基于异构硬件的LSTM训练系统

黄为新[1,2]，胡伟方[1,2]，曹雪娇[1,2]，石宣化[1,2]
1. 华中科技大学计算机科学与技术学院，湖北 武汉 430074；
2. 华中科技大学大数据技术与系统国家地方联合工程研究中心，服务计算技术与系统教育部重点实验室，湖北 武汉 430074

摘要
在大数据时代，以LSTM为代表的深度神经网络模型具有处理海量数据的能力，在语言处理、语音识别、时序数据预测等领域表现优异。随着模型复杂度的提高，训练成本大幅提升。现有的LSTM训练系统使用了算子融合、多流等加速手段，但忽略了训练算子内部计算的可并行性，导致计算资源的利用率低，整体耗时长。为此，设计了基于细粒度模型划分和多流并行调度方法的LSTM训练系统TurboLSTM，在英伟达GPU和国产昇腾NPU这两种异构硬件上构建的全新底层训练算子实现了任务对计算资源的合理利用。与已有训练系统相比，在GPU上TurboLSTM的单算子训练时间缩短了23%，模型的整体训练时间缩短了17%，在NPU上TurboLSTM的单算子训练时间缩短了15%，且对计算资源的利用率显著提高。这表明提出的加速方案是高效的，具有良好的泛化能力。

关键词
LSTM；训练加速；细粒度并行；多流调度

中图分类号：TP183　　文献标志码：A　　doi: 10.11959/j.issn.2096-0271.2024053

LSTM training system based on heterogeneous hardware

HUANG Weixin[1,2], HU Weifang[1,2], CAO Xuejiao[1,2], SHI Xuanhua[1,2]
1. School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2. National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract
In the era of big data, deep neurals network models represented by LSTM have the ability to process massive data, and have excellent performance in the fields of language processing, speech recognition and time series data prediction. However, with the increase of model complexity, the training cost increases significantly. The existing LSTM training systems use acceleration methods, such as operator fusion and multi-stream, but neglect the parallelism of the internal calculation of a single training operator, which leads a low utilization rate of computing resources and a long traning time. Therefore, this paper designs a training acceleration system called TurboLSTM based on fine-grained model partitioning

method and multi-stream parallel scheduling strategy. A new underlying training operator built on NVIDIA GPU and domestic Ascend NPU heterogeneous hardware realizes reasonable utilization of computing resources for tasks. Compared with the existing training systems, TurboLSTM on NVIDIA GPU has about 23% speed improvement of a single operator and about 17% speed improvement of the overall training time of a model, while TurboLSTM on Ascend NPU has about 15% speed improvement of a single operator, and the significant increase in the utilization of computing resources is observed. This shows that the acceleration method is efficient and has good generalization ability.

Key words
LSTM, training acceleration, fine-grained parallelism, multi-stream scheduling

0 引言

深度学习作为一项能够高效处理和分析海量数据的技术，慢慢走进人们的视野。通过模拟大脑的行为方式，深度神经网络使用类似神经元细胞的网络结构对数据进行学习和理解，从而提取复杂的抽象特征。设计能够处理各种数据的网络模型是深度学习的核心驱动力，然而近几年来，由于数据规模的飞速增长，这种扩展性需求逐渐发展为对高性能的需求。伴随着网络模型的改进和拓展，训练一个完备有效的网络模型所需的计算资源呈指数级增长，面临着成本极高、训练耗时长的挑战。因此，研究神经网络的训练加速至关重要。

本文选取的长短期记忆网络（long short-term memory，LSTM）[1]被广泛用于机器翻译、语言模型、语音识别和时序数据预测等领域，其基本网络结构如**图1**所示。LSTM由输入层、输出层和循环结构组成，循环结构在处理长序列的信息方面具有优势，在每个时间步都能接收和处理数据。循环结构内部包括遗忘门、输入门、细胞状态门、输出门4种控制门结构以及两种非线性激活函数单元Sigmoid和tanh。LSTM的训练过程本质上是对隐藏层状态和细胞状态的计算和更新，并将其在相邻的循环结构中传递。

LSTM训练加速的主要难点在于整个前向和后向传播计算存在大量的时间和空间上的数据依赖，难以实现训练的并行化处理。LSTM训练除了自身结构的参与，还离不开其所在的硬件和软件环境，包括图形处理器（graphics processing unit，GPU）、神经网络处理器（neural network processing unit，NPU）、现场可编程门阵列（field programmable gate array，FPGA）等新型专用加速器和TensorFlow[2]、PyTorch[3]、Caffe[4]等深度学习软件框架。针对不同的环境，要根据异构硬件的特点设计面向LSTM的训练系统，而不同的系统采用的加速方案通常不同。在GPU上，NVIDIA推出的统一计算设备架构（compute unified device architecture，CUDA）设计的LSTM训练系统，采用了算子融合的策略以减少逐点计算带来的调度开销[5]。另一种加速方法则使用了针对深度学习加速的NVIDIA cuDNN库，实现了多层LSTM的跨层可并行性[6]，利用GPU设备的多流特性同时执行不同层的计算。华为的昇腾系列NPU使用的达芬奇计算架构，针对LSTM训练中任务量最高的矩阵运算进行了高度的加速优化[7-9]，并结合存储和运算一体化的特性提升了训练的效率。

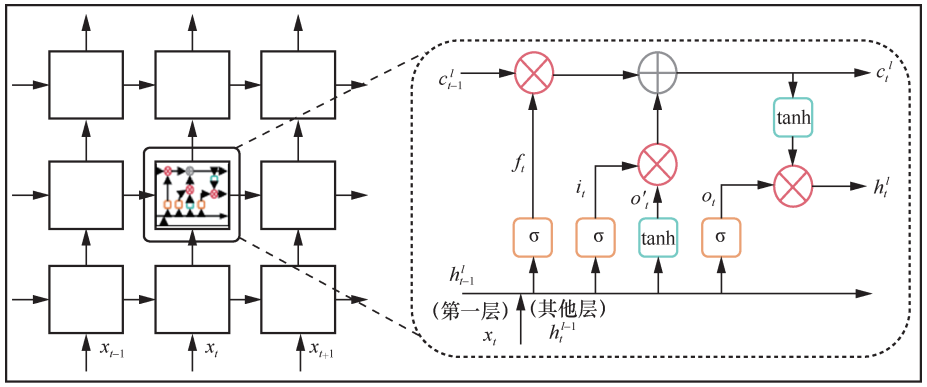

图1 LSTM的基本网络结构

现有的LSTM训练系统使用的加速方案都存在计算资源利用率不高、并行效率低下、部分流上存在空闲时间的问题。因为这些方案都使用了整个梯度计算节点作为并行单元，是一种粗粒度的并行[10]。实际上，每个梯度计算节点内部有多种计算任务，而每个任务自身的紧迫性也不一样。基于以上分析，本文设计了一套基于异构硬件的、以细粒度多流并行方法为核心的LSTM训练系统TurboLSTM。本文的主要贡献如下。

