DOI:10.3969/j.issn.1003-5060.2023.01.009

基于重计算的深度学习加速器容错设计

王乾龙, 许达文
(合肥工业大学 电子科学与应用物理学院,安徽 合肥 230601)

摘 要:2D计算阵列由于高并行性且通信简单,在深度学习加速器(deep learning accelerator,DLA)中经常负责处理卷积的大量计算,若出现硬件故障,则会导致计算错误,从而造成预测精度大幅下降。为了修复2D计算阵列中的故障,文章提出一种用于容错DLA的重计算结构(recomputing architecture,RCA),与传统的在阵列中添加冗余的即时故障修复策略不同,它具有一组基于冗余的重计算单元(recomputing unit,RCU),可以在稍后的周期中一对一地进行故障单元的重新计算。实验结果表明,与之前的容错方案相比,该文提出的方法显示出更高的故障修复能力和可扩展性,并且芯片面积占用更少。

关键词:重计算结构(RCA);深度学习加速器(DLA);容错;重计算
中图分类号:TP183 **文献标志码**:A **文章编号**:1003-5060(2023)01-0054-06

Fault-tolerant design of deep learning accelerator based on recomputing

WANG Qianlong, XU Dawen
(School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei 230601, China)

Abstract: Due to its high parallelism and simple communication, 2D computing arrays in deep learning accelerator(DLA) are often responsible for processing a large number of calculations of convolution. If there is a hardware failure, the calculation error will result in a significant decrease in the prediction accuracy. In order to fix faults in 2D computing arrays, this paper proposes a recomputing architecture(RCA) for fault-tolerant DLA, which is different from the traditional real-time fault repair strategy of adding redundancy in the array. It has a set of redundancy-based recomputing units(RCU) that can be used to recomputing the failure units one-to-one later in the cycle. Experimental results show that, compared with the previous fault-tolerant schemes, the proposed method has higher fault repair capability and scalability, and less chip area occupancy.

Key words: recomputing architecture(RCA); deep learning accelerator(DLA); fault tolerance; recomputing

深度学习加速器(deep learning accelerator,DLA)作为一种专用于深度学习任务的ASIC设计,因为具有高性能、低功耗的特点,所以在可便携设备中大量部署,广泛应用于人工智能场景中。然而深度学习计算高度密集,每一层都包含复杂的卷积计算任务,给硬件实现带来一些难题。2D计算阵列因为能很好地处理矩阵乘法和卷积计算,所以在DLA中被广泛使用。比如谷歌的深度学习加速器TPU包含了规模为256×256大小的计算阵列,其中每个处理单元(processing element,PE)承担着复杂的卷积计算任务。

随着半导体技术变得越来越复杂,工艺偏差也可能使DLA制造过程中产生很多缺陷,一个严重的影响就是永久性故障的增加。一方面,在本文实验中,PE的错误会导致很多计算结果错误,错误偏差在神经网络任务执行期间不断累积,

收稿日期:2021-06-23;**修回日期**:2021-11-21
基金项目:国家自然科学基金资助项目(61834006)
作者简介:王乾龙(1997—),男,安徽阜阳人,合肥工业大学硕士生;
许达文(1986—),男,江西新干人,博士,合肥工业大学副教授,硕士生导师.

显著降低分类精度;另一方面,尽管可以在芯片制造后期测试过程中检测出故障,但丢弃每一个带有故障的芯片会显著降低良品率。因此对出现故障的芯片进行修复是维持芯片正常功能、减少生产商损失的重要手段。因为 2D 计算阵列承担着 DLA 的几乎全部计算任务,对神经网络任务的执行非常重要,所以本文主要关注对 2D 计算阵列中故障的修复。

