鹏程·盘古：大规模自回归中文预训练语言模型及应用

Pengcheng-PanGu: Large-Scale Autoregressive Pre-Trained Chinese Language Model with Auto-Parallel Computation and Its Application

曾炜/ZENG Wei[1,2]，苏腾/SU Teng[3]，王晖/WANG Hui[1]，
田永鸿/TIAN Yonghong[1,2]，高文/GAO Wen[1]
（1. 鹏城实验室，中国 深圳 518055；
2. 北京大学，中国 北京 100871；
3. 华为技术有限公司，中国 杭州 310052）
(1. Pengcheng Laboratory, Shenzhen 518055, China;
2. Peking University, Beijing 100871, China;
3. Huawei Technologies Co., Ltd., Hangzhou 310052, China)

DOI：10.12142/ZTETJ.202202006
网络出版地址：https://kns.cnki.net/kcms/detail/34.1228.tn.20220408.1733.010.html
网络出版日期：2022-04-12
收稿日期：2022-02-26

摘要：在鹏城云脑Ⅱ上训练了全球首个拥有全开源2 000亿参数的自回归中文预训练语言大模型——鹏程·盘古。鹏程·盘古模型基于1.1 TB高质量中文训练数据，采用全场景人工智能计算框架MindSpore自动并行技术实现了五维并行训练策略，从而可将训练任务高效扩展到4 096个处理器上。对比实验表明，在少样本或零样本情况下，鹏程·盘古模型在多个中文自然语言理解或生成任务上都具有较优的性能。在此基础上，鹏程·盘古模型在大模型压缩、提示微调学习、多任务学习以及持续学习等方面也取得了很好的应用效果。

关键词：大规模预训练语言模型；鹏城云脑Ⅱ；大规模分布式训练；中文理解与生成；提示微调学习

Abstract: The world´s first large-scale autoregressive pre-trained Chinese language model named Pengcheng-PanGu with up to 200 billion parameters is presented. Pengcheng-PanGu is developed under the Pengcheng cloud brain Ⅱ. 1.1 TB high-quality Chinese data from a wide range of domains to pre-train the model are collected. The training parallelism strategy is implemented based on all-scenarios artificial intelligence computing framework MindSpore Auto-parallel, which composes five parallelism dimensions to scale the training task to 4 096 processors efficiently. The experimental results demonstrate the superior capabilities of Pengcheng-PanGu in performing various natural language understanding and natural language generation tasks under few-shot or zero-shot settings. On this basis, Pengcheng-PanGu model has also achieved better application results in large model compression, prompt fine-tuning, multi-task, and continuous learning.

Keywords: large-scale pre-trained language models; Pengcheng cloud brain Ⅱ; large-scale distributed training; Chinese language understanding and generation; tip fine-tuning learning

近年来，有关大规模预训练语言模型（PLM）的研究[1-9]在自然语言处理（NLP）领域取得了巨大的突破。通过自监督方式从大规模语料库中学习文本的上下文表示，预训练语言模型在完成自然语言理解和自然语言生成（NLG）等任务时所表现的性能已达到国际先进水平。A. RADFORD[10]等首次提出基于自回归语言模型（ALM）的预训练模型——GPT。通过在大规模文本数据上进行无监督预训练，并针对不同有监督任务进行微调，GPT模型的性能在各种NLP任务上均获得了显著提升。

2020年，美国OpenAI团队推出GPT系列模型的最新版本GPT-3[11]。其中，最大的GPT-3模型包含1 750亿个参数，能使用570 GB的文本数据进行训练。除了具有高质量的文本生成能力外，在没有进行特定任务微调的情形下，GPT-3模型小样本学习和零样本学习的性能会随着模型参数的增加而稳步提升。有些任务的性能甚至达到了当前最高水平。GPT-3模型的提出是革命性的，它减轻了人们为新任务标记更多示例和再次训练模型的负担，成为模拟人类小样本学习能力的新范式，为探索通用人工智能（AI）开辟了新途径。

目前，GPT-3模型主要是基于英文语料数据训练出来的，且只能通过OpenAI应用程序接口（API）进行有限度访问。为了促进中文预训练语言模型的研究和应用，以鹏城实

基金项目：广东省重点领域研发计划“新一代人工智能”重大专项（2021B0101400002）

验室为首的联合团队在基于昇腾910芯片的E级智能算力平台（鹏城云脑II）上训练了全球首个全开源2 000亿参数的自回归中文预训练语言大模型——鹏程·盘古。

当模型规模超过100亿时，模型越大，模型训练的难度就越高。其中，模型训练面临的技术挑战主要包括以下几个方面：

（1）模型设计。随着模型规模的扩大，训练过程中可能会出现收敛缓慢甚至发散的问题。在前期工作的基础上，鹏程·盘古模型将基于Transformer的ALM作为基础架构，并在Transformer层之上增加了Query层以诱导模型的预期输出。实验证明，该架构具有很好的扩展性，能够有效支持2 000亿参数规模的模型训练。

（2）训练语料库。训练语料对一个强大、可扩展的预训练模型至关重要。一方面，语料的数据量应该足以满足一个预训练大模型的需求；另一方面，语料数据应是高质量和多样性的，以确保PLM的通用性。为了覆盖广泛的中文语料库，鹏城团队从Common Crawl、电子书、百科全书等资源中收集大量数据，并在此基础上，对数据进行多重过滤和清洗，以确保语料数据满足高质量和多样性需求。

（3）分布式训练。2 000亿参数规模的鹏程·盘古模型对内存的需求远远超出了目前普通多机多卡集群。因此，模型需要在大规模AI集群上进行基于模型切分的并行训练。然而，在大规模AI集群上保持高资源利用率的同时，模型很难获得较大的端到端吞吐量。当涉及硬件拓扑结构时，这个问题变得更具挑战性。通过将五维并行与精心设计的并行策略结合起来，鹏城团队在2 048个Ascend 910处理器[12]大集群上，基于昇腾处理器的异构计算架构（CANN）完成了鹏程·盘古模型的高效并行训练。

鹏城团队在1.1 TB高质量中文文本语料库上训练了鹏程·盘古2.6B、鹏程·盘古13B和鹏程·盘古200B 3个模型，并评估了鹏程·盘古2.6B、鹏程·盘古13B两个模型在16个NLP下游任务上的小样本学习能力。实验结果表明，随着模型参数规模的扩大，鹏程·盘古模型在各种下游任务上的性能表现会更优异。

然而，大模型如何赋能实际应用仍然面临很大挑战。例如，当模型太大时，如何通过有效的模型压缩来赋能边端应用场景？如何将应用任务转化为大模型的原始任务，并通过提示微调学习技术来实现NLP模型训练新范式？如何针对新的数据集和任务，在大模型基础上开展持续学习，并构建高效持续演化的大模型生态？针对这些挑战，我们进一步研发了鹏程·盘古增强版模型。该模型在大模型压缩、提示微调学习、多任务学习以及持续学习等方面均表现出很好的效果。

1 模型架构

鹏程·盘古是一个基于海量文本语料库进行预训练得到的大规模ALM。该模型的训练语料绝大部分是中文。该模型会对语料库中所有Token的生成过程进行建模。一个序列中的一个Token的生成取决于它前面的所有Token。假设一个序列$X=\{x_1,x_2,...,x_N\}$由N个Token组成，那么训练目标可以表述为最大化对数似然：

