分　类　号：TP391　　单位代码：10183

研究生学号：　2020532068　　密　　级：公　开

吉林大学

硕士学位论文

（学术学位）

面向脉动阵列加速器的可靠性分析及优化研究

Reliability Analysis and Optimization Study for Systolic Array Based Accelerator

作者姓名：王麒翔

专　　业：计算机科学与技术

研究方向：计算机系统可靠性

指导教师：谭婧炜佳 副教授

培养单位：计算机科学与技术学院

2023 年 5 月

面向脉动阵列加速器的可靠性分析及优化研究

Reliability Analysis and Optimization Study for Systolic Array Based Accelerator

作 者 姓 名：王麒翔

专 业 名 称：计算机科学与技术

指 导 教 师：谭婧炜佳 副教授

学 位 类 别：工学硕士

答 辩 日 期： 2023 年 5 月 23 日

摘要

面向脉动阵列加速器的可靠性分析及优化研究

最近兴起的专用深度神经网络（Deep Neural Network, DNN）硬件加速器的设计热潮为深度神经网络带来了巨大的执行性能和能耗效率提升，并推动了卷积神经网络（Convolutional Neural Network, CNN）在人工智能，模式识别，图像分析等领域的广泛应用。硬件算力快速提升的背后是制造工艺的高度微缩化，这加剧了CNN加速器受到瞬时故障和永久故障影响的概率。随着加速器在新兴的安全关键领域（如自动驾驶汽车）的采用，其可靠性问题已变得非常严峻。其中脉动阵列（Systolic Array）作为一种具有代表性的CNN加速器的计算核心，能够实现数据的高效处理，故深入分析并优化脉动阵列的可靠性至关重要。

在本文中我们开发了一个微体系结构级别的故障注入框架saca-FI，用于分析基于脉动阵列的CNN加速器的可靠性。Saca-FI能够灵活地评估多种CNN架构和模型的可靠性，并且可以针对多种故障来源（如单比特翻转、多比特翻转、多个故障、持久性故障）和不同的层次（如模型的不同卷积层、脉动阵列中不同存储位置）进行深入分析。基于saca-FI框架我们评估了LeNet-5、CIFAR-10 CNN和VGG-16几个常用的卷积神经网络的可靠性。根据实验结果，我们观察到几个重要的架构级可靠性特征：CNN加速器架构的错误率与脉动阵列的大小和计算量密切相关；包含固定数据的寄存器比其他寄存器更容易受到影响；从0到1的翻转比相反的翻转更容易导致输出错误。然后，我们使用可靠性分析模型saca-AVF与saca-FI的执行结果进行了对比与分析，发现两者结论具有很高相关性系数。根据实验观察，我们提出添加调整系数，使得两者结果能够直接做比较。之后我们结合了CNN加速器的面积、执行速度、能耗与可靠性并综合讨论了各项指标之间的权衡。最后，基于上述发现的CNN加速器可靠性特征，我们提出了两种通过ECC校验技术来分别对寄存器类型和比特位置的保护措施。同时保护权重寄存器和输入寄存器可以在权重固定（Weight Stationary, WS）数据流模式下达到85.4%和88.9%的错误覆盖率。这样保护的平均开销为完全保护开

销的56.6%。保护更容易受到错误影响的指数位可以为模型带来100%分类精度。

关键词：

卷积神经网络加速器，脉动阵列，可靠性，故障注入，体系结构脆弱性因子

ABSTRACT

Reliability Analysis and Optimization Study for Systolic Array Based Accelerator

The recent surge in the design of dedicated deep neural networks (DNNs) hardware accelerators has brought huge performance and energy efficiency gains to deep neural networks and has driven the adoption of Convolutional Neural Network (CNN) in artificial intelligence, pattern recognition, image analysis, and other fields. Behind the rapid increase in hardware computing power is a highly miniaturized manufacturing process, which has also increased the probability of CNN accelerators being affected by transient and permanent failures. With the adoption of accelerators in emerging safety-critical areas such as self-driving cars, their reliability issues have become critical. The Systolic Array, a representative computational core of CNN accelerators, enables efficient data processing, so it is critical to deeply analyze and optimize the reliability of the systolic array.

In this thesis we develop a microarchitecture-level fault injection framework, saca-FI, for analyzing the reliability of systolic array-based CNN accelerators. saca-FI is able to flexibly evaluate the reliability of multiple CNN architectures and models, and can target multiple sources of faults (e.g., single-bit flip, multi-bit flip, multiple faults, persistent faults) and different layers (e.g., different convolutional layers of the model, different convolutional layers of the model, and different convolutional layers of the model). e.g., different convolutional layers of the model, different storage locations in the systolic array) for in-depth analysis. Based on the saca-FI framework we evaluated the reliability of several commonly used convolutional neural networks such as LeNet-5, CIFAR-10 CNN and VGG-16. Based on the experimental results, we observe several important architecture-level reliability features: the error rate of the CNN accelerator architecture is closely related to the size of the systolic array and the amount of computation; registers containing fixed data are more susceptible than others; and

flipping from 0 to 1 is more likely to lead to output errors than the opposite. Then, we compare and analyze the results of the execution using the reliability analysis model saca-AVF with those of saca-FI and find that the conclusions had a high correlation coefficient. Based on the experimental observations, we propose to add adjustment factors to make the two results directly comparable. We then combine the area, execution speed, energy consumption and reliability of the CNN accelerator and discuss the trade-offs between the metrics. Finally, based on the CNN accelerator reliability characteristics found above, we propose two protection measures for register type and bit location by ECC check code technique protection, respectively. protection of weight registers and input registers can achieve an error coverage of 85.4% and 88.9% in Weight Stationary (WS) dataflow. The average overhead of such protection is 56.6% of the full protection overhead. Protecting the more error-prone exponent bits can result in 100% classification accuracy for the model.

Keywords:

Convolutional Neural Network Accelerator, Systolic Array, Reliability, Fault injection, Architectural Vulnerability Factor

目　录

第 1 章 绪论

1.1 研究背景及意义

深度神经网络[1]被大量部署于大型数据中心，并且在自动驾驶、模式识别等领域有着广泛应用，已经成为了全球经济发展的重要推动力。作为一种常见的DNN，卷积神经网络凭借其卓越的计算性能在图像[2]、语音识别[3]、目标检测[4]等领域得到了广泛应用。近年来，卷积深度神经网络专用硬件加速器开始兴起，如谷歌的张量处理单元（Tensor Processing Unit，TPU）TPU[5][6]、加速器Eyeriss[7]等。这些CNN加速器能够优化存储层次结构中不同类型的数据移动，因此可以实现对神经网络的高性能及高效率处理。

随着CNN加速器被应用于自动驾驶汽车[8]和飞行控制[9]等新兴的安全关键领域，其可靠性变得至关重要。近些年，随着晶体管制备程工艺的发展，其特征尺寸不断减小且芯片上集成的晶体管数量成规模的增加，这将导致硬件故障的概率不断上升。这种影响具体体现在以下的两个方面：首先，空间中高能粒子（例如，α粒子和中子）扰动而导致瞬时故障的出现变得频繁。这已经成为影响CNN加速器等超大规模集成电路芯片可靠性的重要因素与亟需解决的问题[10]。瞬时故障能够导致电路的逻辑状态、存储单元以及处理器中的比特位发生翻转（如从“0”翻转成为“1”），造成操作数、指令等的改变或执行失效。虽然瞬时故障不会对集成电路造成持久性的伤害，但是却可能会在实际运行时导致不可预知的错误的运行结果，而进一步造成碰撞甚至坠毁等严重后果[11]。其次，由晶体管老化和电路损坏等因素引起的持久性故障也会严重影响硬件的可靠性。持久性故障会造成部件持久性损坏，所有映射到该部件的任务都无法正确执行或存储。由此可见，瞬时故障和持久性故障严重阻碍了CNN应用在对安全性和可靠性要求较高的领域中的使用与部署。故分析故障对CNN加速器执行过程的影响并提出高效的可靠性提升策略十分重要。

在本文中，我们使用两种方法对CNN加速器的执行过程进行可靠性分析。首先我们根据CNN加速器的执行方式，建立了一个微体系结构级别的分析框架

saca-FI。该框架能够全面分析加速器执行过程中产生的各种类型的故障，并通过统计结果错误率指标来量加速器的化可靠性特征。随后，我们比较了故障注入和一种适用于CNN加速器的体系结构分析模型saca-AVF的结果，并取得了一致性的可靠性分析结论。然后我们提出了一个性能综合分析指标PERA，为硬件设计者提供有效的可靠性提升的权衡。最后我们依据在实验当中的观察与分析结果，提出了两个能够提升加速器可靠性的改进策略。

1.2 课题研究现状

可靠性分析一般有三种方法:分析建模[12-14]、软件故障注入[15][16]、硬件波束辐射试验[17-20]。对于分析建模方法，Schorn[12][13]等人提出了两种可靠性建模和优化方法。他们基于神经网络[12]的深度泰勒分解估计单个神经元的弹性，然后从跨层角度引入弹性属性来预测比特翻转故障[13]。Ping等人[14]提出了一种快速评估CNN层可靠性的分析模型。在故障注入方法方面，Reagen等人[21]提出了一个快速量化神经网络弹性的框架。Chen等人[16]提出TensorFI分析框架，通过将故障注入张量流图中以评估一般机器学习应用的弹性。这几个工作的焦点都集中在DNN模型上，而非加速器架构。Li 等人[11]分析了深度神经网络的故障传播特性，并研究了对DNN加速器硬件的可靠性的影响。而这项工作并没有考虑架构设计的选择，如脉动阵列的大小和数据流映射。对于波束测试，Dos Santos等人[17]使用波束实验来确认故障模拟可以准确预测在GPU上运行的工作负载的错误率。Benevenuti等人[18]通过故障注入和中子辐照两种方法探讨了FPGA中神经网络的推理误差，表明这两种实验方法可以优势互补。Rech Junior等[19][20]则利用高能中子束在TPU上对多个CNN模型的卷积运算进行辐射实验。

此外基于对CNN加速器可靠性的理解，领域内还有很多关于CNN加速器可靠性改进机制的研究，Pandey等人[21]提出了GreenTPU，它可以识别脉动阵列的致错激活序列的模式，并间歇性地增加工作电压，以改善现有的定时错误缓解技术。Zhang等人[22][23]通过门电路仿真分析了TPU性能的下降原因，并提出了故障感知剪枝技术来应对持久性故障，如可以通过修剪或增加旁路电路去除故障单元的连接提升系统的容错能力。Cho等人[24]针对脉动阵列提出了一种基于分割和重新安排脉动阵列元素之间的连接冗余架构。此外他们采用基于群体的修复策略在