研究人员在解决计算阵列中故障带来的影响时,通常有 2 种思路。第 1 种思路是通过硬件方法解决,早期的学者通常会删去故障所在的行或列,从而得到无故障的阵列。这类方法虽然简单,但是会带来严重的性能损失。后来研究人员提出利用冗余 PE 来替换阵列中的故障 PE[1-2],但当故障分布不均匀时,这些方法也很容易失效。还有研究人员[3]采用统一的点乘单元作为冗余处理单元,并在进一步的工作中增加了分组的点乘单元与故障检测功能[4]。虽然故障修复效果较好,但是存储连续周期内所有进入阵列的数据需要额外的存储开销。第 2 种思路是通过软件方法来解决,一些学者针对故障计算阵列重新训练深度学习模型,虽然可以很好地维持模型的精度[5],但是模型重新训练的过程非常耗时,而且很难用于其他故障配置。总之,仍然缺少 DLA 的容错计算阵列架构、不能够在原始模型下运行并以较低的资源开销可以同时容忍各种故障配置。

为了解决这些问题,本文提出一种用于容错 DLA 的重计算结构(recomputing architecture,RCA)。与将同类冗余 PE 添加到计算阵列内部的工作不同,本文实现了计算阵列外基于冗余的重计算单元(recomputing unit,RCU)。RCU 中的每个 PE 负责计算阵列中单个故障 PE 的重新计算,通过将故障重新计算所需的部分数据进行延迟,RCU 在稍后的周期中将重新计算 2D 计算阵列中故障 PE 上的所有输出特征。当 2D 计算阵列中故障 PE 的数量不超过 RCU 大小时,RCA 可以完全修复 2D 计算阵列。即使 PE 错误率进一步增加,RCA 可以通过增加 RCU 的冗余规模,进一步提高容错能力,也可以选择修复出最大的计算阵列区域,减少性能损失。

1　相关工作

1.1　传统的冗余设计方法

早期的研究人员通常在计算阵列中添加冗余单元来减轻故障对阵列的影响。不同冗余设计方法中冗余的放置如图 1 所示。图 1a、图 1b 所示分别为行冗余(row redundancy,RR)和列冗余(column redundancy,CR)[1],它将冗余放置在阵列的一侧,其中的每个 PE 完成对一列或一行中故障的替换。图 1c 所示为对角线冗余(diagonal redundancy,DR)[2],它将冗余 PE 放置在阵列的对角线上,每个 PE 完成对一行和一列中故障的替换。阵列中的 PE 出现故障时,都能找到对应的冗余 PE 来替换,但是当一个区域中故障 PE 的数目多于其对应的冗余个数时,这些方法往往也无法完全恢复计算阵列。

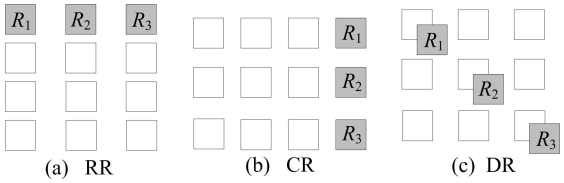


图 1　不同冗余设计方法中冗余的放置

1.2　阵列中故障对精度的影响

为了分析故障对计算阵列的影响,本文将随机位翻转故障注入 32×16 计算阵列中 PE 的寄存器中,实现用于故障分析的模拟器,并使用 Mnist 数据集运行典型的手写数字分类任务,实验结果如图 2 所示。从图 2 可以看出,精度随着 PE 错误率的增加而大幅下降,表明错误 PE 的增加对预测准确率有着非常严重的影响。因此,对于关键任务应用程序来说,对计算阵列的保护十分有必要。

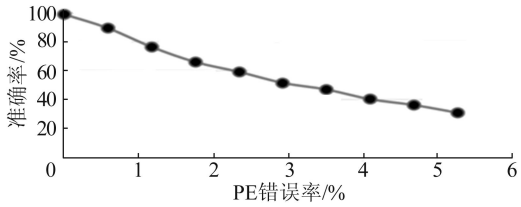


图 2　不同 PE 错误率神经网络模型在 DLA 上的预测精度

2　用于容错 DLA 的重计算结构

2.1　RCA 整体结构

为了修复 2D 计算阵列任意位置的故障,本文提出一种重计算结构(RCA),它具有由多个 PE 构成的重计算单元(RCU),可以一对一地进行 2D 计算阵列中任意位置故障 PE 的重新计算,并完成对错误结果的替换。具有 RCA 的 DLA 如图 3 所示。