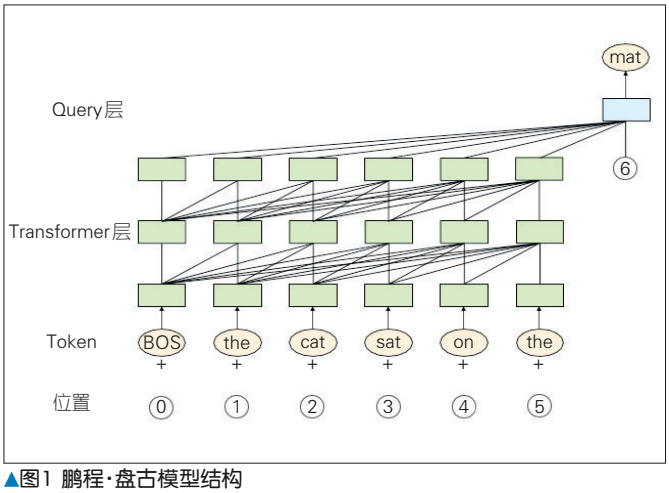
$$\mathcal{L}=\sum_{n=1}^{N}\log p(x_n|x_1,...,x_{n-1};\theta), \tag{1}$$

其中，$p(x_n|x_1,...,x_{n-1};\theta)$是指，在知道前$n-1$个Token $x_{1:n-1}$的情况下，观察到第n个Token x_n的概率；θ表示模型参数。如图1所示，鹏程·盘古保留了Transformer[13]架构，在Transformer层之上还构建了Query层。Query层可用来预测下一个Token。

1.1 Transformer层

（1）多头注意力。第l层的自注意网络由4个投影矩阵组成：$\boldsymbol{W}_h^k$、$\boldsymbol{W}_h^q$、$\boldsymbol{W}_h^v$、$\boldsymbol{W}_h^m$，且它们均属于集合$\mathbb{R}^{d\times d/N_h}$。其中，$d$为隐藏维度，$h$为头的索引，$N_h$为头的数量。根据前一层的输出$H_{l-1}\in\mathbb{R}^{N\times d}$，我们可以计算出当前层的状态。该状态包含3个主要部分：Query $Q_h=H_{l-1}\boldsymbol{W}_h^q$、Key $K_h=H_{l-1}\boldsymbol{W}_h^k$和Value $V_h=H_{l-1}\boldsymbol{W}_h^v$。注意力函数的计算方法为：

$$A_h=Q_hK_h^{\top}=H_{l-1}\boldsymbol{W}_h^q\boldsymbol{W}_h^{k\top}H_{l-1}^{\top},$$
$$\text{Attention}_h(H_{l-1})=$$
$$Softmax(\frac{A_h}{\sqrt{d}})V_h=Softmax(\frac{A_h}{\sqrt{d}})H_{l-1}\boldsymbol{W}_h^v。 \tag{2}$$



▲图1 鹏程·盘古模型结构

在计算完包含多个头的注意力后，输出就可以变成：

$$\begin{aligned}\mathrm{MHA}(H_{l-1}) &= \sum_{h=1}^{N_h}\mathrm{Attention}_h(H_{l-1})\boldsymbol{W}_h^m,\\ H_l^{\mathrm{MHA}} &= H_{l-1}+\mathrm{MHA}(\mathrm{LayerNorm}(H_{l-1}))。\end{aligned}\tag{3}$$

（2）前馈神经网络（FFN）。FFN由两个线性层组成，相关参数为$W^1\in\mathbb{R}^{d\times d_{ff}}$、$b^1\in\mathbb{R}^{d_{ff}}$、$W^2\in\mathbb{R}^{d_{ff}\times d}$、$b^2\in\mathbb{R}^d$，其中$d_{ff}$是内层纬度。如果将多头注意力层（MHA）输出作为输入，那么FFN层的输出为：

$$\begin{aligned}\mathrm{FFN}(H_l^{\mathrm{MHA}}) &= \mathrm{GeLU}(H_l^{\mathrm{MHA}}W^1+b^1)W^2+b^2,\\ H_l &= H_l^{\mathrm{MHA}}+\mathrm{FFN}\left(\mathrm{LayerNorm}\left(H_l^{\mathrm{MHA}}\right)\right)。\end{aligned}\tag{4}$$

对于MHA和FFN，本文采取了Pre-layer Normalization方案。该方案可以使Transformer模型训练变得更加简单、高效[14]。

1.2 Query层

模型在Transformer层之上堆叠了一个Query层，目的是输出一个明确的引导。在ALM的预训练阶段，Query层可被用来预测下一个Token。Query层的结构与Transformer层类似。在计算注意力机制的时候，Query层会对表示下一个位置的位置嵌入$p_n\in\mathbb{R}^d$做Query向量处理。具体来说，假设H_L是最上层Transformer层的输出，则Query层的注意力向量可以表示为：

$$a_h=p_n\boldsymbol{W}_h^q\boldsymbol{W}_h^{k\top}H_L^{\top}。\tag{5}$$

随后，MHA和FFN的计算方式仍与原始Transformer相同。如果把最终的输出表示为o_n，则下一个Token的负对数似然就可以写为：

$$CrossEntropy(x_n,\ Softmax(o_nW^o+b^o))\ ,\tag{6}$$

其中，x_n表示真实Token，W^o、b^o是任务相关的额外参数。

1.3 模型配置

为了评估鹏程·盘古模型的扩展能力，本文训练了3个参数不断增加的模型，即鹏程·盘古2.6B、鹏程·盘古13B和鹏程·盘古200B。表1展示了这3个模型的详细配置，包括参数总数量、Token的隐藏维度、前馈层的内层维度和注意力的头数。

2 数据集

超大规模高质量中文语料数据集对训练千亿级参数规模的鹏程·盘古模型至关重要。目前已有3个100 GB以上规模的中文语料数据集，它们分别是：（1）从Common Crawl抽取得到的CLUECorpus2020[15]，该模型的数据量为100 GB；（2）阿里巴巴集团发布的M6[16]中文多模态模型，该模型使用300 GB语料；（3）北京智源研究院面向合作者发布的包含300 GB高质量中文语料WuDaoCorpus。然而，与目前同等规模参数量的英文预训练模所使用的数据量相比，上面这些中文语料数据仍然不能满足2 000亿中文预训练语言模型的训练数据需求。

尽管像SogouT和Common Crawl等原始网页数据已经包含大量的中文语料数据，但是构建一个可满足2 000亿参数模型训练需求的大规模语料数据集仍需要解决诸多问题。这些问题包括：（1）原始网页数据质量参差不齐，语料预处理流程繁琐复杂；（2）海量原始语料数据处理缺少大规模存储和计算能力的支撑；（3）缺乏一个有效准确的数据质量评估方法。

为解决上述问题，我们搭建了一个大规模中文语料数据处理平台，以提升海量数据采集、清洗、过滤等处理效率，并以此构建了一个1.1 TB的高质量中文语料数据集。在数据集的构建过程中，我们采用人工评估与模型评估相结合的方法为数据集的清洗、过滤以及训练数据集的选择提供指导。