- 提出了一种细粒度的对角线跨层并行策略，通过分析计算子任务之间的依赖和负载对LSTM训练算子进行细粒度划分，充分利用多流特性实现对角线形的跨层并行逻辑。
- 提出了一种基于广度优先搜索的任务映射策略和基于队列的关键任务优先方案，能够减少单个LSTM算子的计算时间，同时充分利用计算资源的低谷，减少端到端的模型训练的时间。
- 在GPU上实现了TurboLSTM训练系统，并在昇腾NPU上完成了对系统的迁移，均观察到一定的加速效果，说明训练系统采用的加速方案的泛化能力良好。

1 研究背景

1.1 LSTM的应用场景

最早的循环神经网络（recurrent neural network，RNN）[11]在训练中容易出现梯度消失或爆炸的问题，也就是在使用基于时间的反向传播训练方法（back-propagation through time，BPTT）时，计算的梯度不断变小趋于0或是梯度不断变大趋于无穷，导致网络性能下降。在加入细胞状态和门控结构后，LSTM作为RNN的变体网络结构，有效缓解了上述问题。随着深度学习技术的发展、数据集规模的增多、算力的增强，LSTM在各个领域发挥作用，具有很大的研究价值。

LSTM设计的编码器-解码器结构，能够捕捉输入和输出序列的复杂关系，被用于大型语料库的学习。这类模型包括Google提出的神经翻译（Google neural machine translation，GNMT）[12]模型、开源的OpenNMT[13]模型等。在语音识别任务中，LSTM可以捕捉信号中的时间依

赖以及语言特征之间的复杂关系，这类模型包括DeepSpeech[14]模型、Tacotron[15]模型等。在最近的研究中，一些新兴领域的计算任务也使用多层LSTM结构训练模型，如金融数据预测[16]、环境数据预测[17]、复杂数据的建模等。不仅如此，很多与LSTM相关的智能应用被开发出来，比如华为的小艺助手、小米的小爱同学和苹果的Siri助手等。

在上述多数复杂任务中，增加LSTM网络结构的深度能有效提高模型的表达能力，实际表现优于单层LSTM的模型。因此，本文针对多层LSTM的训练过程提出一个加速方案。

1.2 LSTM训练的挑战

LSTM的循环结构制约了训练的并行化，这里的并行指的是模型内部的并行，而非数据间并行。输入的同一个序列数据，必须要按照时间顺序一步步计算，循环往复，直到输出一次迭代的结果，这种时间上的强依赖关系使LSTM训练的计算必须串行执行。同时，基于多层LSTM的网络模型越来越多，这些模型在训练多层LSTM时会消耗大量的计算资源。此外，LSTM的训练还涉及大量耗时的矩阵运算，如果训练选择的计算平台没有很好的矩阵加速策略，训练的效率可能较低。

LSTM训练面临的主要难点为并行难、耗时长、成本高，但并行难不代表不能并行。本文的主要目标是在常用的多层LSTM结构中挖掘内部梯度计算之间的可并行性，尽可能充分利用计算资源，设计一套高效快速的LSTM训练系统。

1.3 相关工作

LSTM训练系统在模型训练过程中发挥主要作用的是底层算子，设计一个高效的算子不仅要容纳所有参与训练的计算，还要采用一定的加速方案。本节主要介绍已有LSTM训练系统采用的加速方案及其适用的硬件条件。Li等人[18]根据LSTM计算公式设计了一套基于代数计算的两阶段流水线，分别完成隐藏层状态和输入结果的计算。传统LSTM的结构使得训练难以并行，Hwang等人[19]试图将LSTM定义为一个广义的RNN架构，并将其转换为可并行训练的有向无环图，用一个循环节点来封装循环结构，使用优化图加速训练。xLSTM[20]将记忆单元从标量扩展为矩阵，引入了协方差更新规则，在处理大规模的数据时能够做到完全并行。以上研究通过改进LSTM的结构实现并行，而本文提出的并行方案结合了计算设备的特性来提高训练的并行度。

由**图2**的LSTM训练的计算图可知，在前向传播中，由于给定层的第n个时间步的梯度计算节点只依赖于上一层的第n个时间步的计算节点和本层的第$n-1$个时间步的计算节点，存在无须等待前一层全部算完

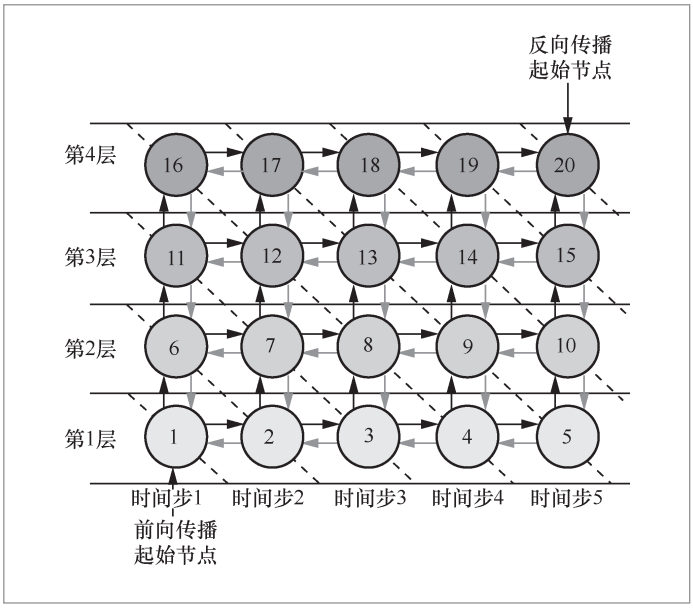

图 2　4 层 5 步的 LSTM 训练

就可以开始当前层任务的可能性。例如，节点1计算完就可以开始节点2和节点6的计算，紧接着是节点3、7、11，此时的前向训练从左对角线向右传播，反向训练同样如此。Sharma基于上述的跨层并行思想在多核CPU架构上实现了对多层RNN波形的训练推理加速，打破了原有的层间串行[21]。理论上，一个4层5步的LSTM进行前向或反向训练原本需要20个时间步，在使用跨层并行逻辑后只需要8个时间步，图中每条虚线表示一个独立的时间步。

上述训练加速方案从LSTM自身的网络特性出发，适用于多个计算平台。除此之外，研究人员结合GPU设备的特性，又提出了CUDA融合算子和cuDNN加速算子。前者在CUDA库的LSTM改进算子中用了算子融合的策略，将多个核函数整合到一个核函数中，避免了不必要的开销，提升了训练的计算效率。后者是目前在GPU上训练速度最快的加速算子，结合了多种优化思想，包括矩阵融合、提前进行矩阵转置、对角线跨层并行、GPU多流技术加速等，但底层算子不开源。