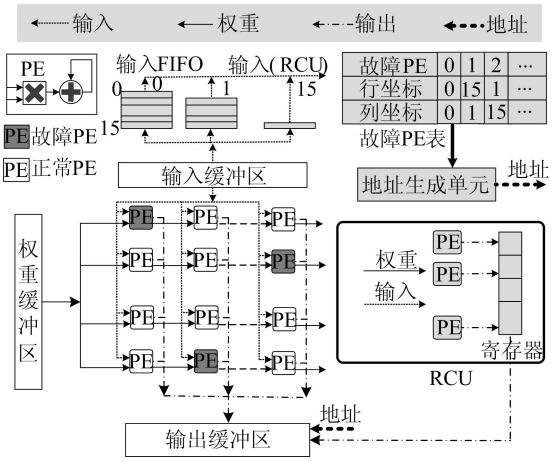


图 3 具有 RCA 的 DLA

2D 计算阵列大小为 16×16,其中每个 PE 按给定的输出固定数据流[6]顺序地计算不同的输出特征。同一列的 PE 共用输入数据,权重则在不同列之间依次向右传递,输出特征的局部计算结果保存在 PE 本地。对于完成一个输出特征的计算,后一列结果的输出总是比前一列慢一个周期。而 RCU 需要同时并行地进行多个不同列中故障 PE 对应的输出特征的计算。因为 RCU 所需的输入数据储存在片上缓冲区多个地址中,并且增加多个读取端口会带来更多芯片面积开销,所以难以从片上缓冲区中获取 RCU 所需要的数据。

为了解决这一问题,本文延迟了重新计算,因为权重在计算阵列中依次向右传递,所以将延迟设置为从 PE 的第 1 列传输到最后一列所需要的 16 个周期。在重新计算的过程中,对于权重数据,RCU 可以直接从最后一列的 PE 中进行读取。因为前一列 PE 总是比后一列 PE 早一个周期用到相同的权重,在权重向右传递的过程中,其对应的输入数据必须保留以进行接下来的重新计算,所以本文增加输入先入先出队列(first in first out,FIFO)来对输入数据进行存储以达到延迟读取的目的,输入 FIFO 采用多个深度依次递减的 FIFO 组成,以对不同 PE 列的输入数据进行对齐操作,确保 RCU 能够同时并行进行多个故障 PE 的计算。因为 RCU 中的每个 PE 都可以完成单个故障 PE 的所有操作,所以当故障 PE 的数目不超过 RCU 大小时,RCU 总是可以完成映射到故障 PE 操作的重新计算。此外,RCA 还有一个故障 PE 表来记录 2D 计算阵列中故障 PE 的坐标。RCU 中的 PE 可以通过坐标选择对应于某个故障 PE 的权重和输入特征,从而进行重新计算。同时,这些坐标还可以指示地址生成单元确定写入输出缓冲区的地址,将重新计算的结果替换错误结果。此外 RCU 输出还有一个寄存器文件,用于存储 RCU 的计算结果以及将结果写入到输出缓冲区。

2.2 RCA 的故障修复过程

本文以带有 3 个故障 PE 的 16×16 规模大小的 2D 计算阵列的 RCA 为例,来进一步说明 RCA 中的故障修复过程,其中 RCU 包括 16 个 PE,RCA 的故障修复过程如图 4 所示。