2.1 数据集构建

为了构建一个大规模高质量中文语料数据集，我们收集了包含开放数据集、Common Crawl原始网页数据、百科数据、新闻数据、电子书籍等近80 TB的原始数据。如图2所示，数据集构建流程包括3个主要步骤：基于规则的数据清洗、基于模型的过滤、数据去重。我们通过人工和模型分别对数据质量进行评估，并且通过不断迭代前两个步骤来提升数据质量。整个数据集的构建过程是基于Spark/Hadoop搭建的大数据处理平台完成的。该平台使数据处理效率得到了明显提升。

2.1.1 数据清洗和过滤

在图2所示的5种数据来源中，Common Crawl的数据占

▼表1 鹏程·盘古模型的规模和参数

模型	参数总数量/亿	层数	内层维度	FFN大小	头数
鹏程·盘古 2.6B	26	32	2 560	10 240	32
鹏程·盘古 13B	131	40	5 120	20 480	40
鹏程·盘古 200B	2 070	64	16 384	65 536	128

FFN：前馈网络

比虽然最大，但是它包含了大量低质量的网页数据。因此，我们首先采用如下规则对 Common Crawl 的原始数据进行清洗。

• 去除中文字符低于 60% 或者字符数小于 150 的数据（仅有网页名称的数据也会被去除）；

• 去除特殊字符，并去除在一个网页中重复出现的段落；

• 通过广告关键词去除包含大量广告的网页数据；

• 将所有繁体中文转换为简体中文；

• 识别并去除网页导航页。

在完成原始数据清洗后，我们采用 3 个过滤器来进一步过滤数据中的敏感词、广告、低质量段落等信息。

• 关键词过滤。构建一个包含 724 个敏感词的词库，并通过敏感词库去除包含 3 个以上敏感词的网页数据。

• 基于模型的过滤。为了进一步去除垃圾广告和垃圾邮件数据，我们通过人工标注数据来训练一个 FastText 文本分类模型。负样本为从 Common Crawl 数据中人工挑选的 1 万条垃圾文本数据，正样本为从高质量中文语料数据中抽样得到的数据。基于 FastText 的文本分类模型可以对语料进行垃圾过滤处理。

• 低质量文本过滤。借鉴 GPT-3 的数据处理策略，我们训练了一个数据质量评分模型。该模型可去除得分较低的文本（详见 GPT-3 附录-A[1]）。

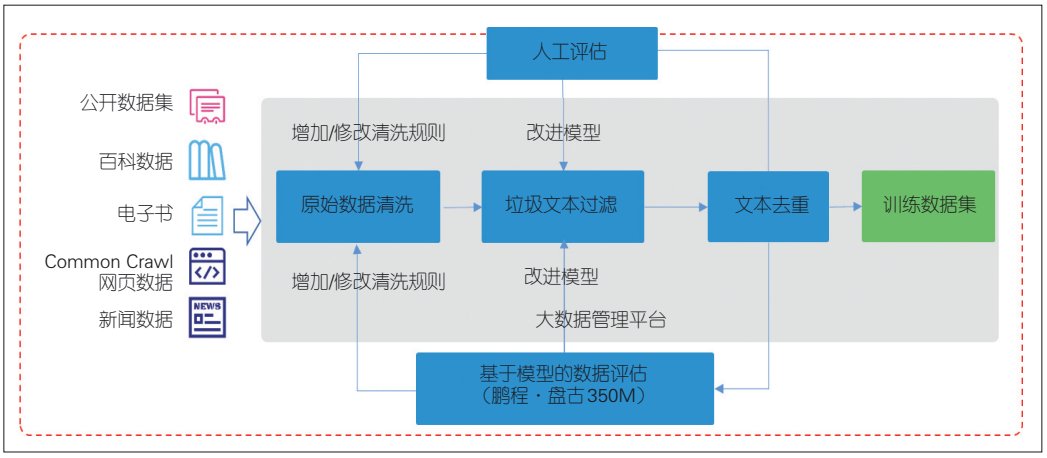
2.1.2 文本去重

由于全量数据太大，基于 Spark 的 MinHashLSH 算法在对 200 MB 数据进行去重时需要消耗至少 8 h 的时间，去重效率较低。为了加速文本数据去重过程，我们设计了一种分布式海量文本数据重复检测和去重算法。针对 500 GB 语料数据的去重任务，该算法能够将原本 20 000 h 的时间缩短至 3.5 h，极大地提升了去重效率。

2.1.3 数据质量评估

数据清洗过程中的一个重要问题是如何确定清洗规则或数据过滤阈值。对此，我们提出人工和模型相结合的数据质量评估方法，在每一轮数据清洗和过滤后，对数据清洗和过滤的效果进行评估，并通过清洗过滤、质量评估的多轮迭代来提升数据质量。其中，人工评估数据原则主要从句子通顺性、文本低质量内容占比（如广告短语、重复短句、敏感词等）两个维度进行评估。

人工评估虽然有效，但是对于大规模语料数据来说，能够评估的语料占比太小，不能充分反应数据集的整体质量。为了提高数据评估准确度，我们从待评估全量数据中抽取 30 GB 的数据来训练鹏程·盘古 350M 模型，并通过该模型在高质量数据集中的困惑度（PPL）指标来评估数据质量。模型在高质量数据中的 PPL 越小，数据集清洗和过滤所采用的清洗规则和过滤模型就越好。



▲图 2 鹏程·盘古模型的训练数据处理流程

2.2 数据采样策略

通过图 2 中的数据处理流程，我们从 5 种来源的近 80 TB 原始数据中清洗并构建了一个 1.1 TB 的高质量中文语料数据集。数据集组成和数据处理方法如表 2 所示。基于由上述步骤构建的语料数据，采样形成的两个训练数据集被用于训练鹏程·盘古 2.6B、鹏程·

▼表 2 1.1 TB 中文语料数据组成

数据来源	大小/GB	数据源	数据处理步骤
开放数据集	27.9	15 个开放数据集，如 DuReader、BaiDuQA、CAIL2018、Sogou-CA 等	数据格式转换、文本去重
百科数据	22.0	百度百科、搜狗百科等百科类数据	文本去重
电子书籍	299.0	不同主题的电子书籍，如小说、历史、诗歌、古文等	敏感词过滤、基于模型的文本过滤
Common Crawl	714.9	2018 年 1 月—2020 年 12 月的 Common Crawl 网页数据	数据清洗、过滤、去重等所有数据处理步骤
新闻数据	35.5	1992—2011 年的新闻数据	文本去重

盘古 13B 和鹏程·盘古 200B 模型。两个训练数据集的数据量分别是 100 GB 和 1 TB。如表 3 所示，训练数据集对每个数据源的数据进行采样。采样比例和重复次数越大，数据源的质量就越好。在两个训练集的词 Token 数量分布方面，100 GB 训练集和 1 TB 训练集的平均段落长度分别是 239、405 个 Token。可以看出，1 TB 的平均段落长度更长。这是因为 1 TB 训练集中 Common Crawl 数据的占比更大。值得注意的是，训练数据段落长短与模型生成效果有关。当训练样本平均长度较短时，模型倾向于生成更短的句子，从而有利于模型处理下游任务中需要生成短句的任务；反之，当训练样本平均长度较长时，模型会倾向于生成更长的句子。

