, et al. Towards distributed machine learning in shared clusters: A dynamically-partitioned approach //Proceedings of the 3rd IEEE International Conference on Smart Computing (SMARTCOMP). Hong Kong, China, 2017: 1-6

[5] Lee W Y, Lee Y, Jeong J S, et al. Automating system configuration of distributed machine learning//Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS). Dallas, USA, 2019: 2057-2067

[6] Alipourfard O, Liu H H, Chen J, et al. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics//Proceedings of the 14th Symposium on Networked Systems Design and Implementation (NSDI). Boston, USA, 2017: 469-482

[7] Bao Y, Peng Y, Wu C, et al. Online job scheduling in distributed machine learning clusters//Proceedings of the 37th Conference on Computer Communications. Honolulu, USA, 2018: 495-503

[8] Chen C, Weng Q, Wang W, et al. Fast distributed deep learning via worker-adaptive batch sizing//Proceedings of the 9th Symposium on Cloud Computing. Carlsbad, USA, 2018: 521-521

[9] Hoffer E, Hubara I, Soudry D. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks//Proceedings of the 31st Advances in Neural Information Processing Systems. Long Beach, USA, 2017: 1731-1741

[10] Abadi M, Barham P, Chen J, et al. TensorFlow: A system for large-scale machine learning//Proceedings of the 12th Symposium on Operating Systems Design and Implementation. Savannah, USA, 2016: 265-283

[11] Navarro C A, Hitschfeld-Kahler N, Mateu L. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. Communications in Computational Physics, 2014, 15(2): 285-329

[12] Sun S, Chen W, Bian J, et al. Slim-DP: A multi-agent system for communication-efficient distributed deep learning// Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. Richland, USA, 2018: 721-729

[13] Reddi S, Zaheer M, Sachan D, et al. Adaptive methods for nonconvex optimization//Proceedings of the 32nd Advances in Neural Information Processing Systems (NIPS). Montréal, Canada, 2018: 9815-9825

[14] Xing Wu-Qiang. Isotonic Regression and the Application Research [M. S. dissertation]. Northwestern Polytechnical University, Xi'an, 2002(in Chinese)
(邢务强. 保序回归的研究及应用[硕士学位论文]. 西北工业大学，西安，2002)

[15] Deng H, Zhang C H. Isotonic regression in multi-dimensional spaces and graphs. Annals of Statistics, 2020, 48(6): 3672-3698

[16] Mair P, Hornik K, de Leeuw J. Isotone optimization in R: Pool-adjacent-violators algorithm (PAVA) and active set methods. Journal of Statistical Software, 2009, 32(5): 1-24

[17] Glover F, Laguna M. Tabu Search. Boston, USA: Springer, 1998

[18] Yu T, Zhu H. Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv: 2003.05689, 2020

[19] Zheng H, Xu F, Chen L, et al. Cynthia: Cost-efficient cloud resource provisioning for predictable distributed deep neural network training//Proceedings of the 48th International Conference on Parallel Processing. Kyoto, Japan, 2019: 1-11

[20] Xu E, Li S. Revisiting resource management for deep learning framework. Electronics, 2019, 8(3): 327

[21] Zheng W, Song Y, Guo Z, et al. Target-based resource allocation for deep learning applications in a multi-tenancy system//Proceedings of the 8th High Performance Extreme Computing Conference. Waltham, USA, 2019: 1-7

[22] Or A, Zhang H, Freedman M. Resource elasticity in distributed deep learning. Machine Learning and Systems, 2020, 2: 400-411

[23] Peng Y, Bao Y, Chen Y, et al. Optimus: An efficient dynamic resource scheduler for deep learning clusters// Proceedings of the 13th EuroSys Conference. Porto, Portugal, 2018: 1-14

[24] Zhang H, Stafman L, Or A, et al. SLAQ: Quality-driven scheduling for distributed machine learning//Proceedings of the 8th Symposium on Cloud Computing. Santa Clara, USA, 2017: 390-404

[25] Zheng W, Tynes M, Gorelick H, et al. FlowCon: Elastic flow configuration for containerized deep learning applications //Proceedings of the 48th International Conference on Parallel Processing. Kyoto, Japan, 2019: 1-10