近几年来，不少研究聚焦LSTM硬件专用加速器。例如文献[22]详细地总结了几种硬件加速器，包括FPGA、用于RNN的ASIC，以及硬件加速器的一些优化技术。基于FPGA设计的LSTM加速器主要由加速单元、存储单元、控制单元组成，其中加速单元负责并行完成矩阵运算。在国产的昇腾NPU芯片上，由于软件生态栈正在构建和逐步完善，大多数研究侧重于满足平台的扩展性需求，还没有针对昇腾的LSTM加速方案。Wang等人[23]分析了现有昇腾LSTM训练系统的性能瓶颈。

2 LSTM训练问题分析

2.1 GPU与昇腾NPU的架构

本文实现的LSTM训练系统选取了两种异构硬件，分别是英伟达的GPU和华为的昇腾NPU，见**表1**。

昇腾的AI计算异构引擎由多个CPU核心和多个AI Core组成，其中AI Core是华为自研的面向AI计算特征的达芬奇计算架构，支持存储、计算一体化，能够解决传统数据传输的瓶颈问题，功耗更低。昇腾NPU相对于GPU在推理上更有优势，但在模型训练方面仍有较大的进步空间。为了加速LSTM模型的训练进程，GPU使用算子融合的技术开发了CUDA算子lstm_cell_backward，使用多流技术设计了cuDNN库算子cuDNNLSTMBackward；而昇腾NPU也集成了多NPU流技术，开

表1 英伟达 GPU 与昇腾 NPU 的对比

环境	GPU	昇腾NPU
计算架构	Pascal，Volta，Ampere等	达芬奇架构
软件生态栈	CUDA架构	CANN架构
存储特性	存储、计算分离式	存储、计算一体化
多流技术	CUDA流	NPU流
计算调度单元	CUDA计算核心	AI Core
支持的精度模式	半精度、单精度、双精度	半精度、单精度
LSTM训练算子	lstm_cell_backward（CUDA库） cuDNNLSTMBackward（cuDNN库）	BasicLSTMCellCStateGrad

发了BasicLSTMCellCStateGrad算子。但在两种异构硬件上的LSTM训练仍存在问题，需要通过一系列测试进行深入分析。

2.2 在GPU上训练LSTM

为了观察在GPU上训练LSTM的性能，实验选择了NVIDIA的P100和V100，训练的模型是GNMT和OpenNMT翻译模型，测试的对象是CUDA加速库编译的PyTorch训练系统（以下简称Py+CUDA）和cuDNN加速库编译的PyTorch训练系统（以下简称Py+cuDNN）的底层LSTM加速算子。实验通过NVIDIA性能分析工具（NVIDIA visual profiler，nvprof）获取底层算子的执行性能。

实验观察到Py+cuDNN训练LSTM模型的速度比Py+CUDA的更快，但是两者的资源利用效率都不高，而在V100架构上的表现优于P100。CUDA设计的LSTM训练算子将所有的核函数调度到一个默认流上，没有考虑内部计算的可并行性。GPU的资源利用率在大多数情况下小于30%，只有少数情况达到60%，峰值和低谷存在较大差异。cuDNN设计的算子使用多流技术进行跨层并行，并行单元是整个梯度计算节点。在nvprof中观察到部分流在大多数情况下是空闲的，并且每经过一次迭代就会重新创建和释放已有的多流，训练时GPU的资源利用率仍然不高，大多数情况低于60%。

2.3 在昇腾NPU上训练LSTM

为了观察在昇腾NPU 910B上训练LSTM的性能，实验采用的测试模型是GNMT，测试的对象是采用CANN库编译的MindSpore深度学习框架组成的训练系统，同样集成了LSTM加速算子。实验通过Profiler接口在训练过程中自动收集性能数据，并通过可视化工具Mindinsight分析4层GNMT模型的训练过程。

表2展示了在昇腾910B训练模型的数据，**图3**则展示了在训练时底层算子的耗时情况。实验观察到矩阵运算的算子数量非常多，同时矩阵运算的耗时占了算子总耗时的74.7%。而昇腾NPU对矩阵、向量、标量的运算做了高度优化，在这3类运算上的性能表现优于GPU。实验还观察到所有的矩阵运算被串行地安排在多流中，创建的多流并没有在矩阵运算并行中发挥作用，任务对NPU的资源利用率都在50%~60%。

由上述实验中可知：①GPU上现有的LSTM训练系统在加速训练的过程中，存在的问题包括反复创建和释放CUDA流带来较大开销、对计算资源利用率不高、CUDA流存在空闲、并行单元包含许多计算使得并行效率低下；②NPU上现有的LSTM训练系统在加速训练的过程中，存在的问题包括创建的多流无法起到并行计算的效果、对计算资源利用率不高、未利用跨层并行的算法思想。

表 2　Ascend 910B 的模型训练性能

模型	框架	数据集	层数	多流个数	算子数目	单step时延/s
GNMT	MindSpore	WMT	2	1	285	0.792
GNMT	MindSpore	WMT	4	2	495	1.438
GNMT	MindSpore	WMT	8	3	915	2.723
GNMT	MindSpore	WMT	16	6	1 756	5.327

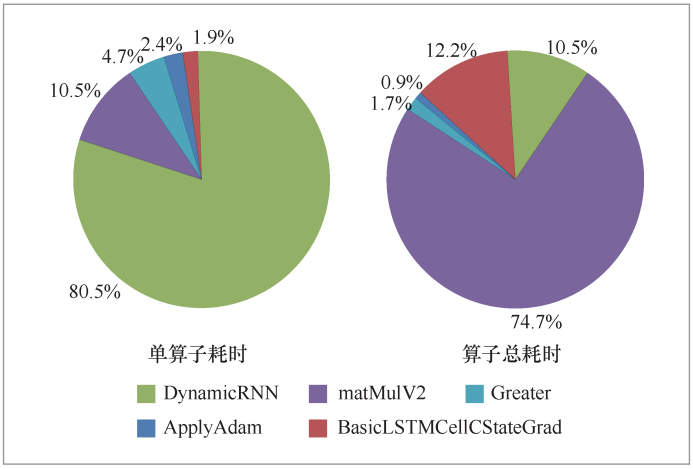

图 3 在 LSTM 训练中底层算子的耗时

3 基于细粒度模型划分和多流并行调度的训练加速方法

在设计LSTM训练系统的具体加速方案前，本节首先对关键任务和非关键任务的定义进行说明[24]。如果一个计算的结果在当前层需要同步并作为下一层输入，说明其紧迫程度较高，被视为关键任务；如果一个计算的输出在下一个迭代才被需要，说明其依赖距离较长，被视为非关键任务，并且在当前迭代的任意时刻都可执行。在多层LSTM的反向训练过程中，每个梯度计算节点内部都有多种类型的子计算，包括计算隐藏层梯度的状态、计算细胞状态的梯度、计算偏置和计算门控中间状态。在现有的加速方法中，这些子计算总是串行的。若对内部子计算进行分类，优先计算关键子任务就可以快速开始下一层的训练，同时延迟非关键任务的执行，将它们安排在计算资源空闲时或者是关键子任务的同步过程中，这样能够最大限度覆盖非关键任务的计算时间，从而缩短总的训练时间。