3 并行训练系统

鹏程·盘古 200B 的模型训练将面临巨大挑战。比如，鹏程·盘古 200B 的内存存储需求就高达 750 GB。由于梯度和优化器状态对参数更新也很重要，因此训练如此庞大的模型所消耗的内存会比参数存储要高好几倍。相比之下，现代 AI 处理器（如图形处理器、Ascend 910 AI 处理器[12]）的内存约为 30 ~ 40 GB。因此，将模型切分到设备（处理器）的集群中是不可避免的。为此，我们需要应对两个方面的技术挑战。首先，多个不同的并行功能应该结合起来，以使模型获得较高的端到端性能。然而，由于策略空间巨大，寻找最佳的策略组合是一个挑战。其次，并行训练应满足易用性与高效性的双重需求，底层与并行相关的处理逻辑应该与模型定义的处理逻辑相解耦。

在基于昇腾 910 芯片的 E 级智能算力平台（鹏城云脑 II）上，我们使用 MindSpore 自动并行技术来应对上述两个方面的挑战，从而最大限度地提高计算通信比。该自动并行技术支持五维度的并行能力，并使用拓扑感知调度将切片的模型映射到集群上，以获得较高的端到端性能。此外，该自动并行技术只需要对单机代码进行最少的代码修改，就可以实现快捷高效的超大模型并行训练。

（1） 五维并行和拓扑感知调度

最常用的并行方式是数据并行，它在设备之间划分训练的批次大小，并在执行迭代优化命令之前与来自不同设备的梯度信息保持同步，如图 3（a）所示。模型并行有 3 种方式。第 1 种是算子级并行[17-23]，它对每个算子所涉及的张量进行切分。如图 3（b）所示，算子级并行通过对参数和显存进行切片来减少显存消耗，同时通过通信优化来使连续算子之间的分布式张量状态保持一致。第 2 种是流水并行[24-28]，它将总的模型层划分为不同阶段，然后将不同阶段的模型层放置到不同的设备上，如图 3（c）所示。每台设备只拥有模型层次的一部分，可大大节省显存占用，并使通信只发生在不同状态的边界上。第 3 种机制是优化器并行[29]，其作用是减少由数据并行所导致的优化器内存冗余和计算消耗。图 3（d）中前向运算阶段的一些中间结果要在显存中驻留相当长的时间，以加速后向阶段的梯度计算。如图 3（e）所示，重计算前向运算结果可以释放部分中间结果，以减少整个训练阶段显存消耗。需要指出的是，每个维度的并行都要通过计算（或通信）开销来换取显存（或吞吐量）收益。因此，为了获得最大的端到端吞吐量，我们需要在多维度并行之间找到一个最佳组合平衡点。而设备集群中的异构带宽使这变得更具挑战性。

（2） 混合并行训练

图 4 展示了鹏程·盘古 200B 模型的混合并行方案。首先，将模型总层次（64 层）划分成 16 个状态，每个状态包含 4 层。每一层会为每个算子切分所需要的参数和张量。具体来说，Query(Q)、Key(K) 和 Value(V)算子相关的参数被切分为 8 片。我们将这 3 个算子的输入张量划分为 16 个切片，并以此确定优化器并行的维度。该层中其他算子的并行策略也以同样的方式进行配置。每层算子都首先被切分，然后再执行下发命令。这有效降低了额外的计算开销。在本方案中，我们总共使用了 2 048 个来自鹏城云脑 II 的 Ascend 910 AI 处理器。

鹏程·盘古 200B 模型具体的混合并行策略为：数据并行 8 路、算子级并行 8 路、流水并行 16 路，在数据并行的同时叠加优化器并行。模型会将通信量大的并行方式（算子级并行）放置在服务器内部的多卡之间，将通信量较小的并行

▼表 3 鹏程·盘古模型训练数据采样策略

数据来源	鹏程·盘古 200B			鹏程·盘古 2.6B 和鹏程·盘古 13B	
	Token 数量/亿	数据占比/%	训练完成时重复次数	Token 数量/亿	数据占比/%
开放数据集	258	10.23	3.65	70	27.99
电子书籍	309	12.23	0.41	56	18.00
Common Crawl	1 762	62.81	0.85	25	10.00
新闻数据	198	7.83	2.20	56	22.00
百科数据	58	6.90	3.00	58	23.00

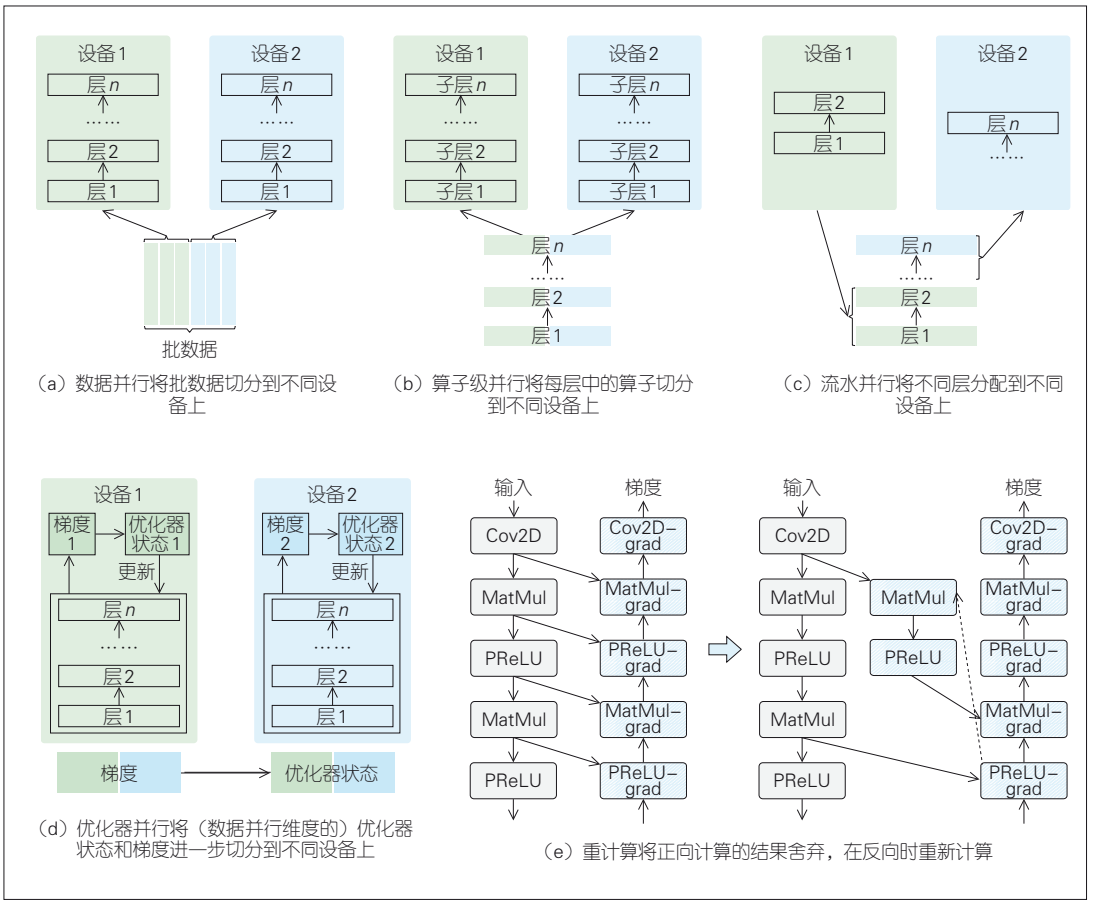
方式（流水并行）放置在同一机架内的服务器之间，将部分数据并行（叠加优化器并行）放置在不同机架之间。因此，通信可以与计算同时进行，对带宽要求较低。