降低能耗开销的同时，最大化冗余利用率。

综合以上来看，建模分析是快速和低成本的，但其结果相对来说不太准确，上界较高。硬件波束辐射试验精度高，但成本高且仅适用于已制造的芯片，因此不能在早期设计阶段应用。软件故障注入是一种成本相对较低、精度较高的可靠性分析方法，通常用于应用层、体系结构层、RTL层等不同层次的可靠性分析。近年来，针对神经网络的可靠性分析[11][25]，提出了很多建模和故障注入方法。然而，他们中的大多数关注CNN模型本身（即在软件级别），只有少数考虑CNN加速器以及架构级别的可靠性特征。Papadimitriou等人[26]比较了软件级和架构级故障注入下的错误率结果，发现高级别可靠性分析可能会提供误导性的结论。此外软件层面的分析存在其固有的局限性，比如高级的指令（数据）可以对应于低级的多个指令（数据），故高级指令（数据）的故障也可以对应于低级指令的多个故障。再如高层软件级分析中有一些操作在低层中没有对应的操作，因此高层技术可能会注入在低层不会出现的虚假错误。因此，仅从软件层面进行分析不足以获取底层硬件的实际可靠性特征。

1.3 本文的工作

从体系结构层面分析有助于识别硬件全面的可靠性特征，并指导硬件架构师设计相应的高可靠性加速器。同时如1.2节所述，领域内现有的CNN加速器可靠性分析工作较少，且他们都没有从体系结构层面来衡量加速器的可靠性特征。此外，仅从软件层面进行分析不足以获取底层硬件的实际可靠性特征[26]。考虑到以上问题，本文主要进行了如下工作：

1. 提出了微体系结构级脉动阵列加速器可靠性分析框架saca-FI。该框架能够按计算周期模拟加速器执行过程中的PE（processing engine）级别运算，并在PE寄存器（register）级别注入指定类型故障。

2. 基于故障注入框架saca-FI进行深入分析。分别探究了模型的层次、寄存器类型、比特翻转方向、不同错误类型等角度，全面分析了脉动阵列加速器对于瞬时故障与持久性故障的可靠性特征。

3. 我们将一个现有的分析模型saca-AVF与分析框架FI这二者的可靠性分析结

果进行比较，发现二者的数据结果具有很高的相关系数。提出了添加适配系数DEF来调整AVF指标的一种改进方式，使得建模分析结果和故障注入分析的结果能够直接相互比较。

4.设计了一种性能综合指标PERA，并进行了性能-能耗-可靠性-面积综合分析。我们针对不同数据流提出了面积以及可靠性的设计权衡。

5. 根据脉动阵列加速器的可靠性特征，提出了两种相应的可靠性提升策略：使用ECC校验技术对脉动阵列加速器的PE内部更脆弱的寄存器进行保护；对脆弱性更高的数据位进行保护。我们通过实验证明了这两个策略都可以有效提升脉动阵列执行结果的精度以及分类正确性。

1.4 本文的结构

本文一共分为六章来介绍研究课题，每一章的主要内容总结如下：

第1章：绪论。首先阐述了对于CNN加速器进行可靠性分析与提升方面的研究背景以及意义。然后介绍了可靠性分析的常用方法，并对每种方法的特点做出了分析与比较。之后总结了领域内的研究现状，简要说明了本文的主要研究工作，最后介绍了本文的组织结构。

第2章：理论基础。首先阐述了CNNs的相关理论，包括了CNN模型结构、组成。随后介绍了CNN加速器的硬件架构及其内部的工作策略。介绍了两种CNNs执行过程中的故障：瞬时故障与持久性故障，解释了其产生原因以及造成的影响。

第3章：面向脉动阵列的可靠性分析框架。首先介绍了saca-FI基于的加速器模拟平台Scale Sim与keras深度学习框架。然后介绍了模拟过程与执行原理，框架内部的组成以及个模块之间的工作过程。

第4章：可靠性分析结果。应用分析框架saca-FI分别从处理器阵列尺寸、数据流映射策略、PE利用率等角度分析了脉动阵列加速器的可靠性特征。采用了一个分析模型saca-AVF与分析框架FI的可靠性分析结果进行对比，发现两者的数据结果具有很高的相关系数。最后还提出了添加适配系数DRF来调整指标AVF的一种改进方式。

第5章：卷积神经网络加速器可靠性提升方法。本章中提出了一个综合评价指标PERA，提出了性能、能耗、面积、可靠性之间权衡的见解。根据脉动阵列加速器的可靠性特征提出了两种保护策略。1）通过ECC校验技术保护PE内寄存器中的所有数据的指数部分。 2）保护更为脆弱的输入、权重寄存器。评估了方案的整体能耗开销。

第六章：总结与展望。在本章中对所有内容进行总结，并对本文工作后续存在的改进空间做出展望。

第2章 理论基础

2.1 卷积神经网络

2.1.1 神经网络模型

在过去的几年中，深度神经网络已经成为解决许多性能关键和能量受限任务的流行方法，例如实时对象检测、关键词识别和自动驾驶汽车。神经网络的层数直接决定了它对现实的表达能力。借由这一特点，深度学习神经网络在设计之初被加入更多的隐藏层来提取输入数据、特征图（ifmap）的更高级抽象。但是对于图像数据来说，参数量在层数加深的同时会产生急剧膨胀，这将会严重拖累网络效率。由此卷积神经网络应运而生，它提出的权重共享机制很好的解决了这个问题。下图2.1为一个小型CNN的整体结构示意图。

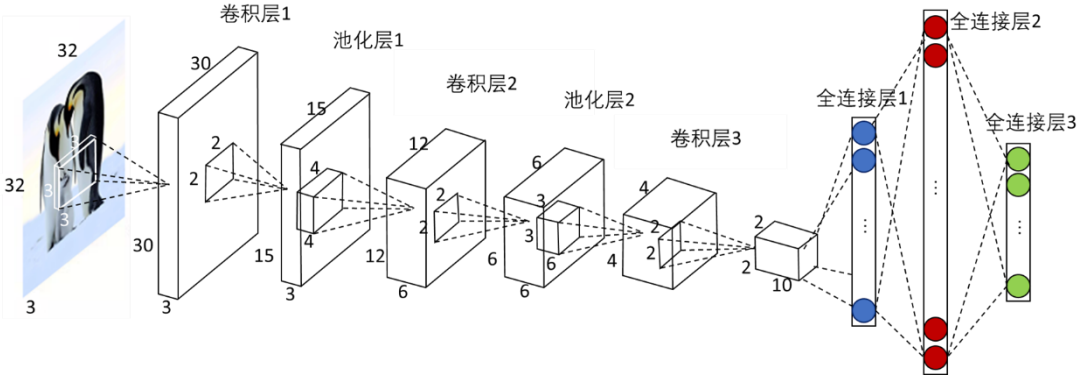


图 2.1 CNN 的整体结构

CNN网络中层的类型丰富多样，但计算主要分布在进行多维卷积计算的卷积层（CONV）和用于分类汇总的全连接（FC）层[1]。最近流行的模型中使用大量的卷积层，数量范围可以从三层到几十层不等。如图2.2，每个卷积层对输入特征图（ifmap）应用一组滤波器（卷积核），以提取底层特征并生成相应的输出特征图 （ofmap）。它的计算结果再通过非线性激活函数（例如，ReLU）处理后用作下一层的输入。通过引入非线性关系能够提升CNN模型处理复杂情境的能力。

全连接层常常被堆叠在卷积层之后，用来将卷积层提取的高级特征与分类标准进行相互关联、映射，并取得输出。此外还有一些具有特殊处理作用的层，例如池化层（Pooling）和批标准化（BatchNorm）层[27]。池化层利用了图像的像素

空间位置相关性特征，通过提取局部最值并舍弃其他冗余值，使输入的规模大幅缩小。批标准化层则是将的数据值标准化处理为0均值，1标准差的分布。如此处理能够提高激活函数对于数据的区分能力。

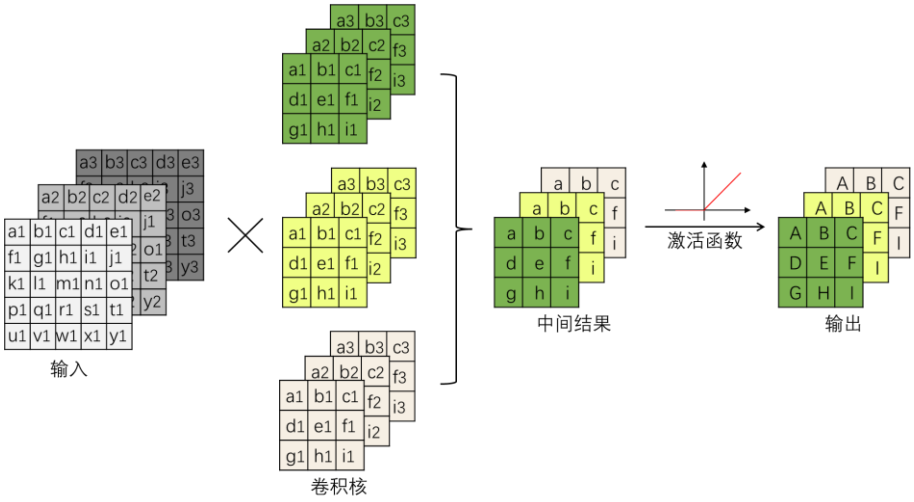


图 2.2 卷积层的计算过程

当确定好了CNN的拓扑结构，就可以利用目标训练数据进行学习。这个学习过程是通过反向传播（back propagation）的过程更新卷积核的权重来获得抽象特征的提取能力。具体表现为依据损失函数（loss function）迭代更新神经元，使网络对数据的有效处理过程，固定成为神经网络参数（相关权重值）。这个阶段通常也被称为网络的训练阶段。训练过程非常花时间，所以通常只进行一次。训练好的模型就可以使用测试数据集进行图像分类任务，这被称为网络的推理阶段，这个过程是可以反复执多次行的。例如在一个实际应用情境中，以一个自动驾驶的神经网络应用为例，自动驾驶系统中的CNN接收到的测试数据可能是路标，障碍物，行人的数字化图像。而作为输出指令的候选列表可能是加速、减速或是转向，每个可能的选择项都有一个置信度得分。

2.1.2 卷积神经网络加速器

深度学习应用程序的庞大计算量对硬件的算力提出了新的要求，传统的CPU与GPU并不能很好的支持大规模的神经元并行执行的MAC操作。CNN的不断发展以及广泛部署，带动了领域内诸多学者以及相关企业对于加速器的研究兴趣。在这其中比较流行的CNN加速器如谷歌张量处理单元TPU、加速器Eyeriss等。它们在设计层面对吞吐量和能源效率进行了优化，能够充分发掘神经网络的潜力并

降低硬件成本。这些加速器的主要设计思路通常集中在加速特定的“内核”，如卷积和矩阵乘法。这是因为对于CNN来说，其中的80%的计算量体现在卷积与全连接操作。通常的加速器设计包含了大量能够独立并行的乘法累加（Multiply And Accumulate, MAC）操作的执行单元（Processing Element, PE）。通过优化向量计算将传统的卷积操作映射为便于PE单元并行处理的计算模式来实现加速。