图4展示了本文提出的LSTM训练系统TurboLSTM的总体设计。该系统抛弃了深度学习框架自身的调度方法，而是把多层LSTM的训练计算放到一个大算子中，采用细粒度划分和多流并行调度方法来提高资源利用率。

3.1 层内细粒度任务划分策略

首先，对任务进行分类。LSTM反向传播过程中的一个梯度计算节点包含多种功能的子任务：任务1是核心梯度计算任务，含隐藏层状态的梯度和细胞状态的梯度；任务2是中间梯度计算任务，含4个控制门状态的梯度，只在隐藏层内部执行；任务3是参数梯度计算任务，含权重和偏置的梯度；任务4是梯度更新任务，根据计算的梯度结果更新下次前向传播所需的参数值；任务5是数据传输任务，负责在CPU和协处理器之间转移数据。分析计算任务的特

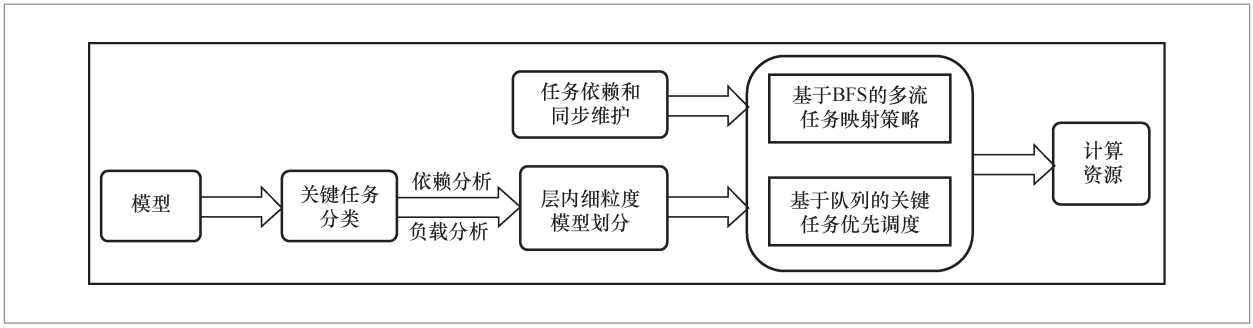

图 4 TurboLSTM 训练系统的总体设计

性可知，任务1和任务2是关键任务，只有其计算完成才能进行下一层的训练；而任务3和任务4是非关键任务，只需要在下次迭代需要用到它的输出结果之前完成即可；任务5同样是非关键任务，传输时间可以和计算重叠。但分类后的每个子任务只包含一个梯度的计算，任务间的负载不均衡使得并行效率较差，可能需要对部分子任务进行融合，因此需要根据数据依赖和计算负载来确定最终合理的细粒度划分方案。

然后，对层内的每个梯度计算节点进行细粒度的划分。通过对内部的各个计算的数据依赖分析，如果一个子计算任务与另一个子计算任务之间没有依赖路径可达，那么使用一定的调度策略可同时执行2个计算任务。此外，还需要考虑划分前每个计算的任务负载，尽可能保证在划分后，两个子任务所需的执行时间、内存、计算资源相差较小，否则会造成负载不均衡，使得计算资源在一些碎片时间内无法被利用。

最后，进行细粒度划分，将一个节点的梯度计算任务分成5类子任务，如**图5**所示。其中，l和t是计算任务所处的层数和时间步，权重x和权重h分别是相对于输入和隐藏层的权重梯度。由**图5**可知，细胞状态梯度和4个控制门的梯度计算需要串行，合并为细胞任务。相比权重的梯度计算量，偏置梯度的计算量可以忽略不计，因此可以合并为权重任务2。统计发现，权重任务1和权重任务2的负载相差不大，同时隐藏层状态任务1和2的运算量相似，这种划分方案能够很好地维持任务间的平衡，并行效果好。此时，关键任务包括细胞任务、隐藏层状态任务1和隐藏

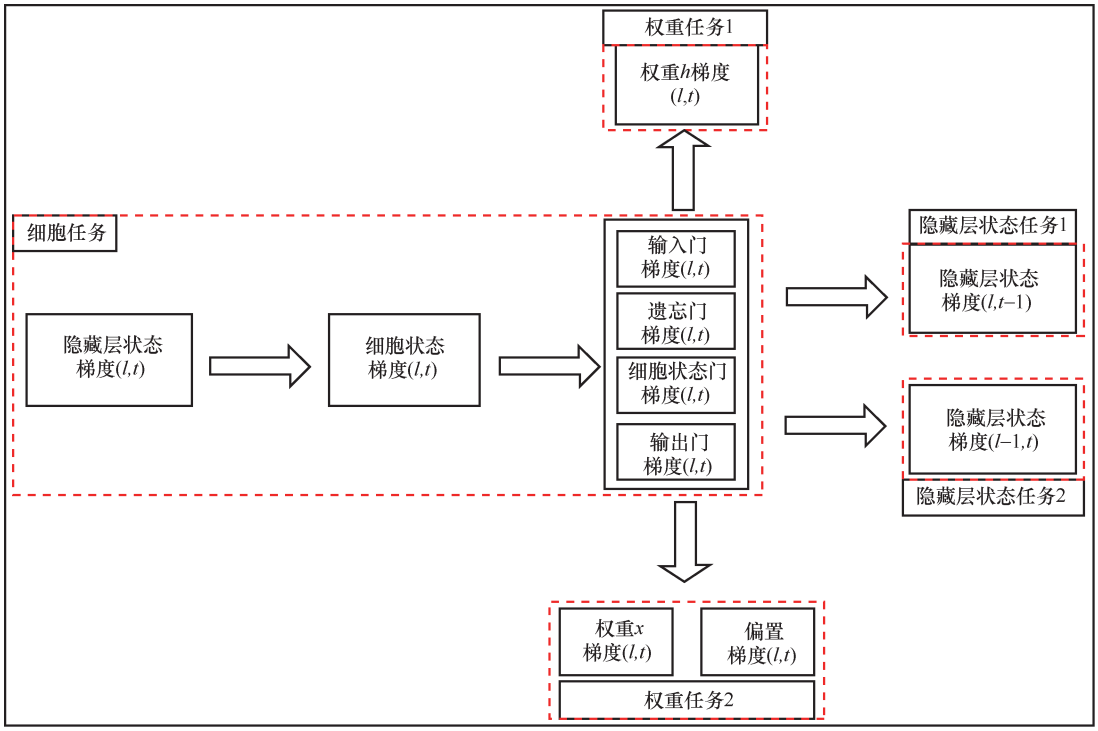

图 5　细粒度的模型划分效果

层状态任务2，这些任务执行完成后就可以开始下一个时间步或者下一层的计算任务。

3.2 层间对角线多流并行调度

在LSTM层间的加速用到了两种策略，第一种是基于队列的关键任务优先调度策略。为了体现关键任务的优先级，定义用来调度关键任务的流为主要流，创建时设定优先级为high；调度非关键任务的流为次要流，创建时设定优先级为low。关键任务会优先调度到主要流上，次要流虽然同时会安排非关键任务，但由于次要流的优先级较低，运行时的计算资源会优先分配给关键任务。本文提出的加速方案使用了两个队列，一个是参与广度优先搜索（breadth-first search，BFS）、提供逐层保存邻节点功能的关键任务队列，一个是用于保存还未被调度的任务的非关键任务队列。使用队列的好处是能够保证关键任务的计算顺序，延后或者重叠非关键任务的计算时间。