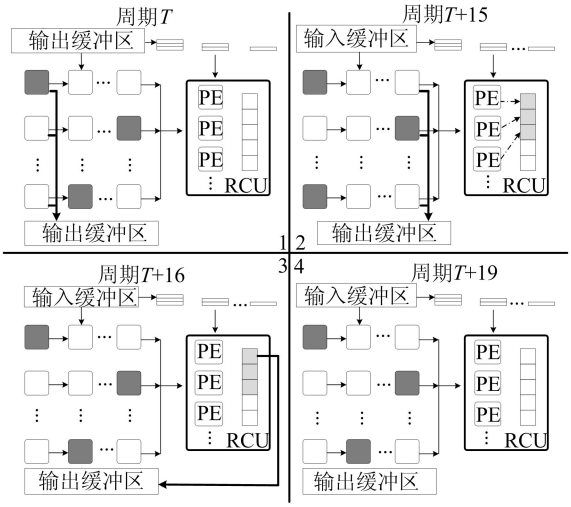


图 4 RCA 的故障修复过程

假设 2D 计算阵列的 PE 需要 T 周期才能产生完整的输出特征。从周期 T 开始,PE 的第1 列开始将输出特征写入到输出缓冲区。2D 计算阵列将连续占用输出缓冲区 16 个周期,将 16 列的输出结果依次写入到输出缓冲区中。在整个过程中,RCU 一直都在进行重新计算操作,由于 RCU 的重新计算只比最后一列慢了一个周期,接下来 RCU 将继续占用输出缓冲区。具体步骤如下:

(1) 在周期 T 时,2D 计算阵列写入开始。PE 的第 1 列将生成的输出特征写入到输出缓冲区,并开始计算新的输出特征。因为有 3 个 PE 出现故障,所以用深色来标记 PE 的错误计算结果。用于故障 PE 上的权重继续向右传递,输入特征则存储在输入 FIFO 中,经过 FIFO 的延迟之后它们会被读取到 RCU 中,以便进行重新计算。

(2) 在周期 $T+15$ 时,2D 计算阵列写入结束。RCU 根据故障 PE 表中的故障 PE 的坐标,分别读取对应 3 个故障 PE 的输入特征和权重,并完成 3 个输出特征的计算。最终计算结果会被写入到输出寄存器中。

(3) 在周期 $T+16$ 到周期 $T+18$ 时,因为 2D 计算阵列中只有 3 个故障 PE,所以需要 3 个周期才能将重新计算的结果从 RCU 中的寄存器写入到输出缓冲区。写入地址由 AGU 根据故障 PE 表中的坐标确定。并通过掩码的形式只更新错误的计算结果。因为 T 通常远大于 16,所以有足够的时间来将重新计算的结果更新到输出缓冲区而不会发生写冲突。同时,RCU 开始重新计算映射到 2D 计算阵列中的故障 PE 的最新输出特征,结果保存在 PE 本地。

(4) 从周期 $T+19$ 到周期 $2T-1$,RCU 占用输出缓冲区结束。2D 计算阵列和 RCU 都在进行新的输出特征计算,并将部分计算结果保存在本地,在计算出新的输出特征之前,输出缓冲区端口将处于空闲状态。

当 2D 计算阵列中的故障数目增加时,RCU 可以通过增加冗余 PE 的数量以进一步提高 RCA 的故障修复能力。当故障 PE 的数量继续增加并超出 RCA 的容错极限时,本文放弃恢复完整计算阵列的目标,转为修复出一片可用的计算阵列区域。为了尽可能保留计算能力,本文选择从左到右进行故障修复以得到连续可用的计算阵列区域,并丢弃无法修复的故障 PE 列。在这项工作中,为了保留连续可用的计算阵列,RCA 先修复最左侧的故障 PE,将其坐标写入故障 PE 表里,按照之前讲述的故障修复过程就可以修复出一块可用的计算阵列区域。