4 实验

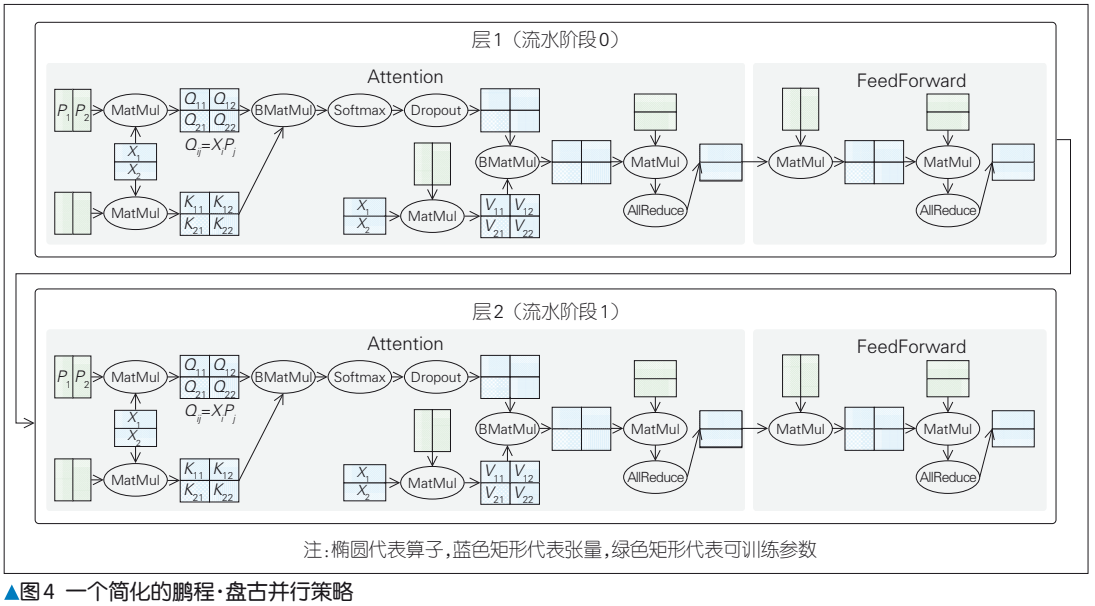
4.1 训练细节

鹏程·盘古模型是基于华为Mindspore框架开发的，它采用由2 048块Ascend910 AI处理器组成的集群进行训练，并最终扩展到全机4 096块Ascend910 AI处理器集群上。模型的详细配置如表4所示。在训练程·盘古200B模型时，我们首先采用2 048块处理器，然后将其切换到1 024块上继续进行训练。实验将字节对编码（BPE）作为分词器，词表的规模为40 000，并且所有模型均采用1 024的序列长度。

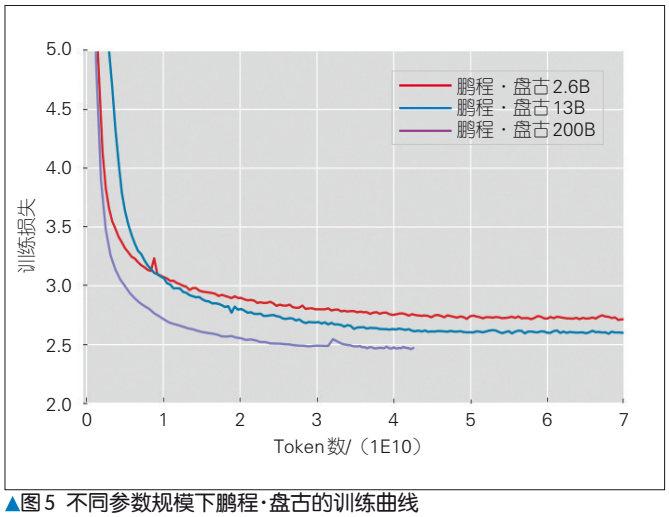
鹏程·盘古模型的训练损失曲线如图5所示。因为鹏程·盘古200B、鹏程·盘古13B和鹏程·盘古2.6B模型训练的批量大小不同，所以我们用Token数作为X轴。由图5可以看出，鹏程·盘古200B、鹏程·盘古13B和鹏程·盘古2.6B的模型训练损失分别收敛在2.49、2.58和2.64，并且在训练结束时训练损失仍然在下降。这表明模型精度仍有提升的潜力。本文在验证集上评估了模型的PPL性能。其中，验证集是从Common Crawl数据集中随机抽取的。评估结果表明，模型越大，PPL就越低，模型



▲图3 5种并行方式及其优化显存和吞吐量的过程



▲图4 一个简化的鹏程·盘古并行策略



▲图5 不同参数规模下鹏程·盘古的训练曲线

▼表4 鹏程·盘古的详细训练配置

模型	训练步长	处理器个数	Adam Betas	学习率	权重衰减
鹏程·盘古2.6B	0~70 000	512	β_1=0.9 β_2=0.999	1E−4	0.01
鹏程·盘古13B	0~84 000	1 024	β_1=0.9 β_2=0.980	5E−5	0.01
鹏程·盘古200B	0~130 000 130 000~260 000	2 048 1 024	β_1=0.9 β_2=0.950	2E−5	0.10

性能也就越优。

4.2 任务描述

本文在多种自然语言处理下游任务的基础上来评估模型的性能。与GPT-3[1]类似，实验采用3种不经任务微调的配置：零样本学习、单样本学习和小样本学习。如果能获取到测试集，每个下游任务就会在测试集上进行评估。参与的16个下游任务包含7个类别：完形填空与补全、阅读理解、闭卷问答、指代消解、常识推理、自然语言推理、文本分类。

4.3 评估细节

由于测试方式不同，本文将所有任务分为两大类：生成类任务和分类任务。

（1）生成类任务的评测方法

生成类任务包含词级别和句子级别的生成任务。鹏程·盘古模型天然具备强大的文本生成能力，能够采用模型自然生成文本的方式生成此类任务的答案。对于中文上下文词语预测数据集（WPLC）、中文填空型阅读理解（PD&CFT）和阅读理解评测（CMRC2017）这类完形填空与补全任务，上下文可作为提示被放置在待预测位置的前面。而对于阅读理解和闭卷问答任务，模型则能够根据需要设计相应的提示模板。例如，阅读理解任务可将样本填充到“*Reading document: $Document Question: $Question Answer:*”模板中，并将其作为提示输入到模型中。类似于GPT-3，小样本学习采用上下文学习的方式，即把K个提示相互拼接。其中，前K−1个提示均包含答案，最后一个提示的答案则通过模型预测来获得。