下面我们以谷歌张量处理器TPU的计算核心为例，介绍CNN加速器的架构。TPU引入层次存储结构来降低数据访存以及移动的能耗成本，内部结构如图2.3所示。芯片上的所有PE构成了TPU的计算核心，称为脉动阵列（Systolic Array）。脉动阵列利用简化的控制逻辑和模式化的数据移动来实现高计算吞吐量和低功率消耗。相邻的权重存储器和统一的全局缓冲区（Global Buffer）负责存储需要输入PE阵列的数据。底层结构被称为累加器，负责收集计算结果。数据可以从一个PE传递到另一个PE，这样既增强了数据重用，又能有效减少访存次数。数据在脉动阵列计算完成之后，累积器、激活和池化被执行。然后，结果被输出到统一的缓冲区，然后通过接口送回主机。右侧是PE的结构图。每个PE由一个乘法器、一个加法器以及三种类型的寄存器组成，分别用于临时保存权重weight、输入ifmap和部分和（partial sum, psum）数据。PE作为基本执行单元，执行并行乘法和累加（MAC）计算。在每个MAC操作当中，需要进行权重和输入的读写操作，最后将运算结果写回部分和寄存器。脉动阵列中的所有PE独立计算，并行执行计算过程。

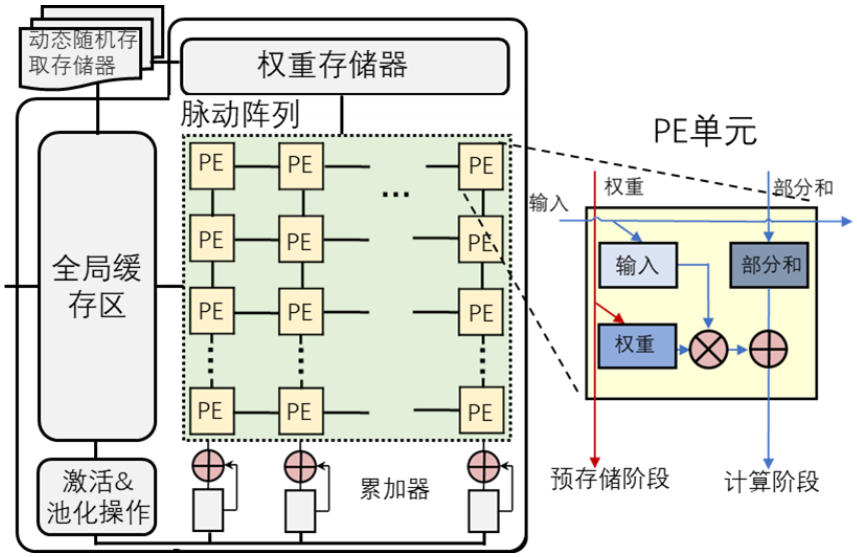
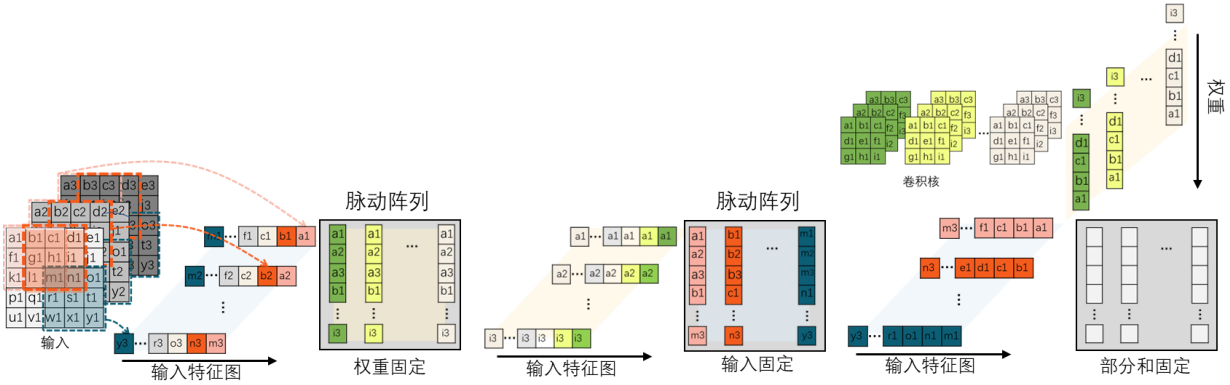


图 2.3 TPU 加速器结构

2.1.3 数据流

脉动阵列对数据进行策略性地缓存和重用，使得每个处理单元PE的数据具有很强的时间和空间位置相关性。处理策略能够有效提升数据重用度并减少运算能耗，这种特定的数据运算的模式被称为数据流（Dataflow）。不同数据流决定了数据的映射策略、数据的计算方法与传递方式，故不同数据流下数据的行为与分布也是不同的。本文分析了三种常见数据流对于脉动阵列加速器可靠性的影响程度。我们选取的三种数据流分别为：权重固定WS，输入固定（Input Stationary, IS）[28]和输出固定（Output Stationary, OS）。



（a）权重固定数据流　　（b）输入固定数据流　　（c）输出固定数据流

图 2.6 三种数据流及其数据映射过程

（1）权重固定数据流

图2.6 (a)显示了WS数据流中的数据映射策略。对于权重固定策略，权重元素在整个运算过程中被保留在PE权重寄存器中，直到所有需要这些值作为操作数的计算都结束。在数据流执行之前，加速器会对输入特征图和卷积核进行预处理。如图2.6 (a)左侧所示，输入特征图ifmap依据每一个卷积窗口被降维形成一维向量。同样如图2.6 (c)左上，参与卷积运算的卷积核被依次序转化成一维向量。

首先，原本的多个卷积核被按次序分配到脉动阵列的对应列。对于每一个脉动阵列的PE纵列分配一个卷积核，它的原本元素从顶部边缘传入，直到所有卷积核的权值都被送入PE权重寄存器当中。这个阶段被称为预存储阶段。卷积核的元素被存储好之后将进入计算阶段，输入特征图ifmap的元素按照周期依次从左边缘馈入横向传播。进行MAC计算之后，部分和会随之纵向传递到下一个寄

存器当中去。当映射在PE当中的权重的计算完成，就用新的权重集替换旧的权重数据，重复之前的映射过程。

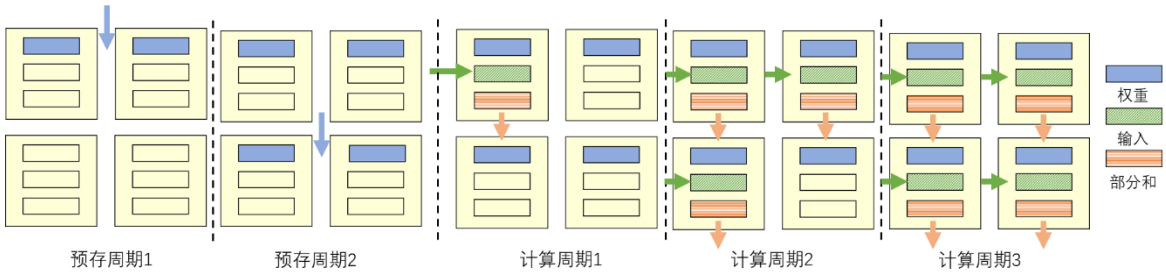
（2）输入固定数据流

图2.6（b)描绘了IS数据流的示意图。这种数据流通过固定输入特征图ifmap来实现输入的重用，执行方式与WS过程类似。但是这种情况下，每一列被分配的不再是一个完整的卷积核，而是一个关于输入特征图ifmap的卷积窗口的一组元素。这个卷积窗口所包含的元素通过计算能够生成ofmap中的一个最终输出元素。与WS的情况一致，对于给定的列，卷积窗口的元素从顶部边缘流入。一旦输入元素馈入完毕，卷积核向量的元素就从左边缘流入。当全部计算完毕的时候替换PE中固定的ifmap值进行下一轮的计算。

（3）输出固定数据流

图2.6(c)描绘了OS数据流的示意图。部分和直到计算完成前都固定在PE内部的部分和寄存器中。这种数据流有简易且较低的部分和读取能耗，它在三种数据流中通常是运行速度最快的。在这种模式中没有需要预存的数据，最终运算结果直接从寄存器中提取。在运算阶段，输入特征图横向传播，权重纵向传播，这两个过程是同步进行的。每个周期中当权重与输入特征图元素相遇时，便会进行一次MAC计算，产生一个部分和作为运算的结果。这个时候，这些部分和会在计算完成前一直驻留在PE中的部分和寄存器中。生成的输出可以认为在计算好之后被直接从PE中提取出去，而不会导致计算停滞。

为了方便理解PE寄存器中数据的流动，图2.7展示了WS数据流下寄存器内部的两种执行阶段：a)预存储阶段 b)计算阶段。如图2.7 (a)在预存储阶段，按周期将需要固定的权重传递到权重寄存器当中去。如图2.7 (b)在计算阶段，每一个周期中PE单元使用寄存器中的数据执行MAC操作，计算完成后部分和数据将纵向传递到相邻的下一个PE单元中去。下一个周期依然重复同样的操作，直到所有的输入特征图都完成MAC计算。而纵向传递的部分和会在离开脉动阵列后被累加器收集，等待进行后继的激活与池化操作。



（a）权重预存储阶段　　（b）计算阶段

图 2.7 WS 数据流执行过程示意图

2.2 故障类型

为了在安全关键应用中使用电子设备，必须对其可靠性进行评估，计算故障可能导致执行错误的概率。硬件可靠性是指硬件在一定的工作环境和条件下，在规定的时间内正确执行所需功能的能力。随着晶体管特征尺寸的缩小，处理器芯片的错误率也相应增加。在没有纠错方案的情形下会导致芯片错误率将与芯片上的位数呈正比增长。这些故障可以被划分为瞬时故障和持久性故障[29]。

瞬时故障基于单比特翻转，它产生的原因在于高能粒子在穿过半导体器件时生成的电子空穴对。晶体管源极和节点可以收集这些电荷。足够量的累积电荷可能会反转逻辑器件的状态，从而在电路操作中引入瞬时的逻辑故障。未来增加代码复杂性和缩小特征尺寸的趋势只会加剧这个问题。此外由晶体管老化和电路损坏引起的持久性故障会严重影响处理器硬件的可靠性。安全关键的DNN系统中出现的这些故障的后果可能是灾难性的，因此需要减轻故障以达到某些可靠性目标。故障对程序运行结果带来的影响大致可以分为以下三种，分别是对结果无影响（Masked），检测到不可恢复的错误（Detected Unrecoverable Errors, DUE）和静默数据损坏（Silent Data Corruption, SDC）。瞬时故障导致的瞬时故障可能会随着影响程序的运行被放大，也可能会被屏蔽。但持久性故障由于硬件特性被永久性改变，错误往往都会表现出来，持续影响应用程序的正确执行。

现在提出了很多相应的技术来应对故障，从特殊的辐射硬化电路设计到局部错误检测和校正到架构冗余[15]。然而，所有这些方法都会为性能、功耗、芯片尺寸和设计时间方面带来巨大的成本或损失。因此一种能够有效权衡系统脆弱性、提升可靠性并降低设计成本的指导性依据是必要的。