2.3　RCU 结构

RCU 可以完成 2D 计算阵列中故障 PE 的重新计算,这就要求其能从多个输入和权重中选出对应于故障 PE 的数据,RCU 结构如图 5 所示。

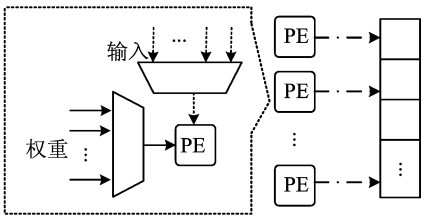


图 5　RCU 结构

由图 5 可知,RCU 中的每个 PE 都是相互独立的,分别承担不同故障 PE 的重新计算任务。每个 PE 都对输入和权重增加了多路复用器。根据故障 PE 表里的坐标,故障 PE 重新计算所需的权重和输入特征将被选出,并在 PE 中进行重新计算。一段周期后,计算完成的输出特征将被保存在输出寄存器中,并在接下来的时间内写入到输出缓冲区替换错误的计算结果。

3　实　　验

3.1　实验设置

(1) RCA 配置。本文提出的带有 RCA 的深度学习加速器在 Verilog 中实现,并用 TSMC 40 nm技术下的 Design Compiler 编译而成。所使用的权重和输入特征的数据宽度均为 8 bit。除此之外,一些单元的配置信息见表 1 所列。

表 1　RCA 配置

单元	计算阵列	RCU	输入 FIFO/kb	故障 PE 的存储量/Byte
配置	16×16	16	0.125	16

(2) 故障模型。受到半导体制造工艺的影响,芯片制造过程中由于相同缺陷所导致的故障往往倾向于聚集在一起。因为不同的故障配置对实验结果的影响很大,为了增加实验的可靠性和说服力,所以本文在随机分布模型的基础上,增加了集群分布模型[7]。同时,为了降低故障分布对实验结果的影响,类似于文献[8]中的工作,本文采用 PE 错误率作为度量,对每个 PE 错误率随机生成 10 000 个错误配置,并对实验结果进行平均。

3.2　芯片面积开销比较

具有不同冗余方法的 DLA 的芯片面积如图 6 所示。

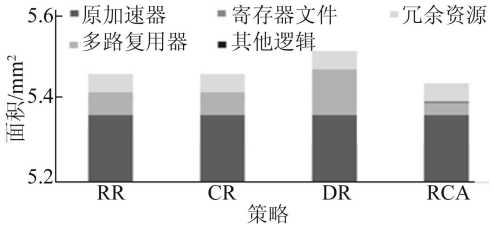


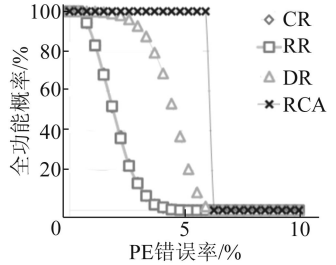
图 6　不同冗余设计策略下的芯片面积

因为 2D 计算阵列为 16×16 规模大小,基于 RR、CR、DR 的传统设计中冗余 PE 数量相同,但是 DR 需要增加 1 倍的多路复用器来实现对一行 PE 和一列 PE 中的故障替换,所以需要消耗更多的芯片面积。与传统的冗余设计相比,基于 RCA 的设计展现出更少的冗余开销。虽然 RCU 在进行数据选择时也使用了一部分多路复用器,但在冗余 PE 相同的情况下,所用的多路复用器和寄

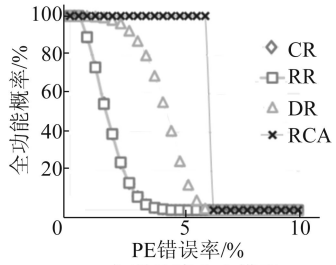
存器文件的开销要小得多。

3.3 故障修复能力比较

本文用 2 个主要指标来评估不同冗余设计方法的故障修复能力,类似于文献[3-4]中的工作。其中一个指标是全功能概率,指的是完全修复 2D 计算阵列中所有故障的概率。因为 RCU 中一个冗余 PE 只能修复一个故障 PE,所以当故障 PE 的数目超过 RCU 的容错能力时,全功能概率迅速降为 0。基于不同冗余方法的 DLA 的全功能概率如图 7 所示。从图 7 可以看出,在相同的 PE 错误率下,RCA 的全功能概率仍然要比其他 3 种冗余设计方法高得多。