第二种加速策略是基于BFS的多流任务映射。首先，细粒度划分后需要重新构建反向训练的计算图，保证每一个子计算节点都能被合理地安排在多流上执行。由于新的计算图相对更规则，采用从起始顶点开始逐层向外扩展的BFS遍历，其遍历的所有子计算节点都在同一个对角线上，这恰好满足了对角线跨层并行策略的要求。其次，为每个子计算节点增加一个新的属性，就是实际调度时指定的流序号。然后，在运行时每遍历一个新的LSTM层，就动态创建两条主要流。因为关键任务的隐藏层状态任务1和任务2可以并行，层内的最大并行度为2。流动态创建能够防止创建过多的流导致训练的性能下降，当流的数量达到设定的阈值时，不再开辟新流，将新的子计算任务放到已经创建好的流上。最后，当BFS遍历到非关键任务时，将其保存到一个队列中延后执行。在运行时通过流查询的接口将非关键任务映射到主要流的空闲时刻或者次要流上，这可以重叠关键任务和非关键任务的计算时间，提高并行度。

在上述的讨论中，还需要重点注意任务依赖和同步维护的实现。同一个流上的子任务的执行顺序是串行的。如果不同流上的子任务之间存在依赖关系，需要使用事件机制来维护依赖。当n层LSTM的输入序列的长度为t时，需要创建$2n+1$条多流，并开辟了$2n\times t$个同步事件。在GPU的某个流中记录一个事件，首先通过cudaEventRecord方法对计算做标记从而判断其是否执行完毕，然后使用cudaStreamWaitEvent在另一个流中等待被标记的事件。非关键任务涉及的参数同步，被延迟至下一个迭代开始前。

算法1详细描述了本文提出的基于细粒度多流并行的LSTM训练加速算法。其中，对模型训练任务的细粒度划分是在CPU上执行的，划分的结果只与网络结构的特征相关，而各个子任务被调度到多流上并行是在GPU上执行的。**图6**展示了TurboLSTM训练系统的算子在GPU上的运行效果，其中绿色部分是关键任务，蓝色和紫色部分是非关键任务。整个算子的执行分为两阶段，一个阶段是使用多流技术重叠关键任务和非关键任务的计算时间；另一个阶段是非关键任务的快速并行，当关键任务按照对角线并行的方式全部执行完成后，就可以利用所有已创建好的流快速地清空非关键任务的队列。针对层数增加或者输入增加导致时间步变长的问题，TurboLSTM的加速方案能够重叠更多的任务从而减少计算开销，提升并行度和资源的利用率。

算法 1：细粒度多流并行算法

输入：输入长度为 T 的序列 $\{x_1,x_2,\cdots,x_T\}$，隐藏层数为 L 的 LSTM 结构

关键任务队列 q，非关键任务 task_queue 均存储有序对（层序号，时间步序号）

创建并初始化 GPU 流数组 s、事件数组 e、标记数组 visited
while q非空 **do** //BFS 遍历所有任务，关键任务优先调度，非关键任务送入队列
    q 弹出头节点 n，节点的层数为 l，时间步数为 t
    计算节点 n 的索引 index $=(L-1-l)*T+T-1-t$
    if 节点 n 在 visited 未遍历 **then**
        if 节点 n 存在等待的事件 **then**
            当前流等待 $e[\text{index}-T]$
        end if
        调度细胞任务 (l,t)、隐藏层状态任务 1 (l,t) 到流 $s[2*l]$
        在流 $s[2*l]$ 上记录事件 e[index]，调度隐藏层状态任务 2 到流 $s[2*l-1]$
    end if
    $(l-1,t)$ 和 $(l,t-1)$ 加入 q，$(l,t,1)$ 和 $(l,t,2)$ 加入 task_queue
end while
初始化查询的流 qs //非关键任务开始从队列调度到空闲流中
while task_queue非空 **do**
    task_queue 弹出头节点 $n1$、$n2$，节点的层序号为 l，时间步序号为 t
    while qs $<2*L-1$ **do**
        if qs 查询的结果是 cudaSuccess，表示空闲 **then**
            调度权重任务 1 (l,t) 和权重任务 2 (l,t) 到流 qs 上
        else
            if qs $<2*L$ **then** qs++
            else qs 赋值为 0，休眠
            end if
        end if
    end while
end while

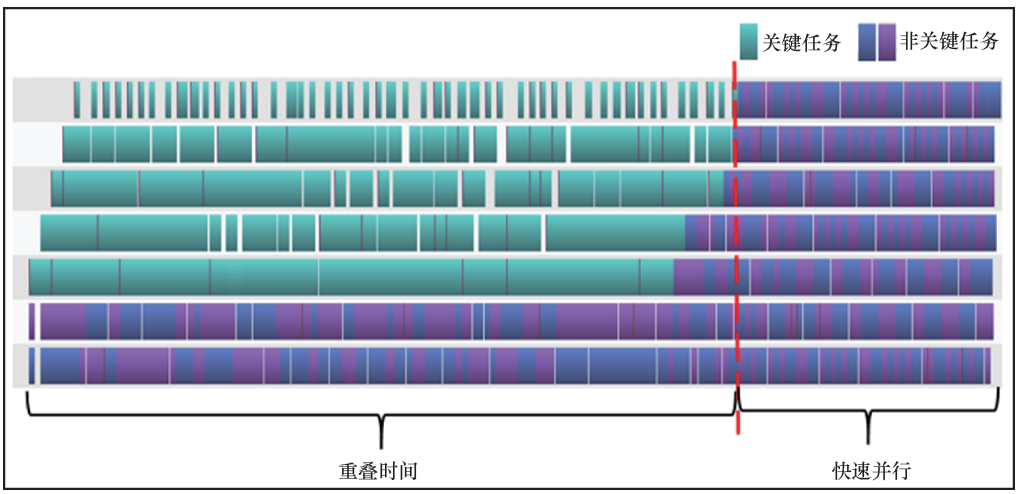

图 6 TurboLSTM 系统的底层运行效果

3.3 针对昇腾NPU架构的映射和调度

基于细粒度模型划分和多流并行算法的训练系统TurboLSTM，先在GPU上对其进行了设计与开发，再将TurboLSTM迁移到昇腾AI处理器上实现。结合昇腾NPU的硬件特性，首先对整体的加速方法进行调整。在GPU和NPU上进行并行优化的思路都是将计算映射到多流上同时执行，而不同之处在于，在GPU上的并行单位是细粒度划分后的5类子任务，而在NPU上的并行单位则是每个子任务对应的具体矩阵乘法算子，粒度更小。因为达芬奇架构针对矩阵计算单元设计了专用的定制电路和后端优化策略来实现加速，能够直接将矩阵乘法算子传送至矩阵计算单元上执行。此外，昇腾NPU上现有的LSTM训练系统并没有用到跨层并行的逻辑，TurboLSTM提出的基于BFS的多流任务映射方法可以解决这一问题。昇腾NPU还具有存储和计算一体化的特性，利用这一特性可以将训练过程中能够复用的大量中间梯度结果直接存放到输出缓冲区。