[26] Wang S, Gonzalez O, Zhou X, et al. An efficient and non-intrusive GPU scheduling framework for deep learning training systems//Proceedings of the 33rd International Conference for High Performance Computing. Atlanta, USA, 2020: 1279-1291

[27] Zhang C, Yu M, Wang W, et al. Mark: Exploiting cloud services for cost-effective, SLO-aware machine learning inference serving//Proceedings of the 49th Annual Technical Conference. Renton, USA, 2019: 1049-1062

[28] Mahajan K, Balasubramanian A, Singhvi A, et al. Themis: Fair and efficient GPU cluster scheduling//Proceedings of the 17th Symposium on Networked Systems Design and Implementation. Santa Clara, USA, 2020: 289-304

[29] Gu J, Chowdhury M, Shin K G, et al. Tiresias: A GPU cluster manager for distributed deep learning//Proceedings of the 16th Symposium on Networked Systems Design and Implementation. Boston, USA, 2019: 485-500

[30] He X, Zhao K, Chu X. AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 2021, 212: 106622

[31] Masters D, Luschi C. Revisiting small batch training for deep neural networks. arXiv preprint arXiv: 1804.07612, 2018

[32] Goyal P, et al. Accurate, large minibatch SGD: Training ImageNet in 1 hour. arXiv Prepr. arXiv1706.02677, 2017

[33] Smith S L, Kindermans P J, Ying C, et al. Don't decay the learning rate, increase the batch size//Proceedings of the 6th International Conference on Learning Representations. Vancouver, Canada, 2018: 201-211

[34] Zhang W, Feng M, Zheng Y, et al. GaDei: On scale-up training as a service for deep learning//Proceedings of the 17th International Conference on Data Mining (ICDM). New Orleans, USA, 2017: 1195-1200

[35] He F, Liu T, Tao D. Control batch size and learning rate to generalize well//Proceedings of the 33rd Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 1141-1150

[36] Kandel I, Castelli M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT Express, 2020, 6(4): 312-315

[37] Takase T. Dynamic batch size tuning based on stopping criterion for neural network training. Neurocomputing, 2021, 429: 1-11

[38] Ye Q, Zhou Y, Shi M, et al. DBS: Dynamic batch size for distributed deep neural network training. arXiv preprint arXiv: 2007.11831, 2020

[39] Devarakonda A, Naumov M, Garland M. AdaBatch: Adaptive batch sizes for training deep neural networks. arXiv preprint arXiv:1712.02029, 2017

LIANG Yi, Ph.D., associate professor. Her research interests include AI systems and cloud computing systems.

DING Zhen-Xing, M. S. candidate. His research interest is AI systems.

ZHAO Yu, M. S. candidate. His research interest is AI systems.

LIU Ming-Jie, M. S., engineer. His research interests include cloud computing and high performance computing.

PAN Yong, M. S., senior engineer. His research interest is cloud computing systems.

JIN Yi, Ph. D., senior engineer. His research interests include virtualization technology and cloud computing systems.

Background

Distributed deep learning model training is the popular way to train large-scale deep learning models, which deploys training tasks on multiple computing nodes and executes tasks in parallel so as to accelerate the model training. Previous works on distributed deep learning model training mainly concern on the accuracy and computational efficiency of the model training. However, as both of the volume of training datasets and the complexity of training models go up, the resource cost of deep learning model training increases significantly, and hence, becomes the new concern in the distributed deep learning. How to meet the training accuracy requirement with the time constraint while minimizing the incurred resource cost is an open issue in distributed deep learning.

Most related works take the resource allocation and batch sizing as the independent methods to optimize the computational efficiency and training accuracy of distributed deep learning model training, respectively. However, our empirical study results demonstrate that, from the perspective of resource cost optimization, the configurations of resource allocation and batch sizing have complex interdependence. In detail, with the given resource allocation, the batch sizing can be optimized to accelerate the training convergence and reduce the resource cost. Moreover, the optimized batch sizing varies among different resource allocations. Based on the above study, we establish two isotonic regression-based models for the elapsed time and accuracy predictions of deep learning model training, respectively. We cooperate these two models and adopt Tabu search heuristics to find the optimized configuration combination of resource allocation and batch sizing, which satisfies the requirements of training accuracy, time constraint and resource cost of distributed deep learning model training.

This work is supported by the Beijing Natural Science Foundation (No. 4192007) and the National Key R&D Program of China (No. 2017YFC0803300).