（2）分类任务的评测方法

分类任务主要采用基于PPL的评测方法。针对每组<段落，标签>数据对，该方法会根据预设计模板自动生成输入。由模板生成的序列将被输入到模型中，同时模型将计算出相应的PPL值。具有最小PPL值的标签将被作为该段落的预测结果。与生成类任务评测类似，分类任务也采用上下文学习策略来完成小样本学习任务。

4.4 实验结果

本文对比了鹏程·盘古2.6B模型和CPM2.6B模型[3]在16个中文下游任务上的表现。鹏程·盘古2.6B模型在11个零样本学习任务、12个单样本学习任务、14个小样本学习任务上的表现均超越CPM 2.6B模型。实验结果表明，相比于CPM2.6B模型，鹏程·盘古2.6B模型具有更强的上下文学习能力（尤其在小样本学习和生成方面）。在生成任务方面，鹏程·盘古2.6B模型要比CPM2.6B模型平均高出6个百分点。具体地，在阅读理解任务和闭卷问答任务上，鹏程·盘古2.6B模型比CPM2.6B模型高出5个百分点；在无选项完形填空任务上，鹏程·盘古2.6B模型比CPM2.6B模型高出7个百分点。在PPL任务方面，鹏程·盘古2.6B模型与CPM2.6B模型相当，而在TNEWS和IFLYTEK分类任务上的表现则不如CPM2.6B模型。造成这种现象的主要原因是CPM2.6B模型和鹏程·盘古2.6B模型的训练语料具有差异性。

我们同时对比了鹏程·盘古13B和鹏程·盘古2.6B在16个中文NLP下游任务上的表现。鹏程·盘古13B在所有生成式任务和绝大多数PPL任务上的表现，均明显优于鹏程·盘古2.6B模型。在CMRC2018、DRCD和WebQA任务上，鹏程·盘古13B小样本学习的性能比零样本学习高10个百分点。这说明鹏程·盘古13B模型具有极强的上下文学习能力。鹏程·盘古13B在16个下游任务中的表现比鹏程·盘古2.6B高出近3个百分点。具体地，鹏程·盘古13B模型在阅读理解和闭卷问答任务上的表现比鹏程·盘古2.6B模型高出近4个百分点，在无选项完形填空任务上的表现比鹏程·盘古2.6B高出近2个百分点。在自然语言推理（NLI）任务上，鹏程·盘古13B模型的表现则不如鹏程·盘古

2.6B，这与GPT-3实验结果是一致的。总之，鹏程·盘古13B模型和鹏程·盘古2.6B模型的对比实验表明：更大规模的预训练模型的性能通常能在小样本学习任务上取得提升。

5 大模型应用

5.1 模型压缩

虽然鹏程·盘古模型具备强大的能力，但超大规模的模型参数量却限制了它的应用。我们通常希望应用端能够在保持几乎同等性能的条件下就可得到较小参数规模的模型，以便提升应用效率。因此，我们研究了鹏程·盘古的模型压缩技术，并采用量化与参数共享的方法实现了鹏程·盘古13B模型和鹏程·盘古2.6B模型在单张Ascend 910卡上的应用。其中，量化是指借助低精度类型加载模型，用FP16代替大部分FP32类型参数，同时对量化噪声和数值溢出进行处理；参数共享是指将部分层的参数进行共享，例如将输出层参数与嵌入层参数进行共享。这种压缩技术使显存占用降低50%，系统性能波动仅为2%左右。为了评估压缩技术对模型性能的影响，实验测试了部分下游任务在压缩前后的性能指标。结果表明，在闭卷问答任务（WebQA）上，压缩后鹏程·盘古13B模型的F1值比压缩前小0.01；在代词消歧任务（CLUEWSC2020）中，压缩后鹏程·盘古13B模型的精度仅比压缩前下降1个百分点。

5.2 模型移植

为了便于更多用户使用鹏程·盘古模型，我们将该模型从Mindspore框架成功移植到PyTorch框架下。移植流程主要包括3个步骤。

第1步：在PyTorch框架上复现鹏程·盘古。复现工作是基于开源分布式Transformer的Megatron框架实现的，即在Transformer的decoder解码结构上加一层Query层。

第2步：手动转换模型文件。由于鹏程·盘古的部分算子不能转成开放神经网络交换（ONNX）通用模型格式，所以我们需要对模型文件进行手动转换。手动转换模型文件的流程包括：（1）提取Mindspore模型中的参数数据和参数名称，并将提取的参数保存为数组数据；（2）手动对齐Mindspore和PyTorch模型的参数名称和参数维度，并将其保存为PyTorch模型文件类型。

第3步：对齐PyTorch实现版本和Mindspore实现版本的并行策略。在进行分布式训练时，我们需要把隐藏态分割到不同的设备上。然而，基于两个不同框架实现的分割方案存在一定的差异。图6展示了Mindspore实现版本和PyTorch实现版本的模型切割策略。可以看出，左边Mindspore的切割策略是直接从中间将隐藏态分成两份，然后为每个设备分配一份；而右边PyTorch的切割策略则是先将隐藏态分成3份，然后再把每份数据平均分配到不同的设备上。为了保证最后输出结果的一致性，我们需要手动调整移植后模型文件的权重。

目前，移植到PyTorch框架后的鹏程·盘古代码和模型文件已经在OpenI社区开源共享。

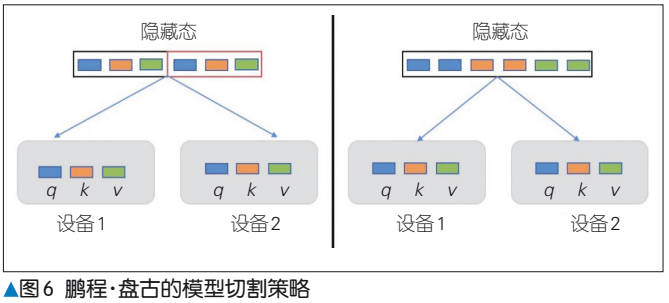
5.3 基于增量推理的在线体验服务加速

为了让更多的用户体验鹏程·盘古模型的强大功能，我们设计并开放了在线体验服务，目前已处理上万条用户请求。鹏程·盘古模型一次完整的在线推理可以包含多个Tokens的生成。模型需要根据上文输入来预测下一个Token，然后将预测的Token追加至输入内容结尾，以便让模型继续生成下一个Token。

通常，在对输入序列Pad补到固定长度（如1 024）后再让模型进行自回归生成的方式，会明显引入冗余计算。这将极大降低模型的推理能力。对此，我们采用状态复用的改进算法（增量推理），来提高模型的推理能力。对于不同步的输入，前部分序列的内容完全相同。当计算索引为i的位置时，前$0 \sim i-1$位置对应的状态在上一步中已进行计算。因此，在推理过程中，第i步可以复用第$i-1$步的状态，并将其和当前推理得到的状态进行拼接，以便作为第i步的完整状态。系统在得到第i步输出的Token时，即可省掉这些重复计算。这将极大提升模型的推理能力。测试结果表明，增量推理可使模型性能提升5倍以上（评估的方法是：输入一段话，预测下一个词的平均输出时间）。基于增量推理的在线体验服务网址为https://pangu-alpha.openi.org.cn/。