2.3 故障注入

对系统运行过程中的故障进行脆弱性评估是其中一种有效的策略，而这些技术在设计、硬件精度水平、评估吞吐量和评估粒度方面有所不同。故障注入是最常用的可靠性评估方法[15]，可以在任何抽象级别执行：从门和寄存器传输到微体系结构和软件级别。通过在每级模型的可访问资源（门、微体系结构结构、体系结构位置或指令）中注入故障[30]来测量故障影响应用程序执行的概率。

故障注入常常是通过大量的数据实验，来获得统计性质的系统应对某种错误类型的可靠性特征。但是由于脉动阵列PE数量、寄存器类型、故障发生的周期数、以及数据的比特位形成了一个非常大的可能性总体，实验很难模拟到所有可能的情景。因此对于这种复杂情境下的故障注入实验，其实验设置，以及抽样的方法需要被设计。在提出计算公式之前，我们先做出以下的一些假设：我们假设总体的特征（在我们的例子中，所有可能的瞬时故障在任何时钟周期）遵循正态分布。初始总体中的每个个体（即给定周期的给定故障配置）有相同的概率在样本中被选择。同时需要在随机抽样过程中使用均匀分布。

在故障注入中，初始个体总数N是有限而巨大的。N取决于硬件状态（可以被干扰的存储元件的数量）、故障模型（给定时间瞬时故障的位置和分布）以及工作过程（输入特征图ifmap所需的计算周期数、不同数据流的数据复用方式）。样本数量n可由工作[31]中的公式（2.1）计算：

$$n = \frac{N}{1 + e^2 \times \frac{N-1}{t^2 \times p \times (1-p)}} \qquad \cdots\cdots\cdots\cdots \qquad (2.1)$$

其中p代表故障可能引起错误的概率，通常取0.5。e与t都是与置信度计算的相关参数，e代表了误差区间的上下界，可以取5%。t代表了与置信水平对应的分界点，对应了精确值在误差区间内的概率，通常取95%。

表2.1抽样表

	t=1.96 (95%置信度)	t=2.57 (99%置信度)	t=3.09 (99.8%置信度)
e=5%	385	663	955
e=1%	9580	16520	23734
e=0.1%	776790	1177855	1503780

表2.1是我们选取初始个体总数N为一百万时，在对应置信度分数与误差区间下所需的抽样数量n。其他的初始个体总数所需要抽样数量n的情形依次类推。

2.4 本章小节

在本章主要介绍了课题涉及到的相关背景知识，对CNN网络结构，卷积运算，训练与推理，脉动阵列加速器架构等相关概念行了简单介绍。随后介绍了瞬时故障、持久性故障产生的原因以及影响。然后引出了故障注入的概念，并介绍了我们使用的置信度分析的方法。为后续的研究内容提供了技术背景。

第 3 章 面向脉动阵列的可靠性分析框架

3.1 引言

目前，基于脉动阵列的CNN加速器还没有微体系结构级的故障注入框架。从微体系结构层面全面分析可靠性有助于识别整体脆弱性特征，并指导硬件架构师相应地设计高可靠性加速器。在本章中，我们提出了基于脉动阵列加速器的可靠性分析框架saca-FI。并先介绍了构建分析框架的基础：一个开源的加速器模拟器Scale-Sim[28]，深度学习框架Keras。由于Scale-Sim并不能进行实际计算，故我们扩展了这项工作，开发了一个周期精确的脉动阵列加速器执行模拟器。随后我们逐一介绍了周期推算的过程、计算子集的划分等细节来再现执行模拟器的实现过程。然后我们详细介绍了可靠性分析框架的组成以及工作流程。

3.2 分析框架的构建

3.2.1 加速器模拟器

为了对加速器做出精确的性能分析与研究，我们选用Scale-Sim[28]作为研究脉动阵列微体系结构的基础，Scale-Sim是一个基于脉动阵列架构的开源、周期精确的DNN加速器模拟器。它能够提供基于脉动阵列加速器的设计权衡和映射策略见解。并展示了带宽、数据流对深度学习内核在整体运行时间和能量的影响。

Scale-Sim展示了各种微体系结构特性，例如阵列大小、阵列纵横比、内存大小、数据流映射策略，以及系统集成参数，例如内存带宽。它使设计者能够全面理解关键设计参数之间的相互作用——阵列大小、纵横比、数据流和内存带宽。Scale-Sim能够在不同的CNN加速器架构设计和不同的CNN模型下对CNN加速器的执行过程进行建模。它以微体系结构参数和每个DNN层的尺寸为输入，报告延迟、阵列利用率、静态随机存取存储器访问、动态随机存取存储器访问和动态随机存取存储器带宽需求。Scale-Sim可以模拟卷积和矩阵-矩阵乘法。Scale-Sim支持WS、OS、IS等数据流。

Scale-Sim为不同的数据流在周期层面上生成数据和脉动阵列的PE之间的映

射。例如，在一个给定的周期，输入数据和权重被用于阵列中哪一个PE。然而，Scale-Sim并没有实现微体系结构，也无法存储任何数据值（不支持卷积计算），它只生成数据的映射信息。故在这项工作中，我们扩展了Scale-Sim模拟器，以执行周期精确的PE级MAC计算操作。

除此之外，我们采用了keras深度学习平台来辅助开发我们的故障分析框架。keras是一个由Python代码编写的开源高级深度学习程序库，它可以选取运行于两种后端TensorFlow或Theano上。TensorFlow从低层次（如张量运算）来调节和控制深度学习模型的各种细节，因此它具有控制能力强、执行效率高的特点，此外它还支持各种平台。然而由于它的层次属于低级别的深度学习链接库，因此它的开发耗时更长，学习门槛很高。而keras能够用更为通用简易的高级接口来更加便捷地建立、训练深度学习模型，进行准确率评估与预测。

3.2.2 数据流的周期推算

为了模拟周期精确的故障注入，就需要推断出脉动阵列的正确执行周期。我们以一个简单的例子说明脉动阵列计算的过程以及周期的计算方法。假设有一个5×5×1的输入特征图与2个3×3×2的卷积核，采用WS数据流进行计算。下面我们将分析计算脉动阵列中数据流动的过程以及具体的执行周期数目。

计算执行过程如示意图3.1所示：可以看出，产生数据“重叠”（权重与输入）的部分即发生了一次MAC计算。当横向馈送的输入特征图“完全通过”权重矩阵的时候，所有有效计算过程完毕，脉动阵列完成了本轮计算过程。为了方便后续描述，我们标记了图中的输入特征矩阵，以及脉动阵列中的权重矩阵的尺寸。L1代表了输入特征在数据传输过程中所形成的平行四边形上底边长。L2代表了权重矩阵的宽，L1+L3则代表了输入特征构成平行四边形在底边投影的长度。

从上述流程可以得知，整个计算周期即输入特征图“完全通过”权重矩阵的周期数目。则计算周期可以计算为C_{calcul}=L1+L2+L3。其中L1由输出矩阵ofmap尺

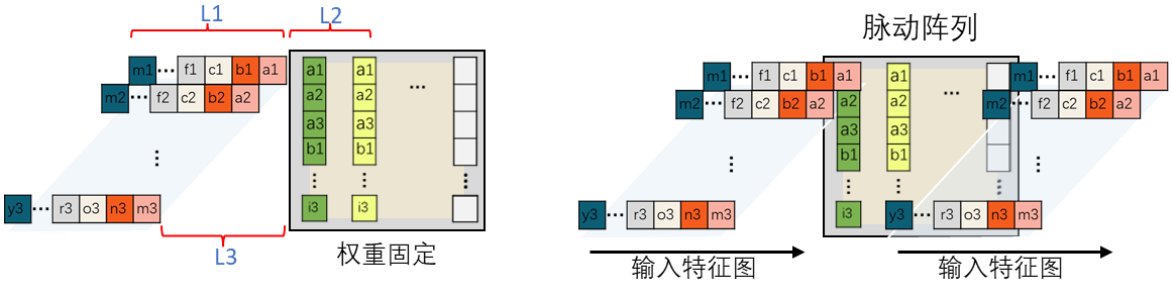


图 3.1 WS 数据流计算过程

寸决定，为$(\frac{5-3+1}{1})^2$=9，L2等于卷积核个数为2，L3则由卷积核的大小决定，值为3×3×2-1=17。此外对于一个完整的计算过程而言，还需要包含前期的数据预存储周期。这个值也由卷积核大小决定，为3×3×2=18。最后还有一个结果的写回周期。故执行计算的周期总计为：$C= C_{calcul}+ C_{pre} +C_{write}$= L1+L2+L3+18+1=47。

3.2.3 计算子集

在加速器执行计算过程中，被固定的模块（如WS模式下的权重矩阵，IS模式下的输入特征图）有时会由于它的形状尺寸没有办法一次性的放入脉动阵列，从而被拆分成好几个块来进行计算。这种划分数据的方式是与脉动阵列尺寸大小紧密相关的。待固定的矩阵会被算法进行切分成一个个计算子集，每次从这些子集中选取一个作为当前执行计算的矩阵。在计算结束之后会发生脉动中的整体数据替换，将当前计算完成的计算子集转换为新的一批数据。如此循环，直到所有计算子集完成计算。在将计算子集的计算结果进行汇总的时候遵循以下的策略：不同卷积核的计算子集结果进行通道并，相同卷积核的计算子集结果进行加操作。

不同的数据流模式，不同的输入数据，不同的脉动阵列尺寸大小影响着计算子集的划分行为，并产生不同的数据行为以及空间利用策略。因为脉动阵列的计算是时间-空间相关的，所以我们要对这些计算子集的切分过程加以考虑与区分，同时考虑到这种策略与周期的对应关系。

3.3 脉动阵列可靠性分析框架 saca-FI

在本节中，我们将详细介绍我们提出的saca-FI框架。图3.2显示了saca-FI的概述，它是由（1）一个用于脉动阵列的周期精确的执行模拟器，（2）一个灵活

的体系结构级故障注入模块，（3）一个可靠性分析框架组成。因为脉动阵列只包含数值计算，而我们只考虑存储错误，因为CNN加速器在几乎所有情况下都不会在故障注入下崩溃。因此，我们的框架中只考虑SDC错误。

图3.2中的配置信息表明，要向脉动阵列中的坐标为（2,2）的PE当中的部分和寄存器数据注入单比特翻转故障，注入的目标位为比特位的第28位（机器数）。触发故障注入的时机为当模拟器执行到第1057个计算周期的时候。故障注入完成之后执行模拟器继续进行脉动阵列仿真计算，执行结果作为ofmap被回传keras框架中进行后继推理，得到最终预测结果（狗0.94，狐狸0.03，猫0.02等等）。这个结果将会被传入分析框架与标准输出进行对比，同时生成本次完整故障注入过程的记录信息。以下我们将分别介绍每个模块的功能及其实现原理。