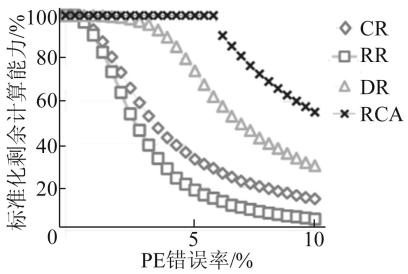
(a) 随机故障分布模型



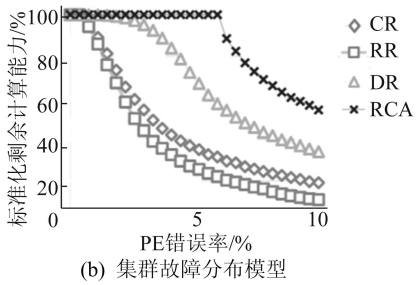
(b) 集群故障分布模型

图 7 基于不同冗余方法的 DLA 的全功能概率

当冗余设计方法无法对所有故障进行修复时,剩余的计算能力是衡量故障修复策略是否有效的关键。标准化剩余计算能力指的是剩余可用计算阵列占原始阵列大小的百分比。具有不同冗余方法的 DLA 的标准化剩余计算能力如图 8 所示。



(a) 随机故障分布模型



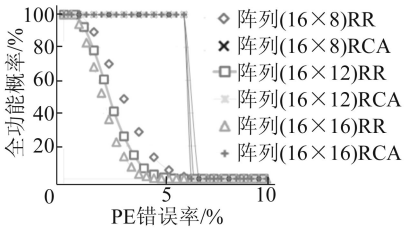
(b) 集群故障分布模型

图 8 具有不同冗余方法的 DLA 的标准化剩余计算能力

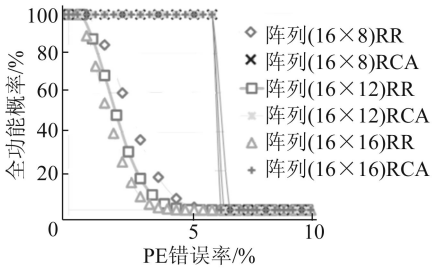
从图 8 可以看出,当故障数目超过 16 时,RCA 在不能修复所有故障的情况下,仍表现出较高的剩余计算能力。这说明了 RCA 的故障修复更加灵活有效,而其他冗余设计的优化空间则很小。

3.4 可扩展性比较

不同的应用程序对 DLA 的性能要求不同。为了分析比较不同的冗余设计方法的可扩展性,本文设置了 3 种阵列规模来进行全功能概率实验,类似于文献[3]中的工作。为了公平比较,实验只对比了冗余数目随阵列规模同等比例扩展的 RR 和 RCA。不同计算阵列大小上 RCA 和 RR 的可扩展性如图 9 所示。从图 9 可以看出,RCA 在计算阵列扩展时,全功能概率几乎没有变化,表明 RCA 具有良好的可扩展性。相反,RR 的全功能概率随着阵列大小的增加明显变差。同时,在计算阵列规模固定时,RCU 的扩展可以修复更多的故障,可以预见到 RCA 的容错能力将与 RCU 的大小成正比,这一点本文不再赘述。