TurboLSTM训练系统的核心工作是将LSTM多个层的计算整合为一个能够内部并行的底层训练算子。算子需要用Ascend CL计算语言重新编写才能调度到昇腾NPU上，并且其实现方法与GPU不同，具体表现为：使用CANN加速库提供的方法替代相应的CUDA方法，在做矩阵运算时使用CANN中的CBLAS接口的aclblasGemmEx方法取代cublasSgemm方法[25]，并增加了可调度的流选项；创建多流时，使用aclrtStream替代CUDA Stream，事件使用aclrtEvent替代CUDA Event；在创建流之前必须先创建当前线程的上下文容器Context，以方便管理流、事件、设备内存等对象的生命周期；数据传输和内存分配的方法也有所不同，例如使用aclrtMalloc和aclrtFree分配和释放内存。昇腾加速库的众多方法还保留了GPU加速库的特征，降低了开发者的学习成本。昇腾还提供了更加便捷的单算子测试软件MindStudio，只需要准备算子模型文件和相关的数据，即可利用昇腾NPU的算力进行测试。在昇腾NPU上可以方便快速地测试LSTM训练算子的性

能，无须再整合到具体模型和框架中进行测试。

4 实验结果与分析

4.1 实验设置

实验设置见**表3**，包括两种异构硬件（GPU和昇腾NPU）的参数、已有LSTM训练系统的软件环境和用于测试的LSTM模型及数据集。

4.2 GPU实验

图7展示了在P100上Py+cuDNN和TurboLSTM训练系统的底层LSTM算子的训练时间，本文提出的加速方案的训练时间均缩短为Py+cuDNN的75%左右。

图8展示了在两种GPU设备上不同训练加速系统的模型训练速度。Py+cuDNN的模型训练速度相比Py+CUDA有明显改进，因为后者采用了CUDA多流技术。与Py+cuDNN相比，TurboLSTM系统在两种GPU上的训练速度都有所提升。因为前者没有重叠反向传播中权重和偏置的计算时间以及隐藏层状态梯度的时间，二者是顺序执行的，而对于TurboLSTM系统来说，只要输入数据的长度在一定的范围内，大多数权重和偏置的计算时间可以被隐藏层梯度的计算时间覆盖。此外，GNMT在V100架构的GPU上的训练速度相较于P100有明显提升，而OpenNMT的训练速度在两种GPU上相差不大。

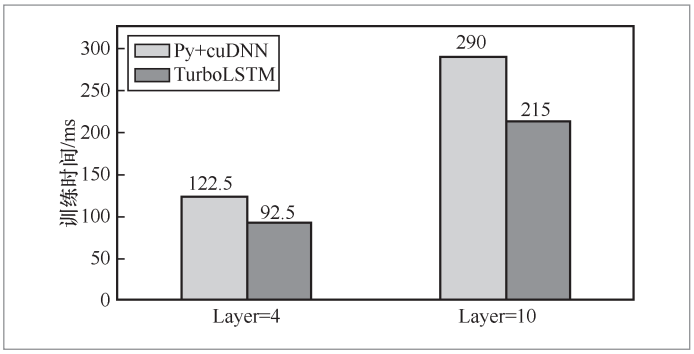

图 7　在 P100 上 Py+cuDNN 和 TurboLSTM 的单算子训练时间

本实验还对TurboLSTM系统中影响训练算子执行效率的因素进行了测试，包括LSTM结构的层数和模型输入的batch_size。由**图9(a)**可知，当层数较小（层数≤4）时，TurboLSTM与Py+cuDNN系统所需的训练时间差异并不明显，因为内核函数的启动开销占据了很大一部分的运行时间。但随着层数的增多，TurboLSTM的底层算子开始发挥其优势，因为多流技术

表 3　实验设置

类别	项目	细节			
硬件	型号	GPU P100	GPU V100	昇腾Ascend 910B	
	架构	Pascal	Volta	达芬奇架构	
	显存	16 GB	32 GB	64 GB	
	协同CPU	Intel XeonE5-2690	Intel Xeon Gold 5117	鲲鹏CPU	
	CPU主频	2.40 GHz	2.00 GHz	2.6 GHz	
软件	操作系统	Ubuntu-16.04.1		OS Euler2.8	
	深度学习框架	PyTorch 1.7.0		MindSpore-v2.1.0	
	算子加速库	CUDA v10.1.243和cuDNN v8.0.5		CANN v7.0.0	
	编译语言	GCC v7.3.0、Python v3.6.9		GCC v7.5.0、Python v3.7.5	
测试模型	模型	GNMT	OpenNMT	GNMT	情绪LSTM[26]
	数据集	wmt16-de-en	toy_en_de	wmt16-de-en	aclImdb_v1和Glove