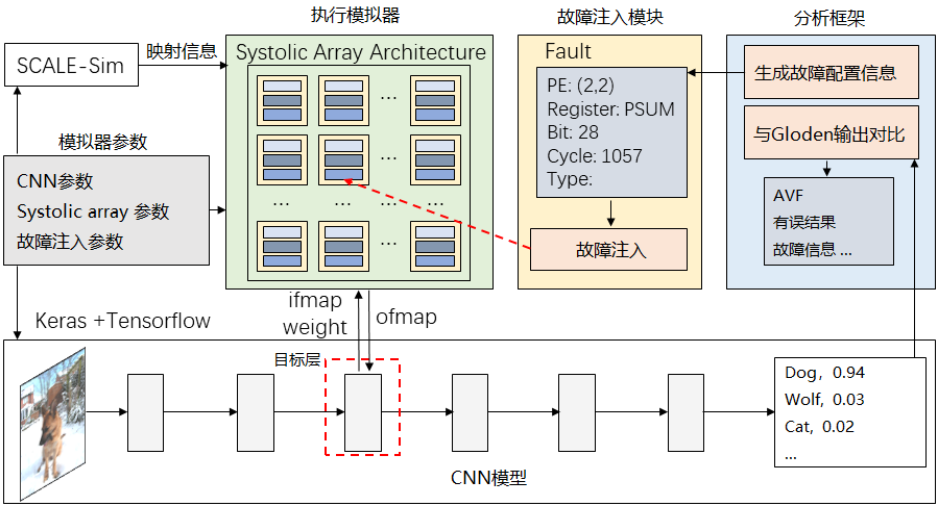


图 3.2 Saca-FI 框架概述

（1）执行模拟器：首先建立了一个周期精确的模拟器，以模拟脉动阵列加速器的微体系结构级别执行。Saca-FI首先需要获取输入特征图、权重与PE之间的映射信息 。而这些映射信息可以借由开源模拟器Scale-Sim获得。执行模拟器为脉动阵列中的每个PE模拟了三个寄存器，包括输入、权重和部分和寄存器。在每个周期中，一个PE将输入寄存器中的数值与权重寄存器中的数值相乘，并将结果与psum寄存器中的数值相加。之后，数据按照数据流策略被传送到相邻的PE。例如，对于权重固定WS和输入固定IS数据流，一个PE的psum寄存器中的值被纵向地传输到下一个PE。

我们使用keras框架和TensorFlow后端来获得CNN模型中各层的实际ifmap和

权重数据。然后，我们将keras的真实数据存储到PE寄存器中，并使用它们在PE层面进行MAC操作。我们的模拟是在卷积网络的层级别上进行的，并使用keras框架直接计算前置和后继的网络层来提高计算速度。为了验证模拟的正确性，我们将计算的层级输出矩阵与keras中得到的ofmap进行了比较，并取得了相同的运算结果。表3.1是saca-FI框架支持参数的列表。

表3.1 saca-FI参数列表

参数	解释
模型	实验选择的 CNN 模型
数据流	数据流，可选 WS,IS,OS
层信息	CNN 模型的层，如卷积层 2 层
寄存器	寄存器类型：权重，输入，部分和
脉动阵列尺寸	可选项：32,64,128,256,512
实验次数	执行故障注入的实验次数，如 10000 次
故障类型	可选项：瞬时故障，持久性故障

（2）故障注入模块：基于执行模拟器，我们在寄存器级别实现了故障注入框架。在这个模块可以为执行器中的PE注入不同类型的瞬时、持久性故障。为了在特定周期注入一个指定类型故障，首先需要在模拟器中定位目标数据。接着需要检查目标位是否有效。如果它不包含活动值或处于空闲状态，该位就被认为是无效的。例如，在WS和IS数据流的数据替换周期中，即使部分和寄存器包含值，它们也不是活动的，因为他们将会被新的计算值所替换。此外，对于卷积核矩阵比脉动阵列尺寸小的情形，脉动阵列中未被使用到的那些寄存器内的位是空闲的。发生在无效位置上的错误不会传播到其他数据上，可视为这些错误被屏蔽。对于这些故障类型模拟器会直接跳过运算的执行，将本次故障注入记录为被屏蔽且不影响输出结果。如果目标位是有效的，故障注入过程就会正常执行。对于瞬时故障注入的实现，本模块会翻转目标位产生错误数据值，并将错误的值提供给执行模拟器进行仿真。对于持久性故障注入，模块在整个执行过程中将PE寄存器中的数据目标比特位持久固定为0或1。

（3）分析框架：负责统计saca-FI在给定的设置下的故障注入的错误率。记录实验过程信息，以及判断实验结束条件。

执行流程：如图3.3所示，执行流程可以分为三个阶段。第一个阶段为参数

生成阶段，包含了输入数据以及系统生成实验配置参数的过程。首先由用户在接口输入实验的配置信息，包括CNN参数（即CNN模型与层信息）、脉动阵列参数（即PE大小和数据流）、故障注入参数（即实验次数和故障类型信息）等等。然后配置信息传入Scale-Sim扩展模块，Scale-Sim会生成ifamp, weight与PE的映射坐标信息，同时设定参数输入会传递到故障注入模块，系统产生当次实验将要注入的故障的配置信息。即：故障在PE级（即哪个PE）、寄存器级（即输入、权重或部分和寄存器）、位级（即寄存器中的哪个比特位）和周期级（即哪个仿真执行的运算周期。只适用于瞬时故障）随机产生。对于持久性故障，在整个执行过程中，目标位置为一个寄存器内部的固定比特位上。

第二个阶段为故障注入阶段，包含了仿真执行与故障注入。执行模拟器进行脉动阵列的仿真执行。当计算到指定周期的时候将整个计算过程停止并触发故障注入，由故障注入模块辅助完成无需人为干预。故障注入结束后，运算继续执行，直到获得本次运行的结果ofmap。Ofmap回传到keras框架完成最终预测结果。

第三个阶段为结果验证阶段，包含了结果分析比较与实验信息记录。第二个阶段获得的预测结果会传入到分析框架中与无故障运行的标准结果（即没有故障注入的正常预测结果）进行比较。若产生不同的结果则表明，注入的故障导致错误的最终输出即产生了SDC错误，反之则说明故障被掩盖了。期间的故障配置与结果信息会被保存到记录文件中。

结束条件：判断是否到达指定的实验次数，如果还没有达到，则再一次进入模拟执行器重中进行计算。此时故障注入框架会为新实验再次生成新的故障注入配置信息。重复上述流程，当实验次数达到设定值则退出程序。

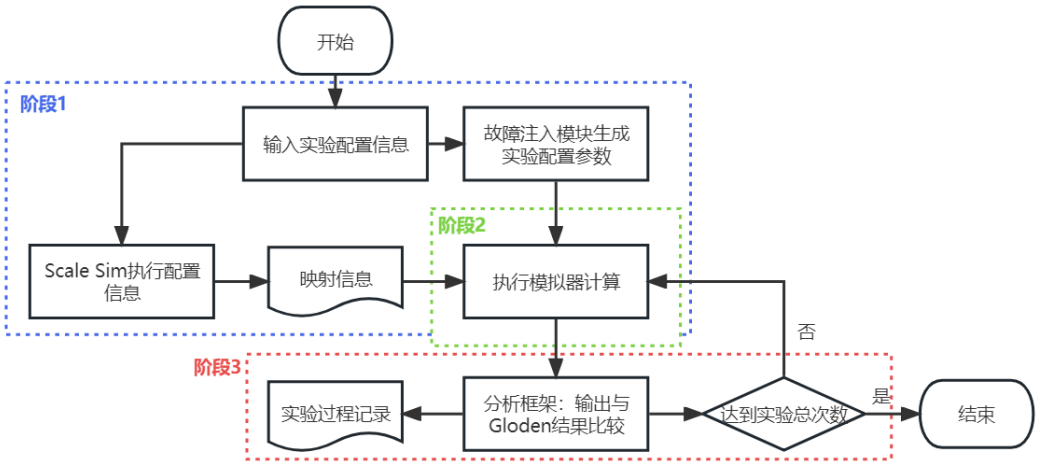


图 3.3 saca-FI 框架执行流程图

3.4 本章小节

本章首先对本框架所基于的加速器模拟器Scale-Sim与神经网络框架keras进行了基本的阐述。随后介绍了我们的故障注入工具saca-FI的实现原理，包括了数据流与计算周期的推算与计算子集的划分。在之后我们介绍了saca-FI的框架构成，以及下属的每个模块的功能和工作流程。

第 4 章 可靠性分析结果

4.1 实验设计

在本节中我们选取了三种常用的CNN模型：手写数字分类模型LeNet-5[32], 小型RGB图像分类模型CIFAR-10 CNN[14]， 大型网络模型VGG-16[33]。应用3.2节提出的saca-FI分析框架对它们的计算过程的可靠性进行分析。这三个模型的配置如下表4.1所示。LeNet-5采用MNIST[32]数据集，输入尺寸为32×32的灰度图像，分类结果为10类；第二个模型CIFAR-10 CNN，采用CIFAR-10[34]数据集，输入尺寸为32×32的RGB图像，分类结果是10个类别；第三个模型VGG-16采用了大型数据集ILSVRC-2012[35]。输入为 224×224的RGB图像，结果共计1000个不同的类别。如第2.3节表格数据所示，在本文中我们为了达到在95%置信度下的结果值1%精度误差，为每一个故障模型进行了10000次故障注入实验。

表4.1 CNN模型和数据集

CNN 模型	输入输出	数据集	层类型
LeNet-5	(28,28,1) 10 类	MNIST	CONV: 1,2; FC:5,7
CIFAR-10 CNN	(32,32,3) 10 类	CIFAR-10	CONV: 0,1,4,5; FC: 9,11
VGG-16	(224,224,3) 1000 类	ILSVRC-2012	CONV:1,2,4,5, 7,8,9, 11,12,13,15, 16,17; FC: 20,21,22

为了有效衡量故障影响程度，我们在本实验中制定了如下几个评价指标：Top1-clas：故障导致CNN模型预测出的排名为首位的对象不同于标准结果。Top1-acc：故障导致CNN模型预测出的排名为首位的对象所对应的置信度分数不同于标准结果（该类错误包含错误类型Top1-class）。Top5-clas：故障导致CNN模型预测不同的排行前五对象，排行前五的对象顺序不同也记为错误。Top5-acc：故障导致CNN模型对排行前五的元素预测的置信度得分不同（包含错误类型Top1-class、Top1-acc、Top5-class）。

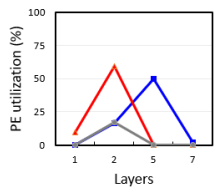
我们采用体系结构脆弱性因子（Architectural Vulnerability Factor，AVF）作