(a) 随机故障分布模型



(b) 集群故障分布模型

图 9 不同计算阵列大小上 RCA 和 RR 的可扩展性

4 结　论

关键场景中的 AI 应用对 DLA 的可靠性要求很高。基于 RR、CR、DR 等常规阵列的冗余方法虽然能简单地修复一些故障，但是容易受到故障分配不均的影响，即使冗余资源充足往往也无法修复所有故障。针对这一问题，本文提出了重计算结构(RCA)，与传统即时的故障修复策略不同，它有一组冗余 PE 组成的重计算单元(RCU)，能够从阵列边缘获得流经故障 PE 的数据，利用冗余 PE 重新进行故障 PE 的计算，并在存储单元中替换错误的计算结果。当 2D 计算阵列中故障 PE 的数量不超过 RCU 大小时，RCA 可以完全修复 2D 计算阵列。即使 PE 错误率进一步增加，RCA 可以通过增加 RCU 的冗余规模，进一步提高容错能力，也可以选择修复出最大的计算阵列区域，减少性能损失。实验表明，RCA 具有较高的可扩展性和较小的芯片面积开销，并在故障修复能力上大大优于之前的冗余方法。

[参　考　文　献]

[1] TAKANAMI I, HORITA T. A built-in circuit for self-repairing mesh-connected processor arrays by direct spare replacement[C]//2012 IEEE 18th Pacific Rim International Symposium on Dependable Computing. [S. l.]: IEEE, 2012: 96-104.

[2] TAKANAMI I, FUKUSHI M. A built-in circuit for self-repairing mesh-connected processor arrays with spares on diagonal[C]//2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC). [S. l.]: IEEE, 2017: 110-117.

[3] XU D, CHU C, WANG Q, et al. A hybrid computing architecture for fault-tolerant deep learning accelerators[C]//2020 IEEE 38th International Conference on Computer Design (ICCD). [S. l.]: IEEE, 2020: 478-485.

[4] XU D, WANG Q, LIU C, et al. HyCA: a hybrid computing architecture for fault tolerant deep learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(10): 3400-3413.

[5] ZHANG J J, BASU K, GARG S. Fault-tolerant systolic array based accelerators for deep neural network execution [J]. IEEE Design & Test, 2019, 36(5): 44-53.

[6] LI L, XU D, XING K, et al. Squeezing the last MHz for cnn acceleration on FPGAs[C]//2019 IEEE International Test Conference in Asia (ITC-Asia). [S. l.]: IEEE, 2019: 151-156.

[7] MEYER F J, PRADHAN D K. Modeling defect spatial distribution[J]. IEEE Transactions on Computers, 1989, 38(4): 538-546.

[8] ZHANG J J, GU T, BASU K, et al. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator[C]//2018 IEEE 36th VLSI Test Symposium (VTS). [S. l.]: IEEE, 2018: 1-6.

(责任编辑　张　锴)

(上接第 46 页)

[13] CHO K, MERRIENBOER B V, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation [EB/OL]. [2022-12-14]. https://arxiv.org/abs/1406.1078.

[14] DZMITRY B, KYUNGHYUN C, YOSHUA B. Neural machine translation by jointly learning to align and translate[C]//Proceedings of Conference on Learning Representations. [S. l. : s. n.], 2015: 1409-1423.

[15] TOMAS M, KAI C, GREG C, et al. Efficient estimation of word representations in vector space [EB/OL]. [2022-12-14]. https://arxiv.org/pdf/1301.3781.pdf.

[16] 郭学慧，王志强. 聋校听觉障碍学生书面语语法偏误研究综述[J]. 绥化学院学报，2020，40(1)：35-38.

[17] 张帆. 国内近年来聋人学生汉语书面语句法研究述评[J]. 长春大学学报(社会科学版)，2015，25(6)：133-136.

[18] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[19] SUTSKEVER I, VINYALS O, LE O V. Sequence to sequence learning with neural networks[C]//Proceedings of Conference and Workshop on Neural Information Processing Systems. [S. l. : s. n.], 2014: 3104-3112.

[20] LUONG M, HIEU P, CHRISTOPHER D M. Effective approaches to attention-based neural machine translation [C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. [S. l. : s. n.], 2015: 1412-1421.

[21] KALCHBRENNER N, BLUNSOM P. Recurrent continuous translation models[C]//Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing. [S. l. : s. n.], 2013: 1700-1709.

(责任编辑　李　凯)