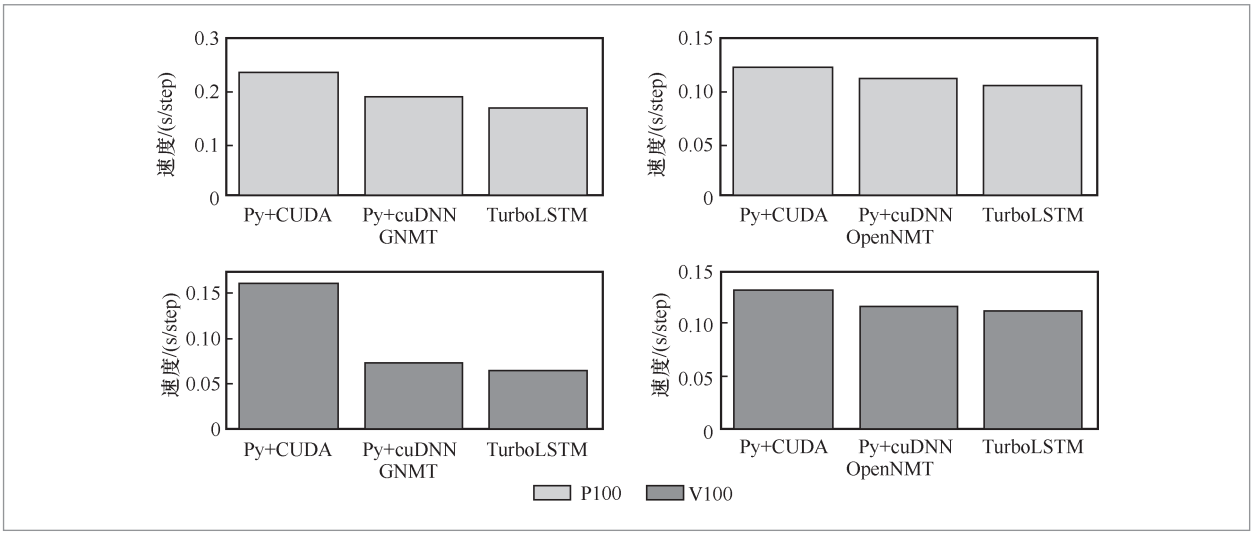

图 8 在两种 GPU 设备上不同训练加速系统的模型训练速度

能够合理并行调度各个细粒度划分的子任务。由图9(b)可知，随着batch_size的增加，算子的加速比有所下降，因为当一个批次的数据放到一个核函数中执行时，batch_size的增加会直接影响子任务的计算量，而非关键任务和关键任务执行的重叠时间会缩短，导致加速比下降，但整体性能还是优于cuDNN的算子。

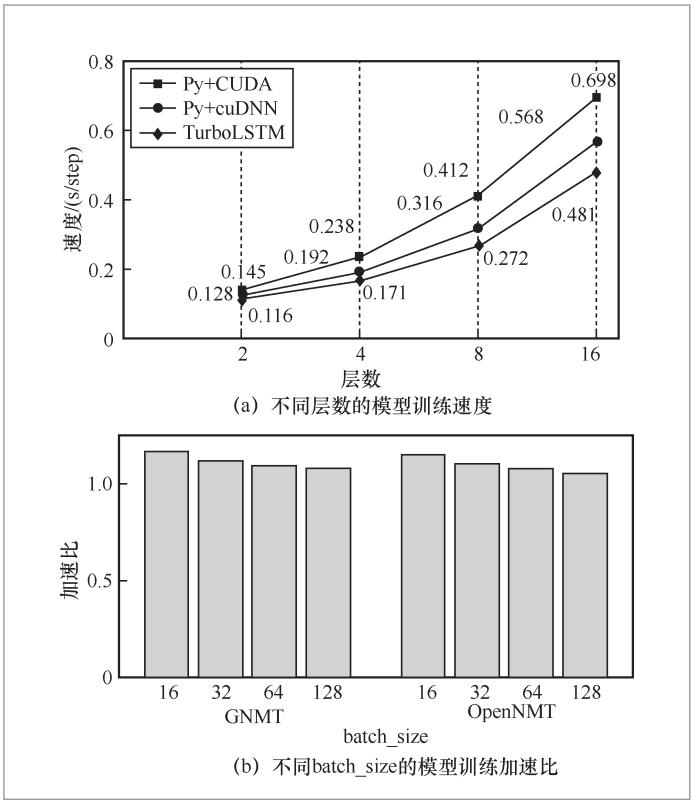

图 9 在 P100 上测试 TurboLSTM 系统的训练算子影响因素

4.3 NPU实验

在原有NPU上CANN算子库实现的LSTM训练加速系统(以下简称CANN)和本文提出的TurboLSTM系统在不同层数上的单算子训练时间如图10所示，在LSTM层数为2、4、8和16时，训练时间分别缩短了12%、15%、24%和28%。本文也测试了不同batch_size对算子运行时间的影响，其效果类似于图9(b)，即随着batch_size的增加，加速效果降低。

图11展示了CANN和TurboLSTM系统的单算子的NPU利用率。由图11可知，相较于CANN系统的原算子，TurboLSTM系统的加速算子的NPU利用率有较大提升，因为加速算子增加了每一时刻同时执行的矩阵算子的数量。综上所述，本文提出的基于细粒度多流并行的TurboLSTM训练系统

在GPU和NPU两种异构硬件上均有一定程度的速度提升和利用率提升，这说明该方法具有较好的泛化能力，未来可应用于更多计算资源平台。

5 结束语

针对现有LSTM训练系统的资源利用率低下和耗时长的问题，本文设计了基于细粒度模型划分和多流并行调度策略的LSTM训练系统TurboLSTM。该方法根据计算间的依赖关系和负载情况，对层内的梯度计算节点进行细粒度划分，为划分后的子任务分配优先级，采用广度优先搜索的策略将优先级高的关键任务映射到多流上执行，并将非关键任务动态地分配到空闲的流上执行，最后在NVIDIA GPU和昇腾NPU两种异构硬件上测试TurboLSTM系统。实验结果表明，与Py+cuDNN系统相比，在GPU上TurboLSTM的单算子训练时间缩短了23%，模型的整体训练时间缩短了17%，在NPU上TurboLSTM的单算子训练时间缩短了15%。这进一步验证了本文提出的训练加速方案的有效性和良好的泛化能力。

在面向国产昇腾平台的TurboLSTM系统迁移实验中，笔者发现以下问题：目前昇腾的软件生态栈还需要完善，NPU无法支持一些经典的LSTM模型；多个NPU集群的模型训练速度优于GPU V100，但在单个设备上训练LSTM模型V100的速度更快；基于昇腾开发算子工程的学习成本较高，大部分核心算子还未开源。

本文提出的训练加速方案也需要进一步研究，当层数非常大时，参数规模也非常大，需要进一步考量如何维持多流及事件的同步开销和更高的并行度之间的平衡以及负载对训练性能的影响。此外，

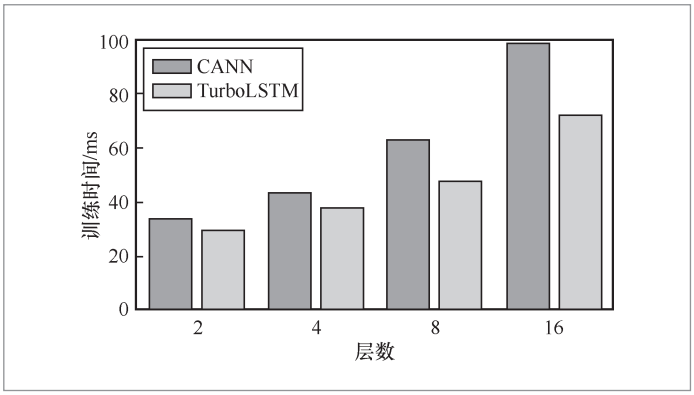

图 10 CANN 和 TurboLSTM 系统在不同层数上的单算子训练时间

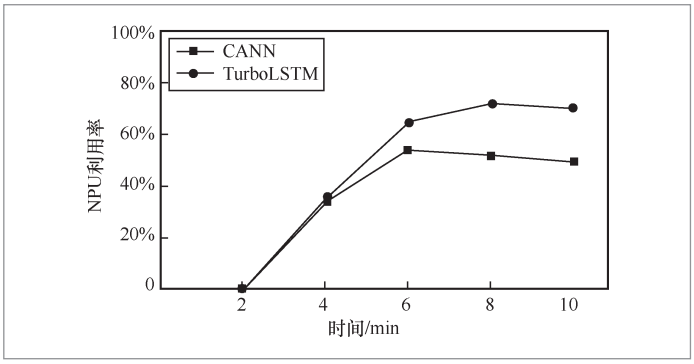

图 11 CANN 和 TurboLSTM 系统的单算子的 NPU 利用率

TurboLSTM提出的加速方法适用于更多的神经网络结构，如BLSTM、xLSTM等变体LSTM网络，卷积神经网络和图神经网络等，需要结合网络特征对本加速方案进行调整。

致谢

非常感谢华为武汉超算的赵佳昕老师、朱思海老师的技术支持以及段欣宇老师的技术指导。

参考文献:

[1] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural

Computation, 1997, 9(8): 1735–1780.