为可靠性分析的度量，它最初是由Mukherjee[36]等人最先提出的。他们通过观察故障造成错误结果（SDC、DUE）的程度来衡量处理器体系结构级别的可靠性。

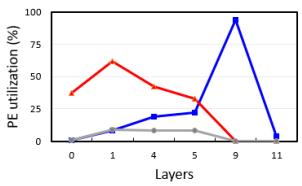
AVF值越大则反应了硬件所对应的可靠性越低，更为容易受到故障的影响。在我们的实验中，AVF的计算方法是结果错误的运行次数与实验总运行次数的比值，如式(4.1)所示。

$$\mathrm{AVF} = \frac{\text{结果错误的次数}n}{\text{故障注入总实验次数}N} \qquad (4.1)$$

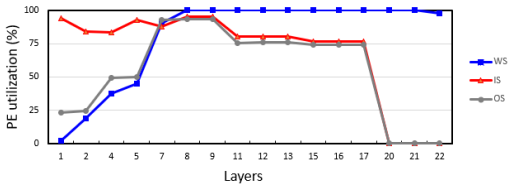
如图4.1(a)—(f)，通过Scale-Sim模拟器我们得到了LeNet-5、CIFAR-10 CNN和VGG-16模型在WS、IS和OS数据流下的各层的PE利用率和执行周期。值得注意的是，此处PE利用率表明在整个脉动阵列的执行过程中使用了多少个PE。换言之，如果一个PE在执行过程中被使用（即使只有一次），它就会被计入利用的情况。故100%的利用率代表着在一个完整的推理执行过程中，所有的PE至少被使用一次。图4.1(d)—(f)展示了三种模型在不同数据流下的各层执行周期数统计图，可以看出全连接层的执行时间长于卷积层。相较来看在三种数据流中OS数据流执行速度最快，WS次之，IS最慢。



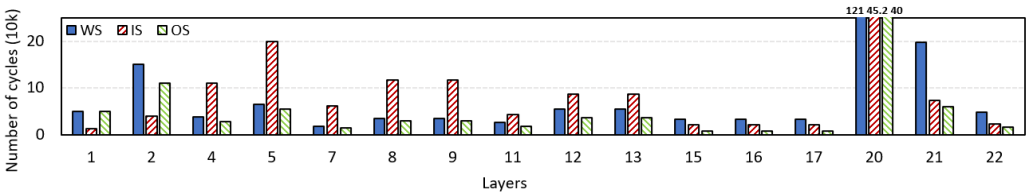
4.1(a) LeNet-5



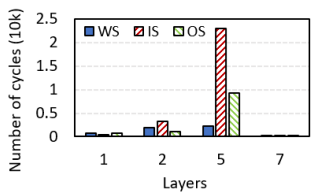
4.1(b) CIFAR-10 CNN



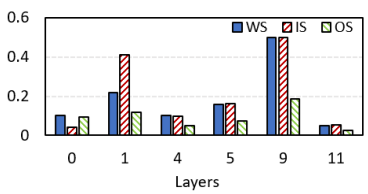
4.1(c) VGG-16PE利用率



4.1(d) VGG-16执行周期



4.1(e) LeNet-5



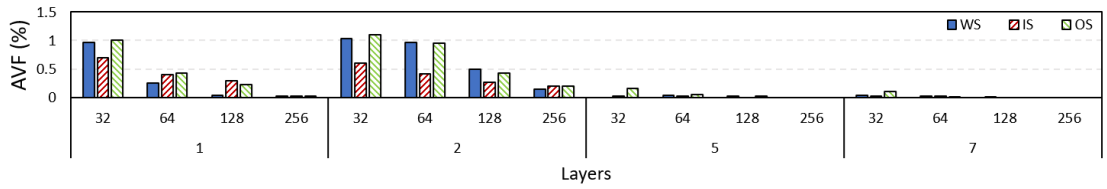
4.1(f) CIFAR-10 CNN

图 4.1 不同层对应的 PE 利用率（PE 阵列：256x256）

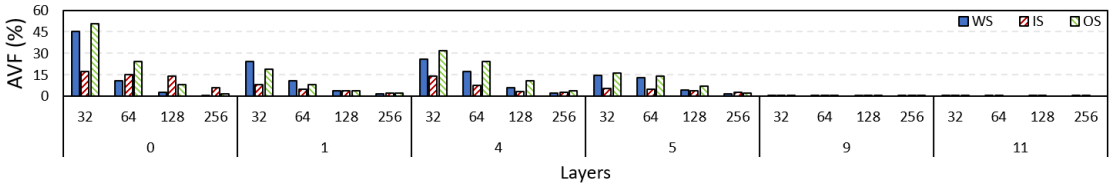
4.2 基于故障注入的脉动阵列可靠性研究

4.2.1 可靠性与模型的关系

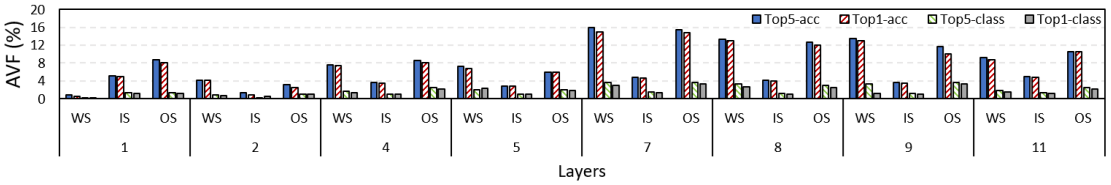
在这一节中，我们在256×256的脉动阵列中向CNN模型的每一层注入单比特瞬时故障。图4.2展示了LeNet-5、CIFAR-10 CNN和VGG-16各层在WS、IS和OS数据流下的AVF。



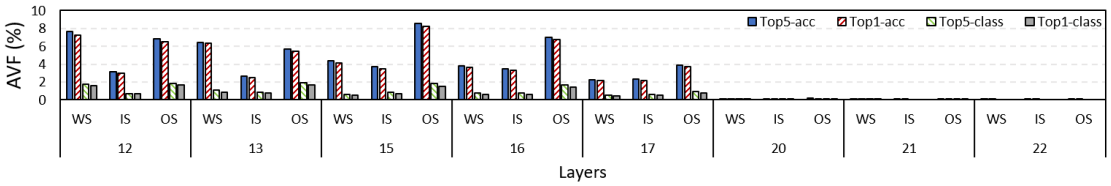
4.2(a) LeNet-5模型AVF



4.2(b) CIFAR-10模型AVF



4.2(c) VGG-16模型（1~11层）AVF



4.2(d) VGG-16模型（12~22层）AVF

图 4.2 三种数据流下模型的各层 AVF

如图4.2所示，从整体上来看在四个指标中，Top5-acc的错误率最高，其次是Top1-acc、Top5-class和Top1-class。这是因为这些指标的标准在这个顺序上严格程度逐渐提升。这表明了模型对分类错误的鲁棒性要高于精度损失的影响。由于Top5-acc指标能够更为精确的反应出模型的脆弱性特征，故我们在后续小节中使用它作为默认指标。

将图4.2中的AVF结果与图4.1中的PE利用率相比较，我们发现CONV层的AVF与PE利用率密切相关。例如，对于LeNet-5的前两层（即CONV层）在三种数据流下，第二层的PE利用率高于第一层，AVF也显示出同样的趋势。在WS、IS和OS数据流下，第一层的PE利用率分别为0.4%、9.3%和0.1%，第二层为16.7%、59.0%和17.3%。它们的AVF在第一层分别为0.02%、0.03%和0.03%，第二层为0.15%、0.20%和0.21%。VGG-16的前三层这种相关性同样成立。对于WS数据流，其PE利用率从2.6%增加到37.8%，而其AVF在这些层中从0.77%增加到7.56%。这种现象的原因是在于故障被注入到一个未使用的PE中，它就会被掩盖从而不影响最终输出。故CONV层的AVF与它的PE利用率高度相关。

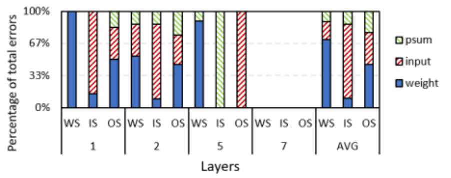
然而，如图4.2(c)和4.2(d)所示，VGG-16在OS数据流的相邻层1&2、4&5、13&15中，PE利用率和AVF之间的趋势并不符合上述分析。图中这些相邻层的PE利用率非常相似。例如，第1层和第2层的PE利用率为23.4%和24.9%。然而，第1和第2层的AVF从8.7%下降到3.1%，第4和第5层从8.5%下降到5.8%。这是是由于通道数量产生了变化，错误率随着通道数的增加而减少。第4层有64个，第5层有128个通道，而第13层有256个，第15层有512个通道。故对于这些相邻的层，生成最终结果所需的特征数量增加，这导致每个单独通道的贡献较小。所以，在单个通道的某个位置出现故障导致输出错误的概率就会降低。与第15层相比，第13层的AVF增加的原因是因为在这两层之间有一个池化层，它使每个通道的大小减少了$\frac{1}{4}$。故每个位置的贡献增加的导致错误率提高。

对于CIFAR-10 CNN模型的第0层，IS数据流的Top5-acc AVF为5.3%，远远高于其他两个数据流。这是因为在这一层中，权重作为输入被一次性重复使用了多次。第0层的通道数量很少，足以让多个输入一次性存储在脉动阵列中，这样一来，空间重用的程度就大大提高了。从图4.1(e-f)中也可以看出，IS数据流在这一层的运算速度是最快的。另外0层与下面几层相比错误率较高，这是因为输入图片直接作为这一层的Ifmap，其中零的比例较小，从而降低了错误掩盖的概率。

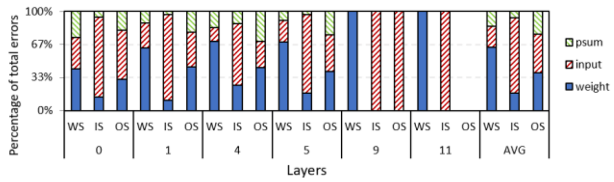
FC层通常具有非常低的错误率。在图4.2中，LeNet-5、CIFAR-10 CNN和VGG-16中FC层的AVF平均分别低于0.001%、0.01%和0.1%，远低于CONV层的错误率。对于IS和OS数据流，全连接层实际上只占用整个PE阵列的一列（或一行）， PE利用率低导致了AVF非常低。如图4.1(a-c)所示，与其他两种数据流相

比，WS数据流的FC层的PE利用率更高。例如，LeNet-5第5层PE利用率为50%，CIFAR-10 CNN第9层PE利用率为94%，VGG-16后3层平均PE利用率为99.3%。这似乎与利用率的导致错误率提升的结论相悖，实际上WS数据流虽然PE利用率高，但每个周期中计算所涉及的PE是稀疏的。由于实际输入数据只有一列，所以大多数PE在处理期间依然是空闲的，并不会导致输出错误。图4.2还显示了OS数据流的AVF高于其他两种数据流。这就需要将图4.1(a-c)与图4.1(d-f)结合起来比较。虽然OS数据流的PE利用率不高，但在三种数据流中，OS数据流计算周期数最少，计算速度最快。这是因为OS数据流不需要将数据预存储在脉动阵列中，也不包含替换周期，因此OS具有更为密集的计算强度。换句话说，OS数据流具有更为高效的数据重用模式。这会增加每项数据的复用程度，从而增加了其脆弱性。