5.4 鹏程·盘古增强版

鹏程·盘古增强版能针对多个下游任务进行持续的提示微调训练，其主要创新包括：



▲图6 鹏程·盘古的模型切割策略

• 创新应用多任务学习、任务统一格式、提示微调和持续学习技术，对基本版模型进行能力扩展和增强，使模型性能得到大幅提升；

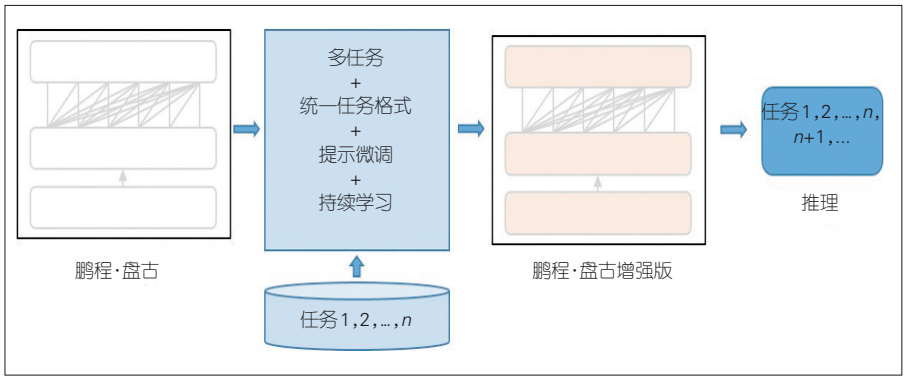
• 形成基于鹏程·盘古模型进行提示微调和持续学习的应用新范式，更好地识别用户的任务说明，同时能尽量保持模型的原始生成能力；

• 参数量为26亿规模，在中英文翻译、开放域知识问答、文本分类、摘要生成等方面的能力提升显著，在一张V100 GPU卡上就可以完成多路并行推理。

（1）技术方案

图7显示了构建鹏程·盘古增强版模型的技术方案。该方案采用多任务、统一任务格式、提升微调和持续学习等技术方法来增强模型的任务处理能力，同时提升对任务描述的泛化能力。

（2）统一任务格式



▲图7 鹏程·盘古增强版技术原理

我们设计了统一的任务数据格式。该统一格式旨在减少下游任务之间的差异，提高知识转移和任务描述的泛化能力。借助统一格式，我们在18个任务上构建了50多个提示，并通过提示微调技术来训练盘古增强版模型。统一任务格式的具体细节可从https://git.openi.org.cn/PCL-Platform.Intelligence/PanGu-Alpha-Evolution网页中查询了解。

（3）实验性能

我们开展了大量实验来比较鹏程·盘古增强版和鹏程·盘古基本版在自然语言理解任务、自然语言生成任务中的性能。对于每个任务来说，如果能获取到测试集则在测试集上进行评估，否则就在验证集上进行评估。为了降低计算资源消耗，部分任务会从数据中随机采样部分子集来进行评估。性能对比结果如表5，表中“Δ”是指鹏程·盘古增强版相对鹏程·盘古基本版提升的绝对值，“相对提升”是指鹏程·盘古增强版相对鹏程·盘古基本版提升的百分比。因为鹏程·盘古不具备翻译能力，所以我们不计算这方面的相对提升百分比。结果表明，鹏程·盘古增强版在各项任务上的表现均远远优于鹏程·盘古基本版（平均相对提升高达1 064.70%）。人工评估表明，鹏程·盘古增强版具有与鹏程·盘古基本版相同的文本生成能力。

6 结束语

本文详细介绍了大规模自回归中文

▼表5 鹏程·盘古增强版优越的性能

序号	任务	指标	鹏程·盘古2.6B基本版	鹏程·盘古2.6B增强版	差值（绝对值）	相对提升/%
1	CMRC2018	Em	1.21	64.00	62.79	5 189.26
2	DRCD	Em	0.80	66.00	65.20	8 150.00
3	Dureader	Rouge-L	21.07	65.57	44.50	211.20
4	CHID	Acc.	68.73	74.00	5.27	7.67
5	PD&CFT	Acc.	38.47	88.00	49.53	128.75
6	CMRC2017	Acc.	37.83	96.00	58.17	153.77
7	CMNLI	Acc.	50.20	80.00	29.80	59.36
8	TNEWS	Acc.	60.95	88.00	27.05	44.38
9	AFQMC	Acc.	59.29	66.00	6.71	11.32
10	CSL	Acc.	50.50	52.00	1.50	2.97
11	WebQA.v1.0	Em	6.00	48.00	42.00	700.00
12	CLUEWSC2020	Acc.	73.36	84.00	10.64	14.50
14	C3	Acc.	53.42	66.00	12.58	23.55
15	LCSTS	Rouge-L	11.21	34.65	23.44	209.04
16	Wmt20 enzh	Bleu-4	0	9.84	9.84	
17	Wmt20 zhen	Blen-4	0	18.52	18.52	

预训练语言模型鹏程·盘古，并探索了该模型的具体应用。大规模语言模型虽然是当前的研究热点，但仍存在很多开放性的问题。

（1）大规模语言模型在NLP任务上表现出较好的小样本学习能力，但目前对大规模语言模型的系统性研究仍然比较缺乏。如何训练出大规模PLM并使模型生成的文本更加规范安全、更加鲁棒、更加符合常识（或知识）仍是最具挑战性的问题。

（2）超大规模语言模型的训练、推理和维护成本非常高。如何高效地训练出一个大模型并使模型具有持续演化能力？大规模PLM的绿色生态、持续学习演化等是一个有趣的探索方向。

（3）大规模语言模型被认为是通向通用人工智能的重要途径。它的自监督预训练模式、小样本学习能力以及单模型多任务的适配能力都具有很好的应用前景。由于目前仍缺乏兼具逻辑推理、常识和认知能力的大模型，人们只能使用巨量参数来拟合并训练语料中长尾分布的记忆“巨兽”。如何大模型它具备人类推理、思考和认知能力仍然任重而道远。

致谢

感谢鹏城实验室提供鹏城云脑支撑本文研究。感谢鹏城实验室的王晖、张艳、颜达森、蒋芳清、易泽轩、陶恒韬、王进，华为公司的苏腾、任晓哲、廖亿、蒋欣、王志伟等为本研究做了大量工作。