[2] ABADI M, AGARWAL A, BARHAM P, et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems[C]//Proceedings of the 12th USENIX Conference on Operating System Design and Implementation. Savannah: USENIX Association, 2016: 265–283.

[3] PASZKE A, GROSS S, MASSA F, et al. PyTorch: an imperative style, high-performance deep learning library[EB]. arXiv preprint, 2019, arXiv: 1912.01703.

[4] JIA Y Q, 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: ACM, 2014: 675–678.

[5] BRAUN S. LSTM benchmarks for deep learning frameworks[EB]. arXiv preprint, 2018, arXiv: 1806.01818.

[6] APPLEYARD J, KOCISKY T, BLUNSOM P. Optimizing performance of recurrent neural networks on GPUs[EB]. arXiv preprint, 2016, arXiv: 1604.01946.

[7] 鲁蔚征, 张峰, 贺寅烜, 等. 华为昇腾神经网络加速器性能评测与优化[J]. 计算机学报, 2022, 45(8): 1618–1637.
LU W Z, ZHANG F, HE Y X, et al. Evaluation and optimization for Huawei ascend neural network accelerator[J]. Chinese Journal of Computers, 2022, 45(8): 1618–1637.

[8] 梁晓峣. 昇腾AI处理器架构与编程: 深入理解CANN技术原理及应用[M]. 北京: 清华大学出版社, 2019.
LIANG X Y. Ascend AI processor architecture and programming: principles and application of CANN[M]. Beijing: Tsinghua University Press, 2019.

[9] 于璠. 新一代深度学习框架研究[J]. 大数据, 2020, 6(4): 69–80.
YU F. Research on the next-generation deep learning framework[J]. Big Data Research, 2020, 6(4): 69–80.

[10] TALLADA M G. Coarse grain parallelization of deep neural networks[C]//Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York: ACM, 2016: 1–12.

[11] HOPFIELD J J. Neural networks and physical systems with emergent collective computational abilities[J]. Proceedings of the National Academy of Sciences of the United States of America, 1982, 79(8): 2554–2558.

[12] WU Y H, SCHUSTER M, CHEN Z F, et al. Google's neural machine translation system: bridging the gap between human and machine translation[EB]. arXiv preprint, 2016, arXiv: 1609.08144.

[13] KLEIN G, KIM Y, DENG Y, et al. OpenNMT: neural machine translation toolkit[C]//Proceedings of the 13th Conference of the Association for Machine Translation in the Americas. Boston: Association for Machine Translation in the Americas, 2018: 177–184.

[14] AMODEI D, ANUBHAI R, BATTENBERG E, et al. Deep speech 2: end-to-end speech recognition in English and mandarin[EB]. arXiv preprint, 2015, arXiv: 1512.02595.

[15] WANG Y, SKERRY-RYAN R J, STANTON D, et al. Tacotron: towards end-to-end speech synthesis[EB]. arXiv preprint, 2017, arXiv: 1703.10135.

[16] GÜLMEZ B. Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm[J]. Expert Systems with Applications, 2023, 227: 120346.

[17] WANG H, YANG J C, CHEN G Z, et al. Machine learning applications on air temperature prediction in the urban canopy layer: a critical review of 2011–2022[J]. Urban Climate, 2023, 49: 101499.

[18] LI B X, ZHOU E J, HUANG B, et al. Large scale recurrent neural network on GPU[C]//Proceedings of the 2014

International Joint Conference on Neural Networks (IJCNN). Piscataway: IEEE Press, 2014: 4062-4069.
[19] HWANG K, SUNG W. Single stream parallelization of generalized LSTM-like RNNs on a GPU[C]//Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE Press, 2015: 1047-1051.
[20] BECK M, PÖPPEL K, SPANRING M, et al. xLSTM: extended long short-term memory[EB]. arXiv preprint, 2024: arXiv: 2405.04517.
[21] SHARMA R K, CASAS M. Wavefront parallelization of recurrent neural networks on multi-core architectures[C]//Proceedings of the 34th ACM International Conference on Supercomputing. New York: ACM, 2020: 1-12.
[22] CHEN Q F, WU J, HUANG F H, et al. Multi-layer LSTM parallel optimization based on hardware and software cooperation[C]//Proceedings of International Conference on Knowledge Science, Engineering and Management. Cham: Springer, 2022: 681-693.
[23] WANG B C, YANG C Y, ZHU R, et al. Analysis of performance and optimization in MindSpore on ascend NPUs[C]//Proceedings of the 2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS). Piscataway: IEEE Press, 2023: 1701-1708.
[24] JIN H, WU W C, SHI X H, et al. TurboDL: improving the CNN training on GPU with fine-grained multi-streaming scheduling[J]. IEEE Transactions on Computers, 2021, 70(4): 552-565.
[25] FATICA M. CUDA toolkit and libraries[C]//Proceedings of the 2008 IEEE Hot Chips 20 Symposium (HCS). Piscataway: IEEE Press, 2008: 1-22.
[26] MAAS A L, DALY R E, PHAM P T, et al. Learning word vectors for sentiment analysis[C]//Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland: The Association for Computational Linguistics, 2011: 142-150.

作者简介

黄为新（2000- ），男，华中科技大学计算机科学与技术学院硕士生，主要研究方向为深度学习系统的优化。

胡伟方（1995- ），男，华中科技大学计算机科学与技术学院博士生，主要研究方向为分布式深度学习系统平台。

曹雪娇（1998– ），女，华中科技大学计算机科学与技术学院硕士生，主要研究方向为边端协同下的模型选择。

石宣化（1978– ），男，博士，华中科技大学计算机科学与技术学院教授，主要研究方向为并行与分布式计算、云计算与大数据处理等。

收稿日期： 2024–05–20

通信作者： 石宣化，xhshi@hust.edu.cn

基金项目： 新一代人工智能国家科技重大专项（No.2020AAA0108501）；湖北省重大攻关项目（JD）（No.2023BAA024）

Foundation Items: National Science and Technology Major Project (No.2020AAA0108501), Major Program(JD) of Hubei Province (No.2023BAA024)