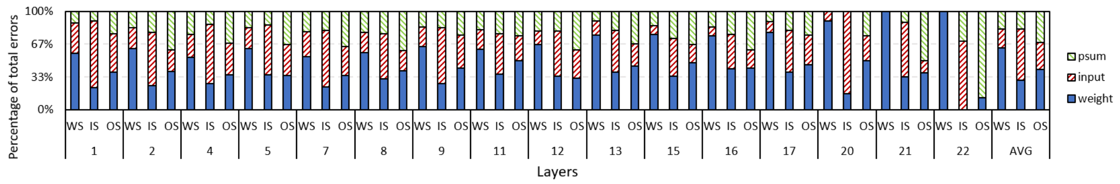
4.2.2 可靠性与寄存器类型的关系



4.3(a) LeNet-5



4.3(b) CIFAR-10 CNN



4.3(c) VGG-16

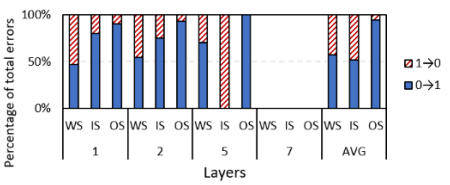
图 4.3 三种数据流下寄存器故障与各层 AVF 的关系（PE 阵列：256x256）

图4.3显示了不同类型寄存器故障引起Top5-acc错误的百分比。图中有些层不包含任何值（如图4.3(a)中三个数据流下的第7层），因为这些层的错误率非常小，在我们的实验中没有检测到错误。此外一些层只包含一种类型的值（如图4.3(b)中WS和IS数据流下的第11层），这些层错误率很低，缺乏一定统计学意义。从图4.3中可以看出，包含固定数据的寄存器故障对最终输出误差的贡献最大。例如在VGG-16模型WS数据流下，权重寄存器导致了平均总输出错误的69.5%；对于IS数据流，输入寄存器导致了平均52.9%的输出错误；对于OS数据流，部分和寄存器导致了平均36.4%的输出错误。包含固定数据的寄存器脆弱性更高的原因

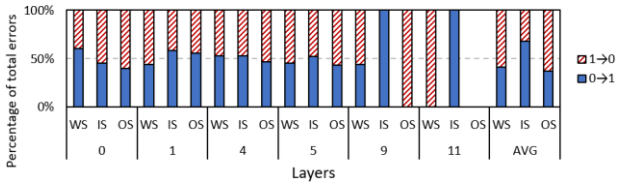
在于：存储在它们当中的固定数据与其他那些移动数据相比参与了更多的计算。此外，我们发现WS数据流下固定权重比IS数据流下固定输入特征图导致的输出错误更多。这是因为权重寄存器值被整个输出通道共享，其中的错误会导致同一通道内所有输出的错误。这些错误在计算过程中传播到后续层，并最终影响到所有PE。此外，权重通常是-1到1之间的小值，小的故障值也可能会导致较大的值变化。因为小的值在指数位中有更多的零，所以位翻转可能会较大地改变值的大小。另一方面，这一特征也可以解释输入的高可靠性。即使输入数值产生了较大的值变化，只要乘以一个较小的权重值，结果仍然被限制在一定范围内。因此与出错的权重相比，错误输入的影响不那么显著。

三种数据流下部分和寄存器故障对可靠性的影响最小，因为从卷积计算角度来说部分和的错误在整个层的计算中传播速度不如权重和输入快。在OS数据流下三种寄存器类型对输出错误的影响程度相近。这是因为OS固定了部分和寄存器中的值，而权值寄存器和输入寄存器中的数据在OS数据流下不断移动，因此它们的故障率相对较小。此外在某些层中，权重寄存器的错误率略高于部分和寄存器。这还是与部分和故障转播错误的能力相关，被固定的部分和只影响当前层的一个输出，故其的错误传播能力不如权重。

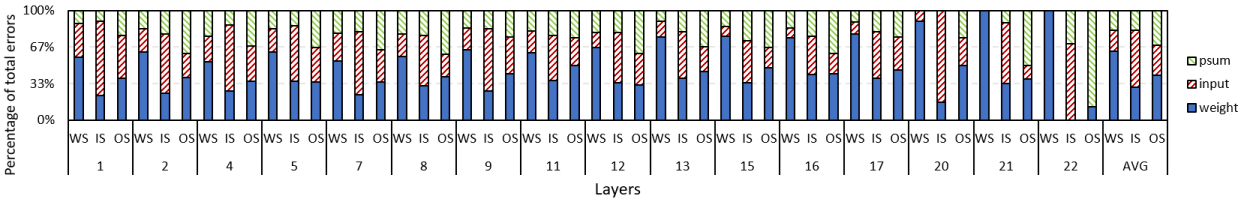
4.2.3 可靠性与比特位翻转方向的关系



4.4(a) LeNet-5



4.4(b) CIFAR-10 CNN



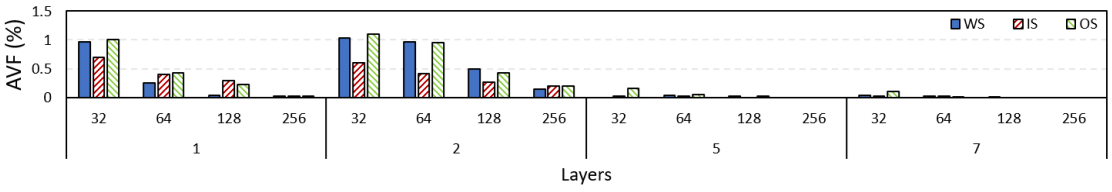
4.4(c) VGG-16

图 4.4 三种数据流下各层故障比特翻转方向（PE 阵列：256x256）

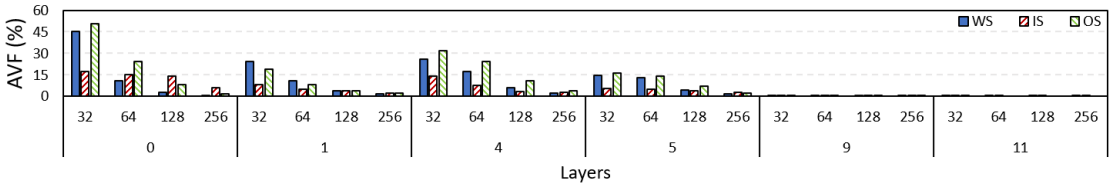
我们进一步分析了比特位翻转方向对加速器可靠性的影响。图4.4显示了三种数据流下各层0→1和1→0翻转引起错误的百分比。一般来说，CNN模型中0→1

的故障更容易导致错误。如图4.4(c)所示，VGG-16所有数据流中超过55%的输出错误是由0→1故障引起的。这意味着0→1错误比1→0错误更有可能导致错误输出。这是因为从0到1的变化会增加一个数的绝对值，而从1到0的变化则减小。增加正数的值有更大的概率抑制附近的正确结果，从而引入错误。虽然对于负数，1→0错误会增加它们的值，但ReLU函数会使得ofmap中的负值被置为0，从而掩盖错误。故从总体而言，0→1的故障更改有更高的概率导致错误。在这三种数据流中，OS数据流出现0→1故障的可能性最大。如图4.4(c)中，VGG-16在OS数据流中0→1故障的百分比平均为63.8%，而在WS数据流中平均为57.5%，在IS数据流中平均为62.8%。如图4.4(a)和图4.4(b)所示，在尺寸较小的LeNet-5和CIFAR-10 CNN中，位翻转方向导致错误的现象不明显。因为在这些模型中前一层在计算过程中累计的数据值并不大，所以他们对0→1的变化不敏感。

4.2.4 可靠性与脉动阵列尺寸的关系



4.5(a) LeNet-5 (PE阵列：32,64,128,256)



4.5(b) CIFAR-10 CNN (PE阵列：32,64,128,256)

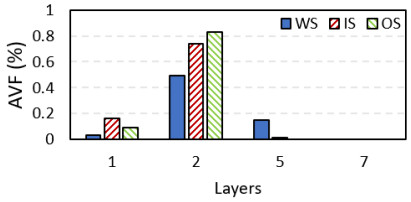
图 4.5 对三种数据流下不同脉动阵列尺寸对各层 AVF 的影响

我们进一步对脉动阵列大小对瞬时故障可靠性的影响进行了分析。图4.5为不同脉动阵列尺寸下LeNet-5和CIFAR-10 CNN的AVF。

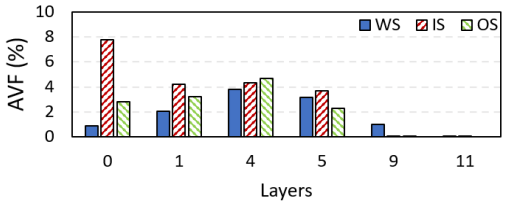
可以看到随着脉动阵列变大AVF减小。例如，LeNet-5的第二层使用WS数据流，当阵列大小为32×32时，AVF为1.08%；当阵列大小为64×64时，AVF为0.96%；当阵列大小为128×128和256×256时，AVF分别为0.54%和0.15%。这是因为具有尺寸较小的脉动阵列需要更多的执行周期来进行计算，这增加了瞬时故障的风险。此外随着脉动阵列尺寸的增加，PE利用率也会降低，从而产生更多的

空闲PE。未使用PE的故障被掩盖，不影响输出结果。我们还发现IS数据流的错误率比其他两种数据流要低。以32 × 32 脉动阵列下CIFAR-10 CNN的 1层为例，WS、IS和OS数据流的Top5-acc AVF分别为24.2%、8.1%和19.5%。这是因为对于IS数据流，它固定了输入特征图的数据，而权重在PE之间移动。由于IS数据流需要对庞大的输入特征图进行计算子集的划分，因此产生的计算子集比WS和OS多。在IS数据流中，由于输入被切分的数量较多，数据替换的频率也会增加。在数据替换过程中，会有大量PE等待数据传输。大量的空闲PE降低了整个脉动阵列的错误率。