参考文献

[1] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners. [EB/OL]. [2022-02-25]. https://arxiv.org/abs/2005.14165
[2] DEVLIN J, CHANG M, LEE K, et al. BERT: pre-training of deep bidirectional transformers for language understanding [EB/OL]. [2022-02-25]. https://arxiv.org/abs/1810.04805
[3] ZHANG Z Y, HAN X, ZHOU H, et al. CPM: a large-scale generative Chinese pre-trained language model [J]. AI open, 2021, 2: 93-99. DOI: 10.1016/J.AIOPEN.2021.07.001
[4] YANG Z, DAI Z, YANG Y, et al. XLNet: generalized autoregressive pretraining for language understanding. [EB/OL]. [2022-02-25]. https://paperswithcode.com/paper/xlnet-generalized-autoregressive-pretraining
[5] LIU Y H, OTT M, GOYAL N, et al. RoBERTa: a robustly optimized BERT pretraining approach [EB/OL]. [2022-02-25]. https://arxiv. org/abs/1907.11692
[6] RAFFEL C, SHAZEER N, ROBERTS A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer [EB/OL]. [2022-02-25]. https://arxiv.org/abs/1910.10683
[7] SUN Y, WANG S H, LI Y K, et al. ERNIE: enhanced representation through knowledge integration [EB/OL]. [2022-02-25]. https://arxiv. org/abs/1904.09223
[8] ZHANG Z, HAN X, LIU Z, et al. ERNIE: enhanced language representation with informative entities. [EB/OL]. [2022-02-25]. https://arxiv. org/abs/1905.07129
[9] WEI J Q, REN X Z, LI X G, et al. NEZHA: neural contextualized representation for Chinese language understanding [EB/OL]. [2022-02-25]. https://arxiv.org/abs/1909.00204
[10] RADFORD A, NARASIMHAN K. Improving language understanding by generative pre-training [EB/OL]. [2022-02-25]. https://paperswithcode.com/paper/improving-language-understanding-by
[11] RADFORD A, WU J, CHILD R, et al. Language models are unsupervised multitask learners [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/language-models-are-unsupervised-multitask
[12] LIAO H, TU J J, XIA J, et al. DaVinci: a scalable architecture for neural network computing [C]//Proceedings of 2019 IEEE Hot Chips 31 Symposium. IEEE, 2019: 1-44. DOI: 10.1109/HOTCHIPS.2019.8875654
[13] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need. [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/attention-is-all-you-need.
[14] XIONG R, YANG Y, HE D, et al. On layer normalization in the transformer architecture [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/on-layer-normalization-in-the-transformer-1
[15] XU L, ZHANG X W, DONG Q Q. CLUECorpus2020: a large-scale Chinese corpus for pre-training language model [EB/OL]. [2022-02-25]. https://arxiv.org/abs/2003.01355
[16] LIN J, MEN R, ZHOU C, et al. M6: A chinese multimodal pretrainer [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/m6-a-chinese-multimodal-pretrainer
[17] SHAZEER N, CHENG Y L, PARMAR N, et al. Mesh-TensorFlow: deep learning for supercomputers [EB/OL]. [2022-02-25]. https://paperswithcode.com/paper/mesh-tensorflow-deep-learning-for
[18] JIA Z H, LIN S N, QI C R, et al. Exploring hidden dimensions in parallelizing convolutional neural networks [EB/OL]. [2022-02-25]. https://paperswithcode.com/paper/exploring-hidden-dimensions-in-accelerating
[19] WANG M J, HUANG C C, LI J Y. Supporting very large models using automatic dataflow graph partitioning [C]//Proceedings of the Fourteenth EuroSys Conference 2019. ACM, 2019. DOI: 10.1145/3302424.3303953
[20] LEPIKHIN D, LEE H, XU Y Z, et al. GShard: scaling giant models with conditional computation and automatic sharding [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/gshard-scaling-giant-models-with-conditional
[21] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: training multi-billion parameter language models using GPU model parallelism [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/megatron-lm-training-multi-billion-parameter
[22] SONG L H, CHEN F, ZHUO Y W, et al. AccPar: tensor partitioning for heterogeneous deep learning accelerators [C]//2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2020
[23] JIA Z H, ZAHARIA M, AIKEN A. Beyond data and model parallelism for deep neural networks [EB/OL]. [2022-02-25]. https://arxiv. org/abs/1807.05358
[24] HUANG Y P, CHENG Y L, BAPNA A, et al. TensorPipe: easy scaling with micro-batch pipeline parallelism [EB/OL]. [2022-02-25]. https://paperswithcode.com/paper/gpipe-efficient-training-of-giant-neural
[25] NARAYANAN D, HARLAP A, PHANISHAYEE A, et al. PipeDream: generalized pipeline parallelism for DNN training [C]//Proceedings of the 27th ACM SOSP. ACM, 2019: 1-15
[26] FAN S Q, RONG Y, MENG C, et al. DAPPLE: a pipelined data parallel approach for training large models [EB/OL]. [2022-02-25]. https://cs.paperswithcode.com/paper/dapple-a-pipelined-data-parallel-approach-for
[27] TARNAWSKI J, PHANISHAYEE A, DEVANUR N R, et al. Efficient algorithms for device placement of DNN graph operators [EB/OL]. [2022-02-25]. https://paperswithcode. com/paper/efficient-algorithms-for-device-placement-of
[28] PARK J H, YUN G, YI C M, et al. HetPipe: enabling large DNN training on (whimpy) heterogeneous GPU clusters through integration of pipelined model parallelism and data parallelism [EB/OL]. [2022-02-25]. https://arxiv.org/abs/2005.14038
[29] RAJBHANDARI S, RASLEY J, RUWASE O, et al. ZeRO: memory optimizations toward training trillion parameter models [C]//Proceedings of SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020. DOI: 10.1109/sc41405.2020.00024

作 者 简 介

曾炜，北京大学视频与视觉技术国家工程研究中心副研究员；长期从事人工智能、智能计算系统、图像处理、计算机视觉、模式识别、多媒体领域的研究工作；曾获10余项国家和省部级科研项目资金资助；发表论文50余篇，申请专利20余项。

苏腾，华为技术有限公司分布式并行计算技术专家、MindSpore副首席专家；长期从事分布式并行计算系统的设计与开发工作；主导华为分布式任务计算框架、分布式强一致性K-V存储框架、容器化资源管理与调度框架的设计与开发，在大规模分布式系统方面拥有丰富的实践经验。

王晖，鹏城实验室网络智能部开源所研究员，曾为国防科技大学系统工程学院信息系统工程国家重点实验室教授、博士生导师；在NLP大模型、分布式机器学习、联邦学习等领域开展关键技术及应用研究工作；获得军队科技进步奖一等奖2项、二等奖3项。

田永鸿（通信作者），北京大学博雅特聘教授、IEEE Fellow、鹏城实验室网络智能部副主任兼云脑研究所所长、2018年国家杰出青年科学基金获得者、首届“高校计算机专业优秀教师奖励计划”获奖者；主要研究方向为分布式机器学习、神经形态视觉和视频大数据；累计主持国家、省部级与企业合作项目40余项；获国际期刊和会议最佳论文奖2次、国家与省部奖4次；发表论文280余篇，拥有中国和美国发明专利90项。

高文，北京大学博雅讲座教授、中国工程院院士、ACM Fellow、IEEE Fellow、鹏城实验室主任、北京大学信息与工程科学部主任，曾任中国科学院计算技术研究所研究员、副所长、所长，中国科学技术大学副校长，中国科学院研究生院常务副院长；主要从事人工智能应用和多媒体技术、计算机视觉、模式识别与图像处理、虚拟现实方面的研究；获得国家技术发明奖一等奖1次、二等奖1次，国家科技进步奖二等奖5次，获得2005中国十大教育英才称号和中国计算机学会王选奖。