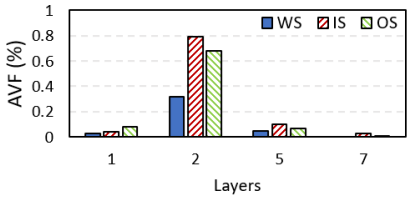
4.2.5 可靠性与其他故障类型的关系



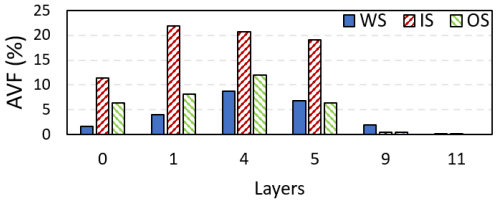
4.6(a) LeNet-5多位故障



4.6(b) CIFAR-10 CNN多位故障



4.6(c) LeNet-5多处故障



4.6(d) CIFAR-10 CNN多处故障

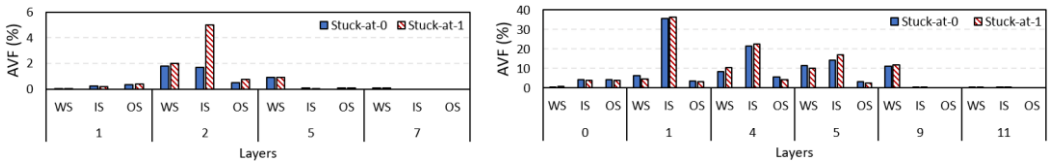
图 4.6 多处故障与多位故障（PE 阵列：256x256）

有一种罕见的瞬时故障会翻转多个比特位。这可能会导致一个数据中的多个比特位的改变或多个数据的单比特位翻转。我们进一步用saca-FI对这类故障进行建模和分析。我们随机生成2~6个多处故障和多位故障情况。图4.6给出了LeNet-5和CIFAR-10 CNN的结果。

对比图4.2(a) (b)和图4.6(a) (b)，我们发现多位和单位故障的错误率接近，这与Sangchoolie 等人[37]在基准程序上对故障注入的结论一致。这是因为多位故障和单位故障对结果的影响仅在改变值的程度上有所不同。同时因为0→1变化会增加数据的绝对值，而1→0翻转则会减少数据的绝对值。故多位故障对错误率的影响程度是有限的。此外，多处故障的错误率比多位故障高。如图4.6(b)和图4.6(d)所示， CIFAR-10 CNN模型WS数据流下，多位故障的第4层AVF为3.8%，多处故

障的AVF为8.8%。这是因为一次注入多个错误有更高的机会命中关键计算（例如，非空闲PE，非零计算），而不是那些空闲的位，从而增加了后续错误传播的可能性。此外我们发现IS数据流在多处故障情况下比WS和OS数据流的错误率高，这与PE利用率有关。如图4.1(b)所示，在前4个CONV层中IS数据流的PE利用率最高。因此注入的错误越多，错误率就越高。

4.2.6 持久性故障的可靠性



4.7(a) LeNet-5　　4.7(b) CIFAR-10 CNN

图 4.7 不同层的持久性故障（PE 阵列：256x256）

持久性故障通常以故障位置的数据持久性改变的形式发生。我们的实验涉及两种持久性硬件故障：位定位卡在0（stuck-at-0）和卡在1（stuck-at-1）。图4.7为LeNet-5和CIFAR-10 CNN在单比特持久故障下的Top5-acc AVF图。

对比图4.2可知，持久性故障下的AVF明显高于瞬时故障下的AVF。例如，IS数据流下LeNet-5 中2层的AVF在持久性卡在0和持久性卡在1时分别为1.7%和4.9%，而瞬时故障情形下仅为0.2%，错误率提高了两个数量级以上。这是因为持久性故障影响时间更长，会影响整个执行过程中映射到故障位置的所有数据。从图4.7也可以看出，卡在1处的影响略高于卡在0处的影响。原因和瞬时故障中位翻转方向是一样的，因为将位固定在1会增加任何经过该位的数据的绝对值，从而放大异常值，使特征提取异常。此外，IS数据流的AVF在三个数据流中最高，这与IS的PE利用率有关。从图4.1可以看出，IS数据流在CONV层PE利用率最高。计算中需要的PE越多，使用带有持久性故障的PE的可能性就会越大。

4.3 建模分析与故障注入的对比

在本节中我们使用了一个现有的体系结构可靠性评价指标saca-AVF[38]，它通过ACE位分析法[36]来为脉动阵列进行可靠性建模，能够快速便捷地度量CNN加速器受故障影响程度。该模型考虑了所有体系结构级别的ACE位在整个脉动阵

列执行过程之中所占据的比例。这个比例越高则说明体系结构的脆弱性因子越高。它的计算方法如下：

$$\text{saca-AVF} = \frac{\sum_0^{\text{cyc}} \sum_0^{\text{PEs}} \left(\text{输入ACE位} n_1 + \text{权重ACE位} n_2 + \text{部分和ACE位} n_3\right)}{\left(\text{输入位} N_1 + \text{权重位} N_2 + \text{部分和位} N_3\right) \times \text{脉动阵列尺寸} S_{PEs} \times \text{周期cyc}} \quad (4.2)$$

上式（4.2）中，分子代表了整个执行周期中所有PE中存储输入、权重和部分和的寄存器中的总ACE位数。分母代表了PE寄存器的总位数同脉动阵列尺寸、执行总周期的乘积。

我们在脉动阵列的PE寄存器中随机注入单个位翻转故障。在每次实验为每一层执行过程中平均注入3000个错误。加速器的存储使用了Scale-Sim设置的默认大小。表4.2显示了256×256脉动阵列中saca-AVF与故障注入AVF之间的Pearson相关系数。所有实验的平均相关系数为0.94，表明saca-AVF与saca-FI高度相关，可用于CNN加速器架构的有效可靠性估计。LeNet-5、Cifar-10 CNN和VGG-16模型在不同数据流下的平均相关性分别为0.99、0.92和0.96。

表4.2 Saca-AVF与Saca-FI结果的相关性系数

	LeNet-5	Cifar10 CNN	VGG16
WS	0.99914	0.94601	0.98812
IS	0.99248	0.90462	0.92531
OS	0.99513	0.92153	0.96834

我们进一步比较了saca-AVF和故障注入下的AVF值。我们观察到并非浮点数中的所有比特位都会引起错误，例如32位浮点数中的低尾数位部分。错误常常是那些由指数位发生比特翻转引起的。因为指数位的变化能够使得数值产生非常大幅度的跳跃，而尾数位并不能做到这一点，尾数位只是改变了值的精确程度。图4.8给出了故障注入框架saca-FI得到的VGG-16模型在WS数据流下，采用IEEE 754单精度浮点数据时不同位故障引起错误的分布。如图所示，输出错误都是由符号位（即第31位）、指数位（即第23-30位）和几个尾数位（即第15-22位）的故障引起的，而其他位的故障是被掩盖的。

为了考虑比特位对AVF不均匀的影响程度，我们引入了数据表示因子（data representation factor，DRF）。DRF-adjusted saca-AVF是DRF和saca-AVF的乘积。根据图4.8，我们发现9位以上的误差占比超过85%，因此我们选择$\frac{9}{32}$为作为本例

的DRF，使得比特位对实际故障产生的贡献的评估程度更接近于实际情况。

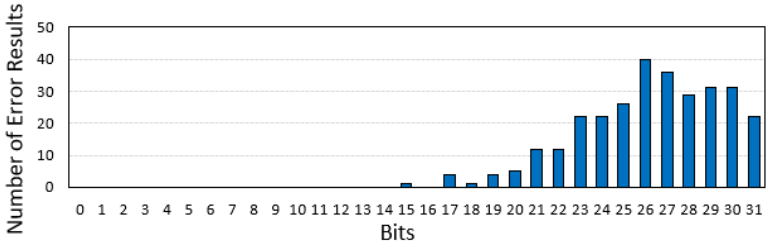


图 4.8 不同位的故障导致输出错误的分布图（VGG-16 7 层）

图4.9为VGG-16的saca-FI AVF与DRF saca-AVF的分层比较结果，WS的平均AVF分别为5.91%和6.27%，IS的平均AVF分别为2.85%和3.55%，OS的平均AVF分别为6.03%和6.81%，说明DRF saca-AVF与故障注入时的AVF结果非常接近。特定CNN模型的DRF可以通过少量的故障注入运行来获得。DRF仅依赖于CNN模型的数据表示特征，在特定CNN模型的不同加速器架构之间是固定的。DRF saca-AVF仅用于比较不同CNN模型的AVF。此外，saca-AVF与故障注入相比速度更快。这是因为对于相同的输入，saca-AVF只执行一次CNN加速器的计算过程，得到ACE位并计算AVF结果。相比之下，saca-FI通常需要数千次故障注入才能达到置信区间。每一次故障注入运行都执行了CNN加速器的整个计算过程。而saca-FI的准确度更高，可以作为建模分析的支持。

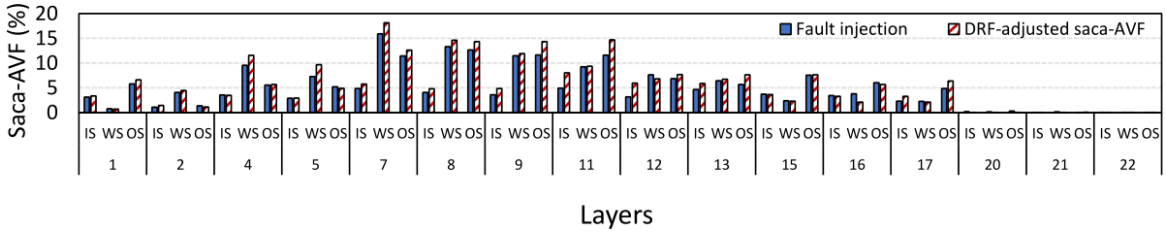


图 4.9 加权 Saca-AVF 与 Saca-FI AVF 的比较

4.4 本章小节

接着第三章的工作，我们选取了三种常见的CNN模型进行实验，在脉动阵列模拟器框架saca-FI上研究了瞬时故障、多位故障、多处故障以及持久性故障影响情况。我们利用单比特翻转来模拟瞬时故障，将寄存器中比特位永久置为0或1来模拟持久性故障导致的卡在0或1的持久故障。接着从不同模型、寄存器类型、翻转方向、PE阵列尺寸等角度分析了脉动阵列加速器的可靠性特征。我们发现OS

数据流具有最快的运行速度，但相较于WS, IS两个数据流来说，它的可靠性最低；权重寄存器，输入寄存器可靠性较低，而部分和寄存器的可靠性更高；脉动阵列尺寸的增加能够在一定程度上提升计算的速度以及可靠性。

随后我们使用分析模型saca-AVF获得的结果与故障注入结果进行了比较，发现两者具有平均高达0.94的相关系数。对于两者AVF存在的数值结果差异，我们依据比特的重要性程度为saca-AVF赋予了一个调整系数DRF，使得saca-FI故障注入结果能够直接与saca-AVF进行相互比较。这两种工作可以相互结合，各取所长。

第 5 章 卷积神经网络加速器可靠性提升方法

5.1 可靠性提升的权衡

前两章我们通过故障注入与建模分析的方法综合分析了脉动阵列加速器对于瞬时故障和持久故障的可靠性特征。但实际上的硬件优化并不是单一指标的，往往要考虑到综合性能提升以及成本之间的权衡。故在本节中我们提出了一种性能-能量-可靠性-面积（PERA）的指标来量化分析这四个指标的组合效果，为加速器设计者提供系统可靠性提升的见解与思路。

5.1.1 综合分析指标PERA

只关注可靠性可能会误导CNN加速器架构的设计。性能、能量和面积是计算机架构师在设计处理器时考虑的三个主要指标。因此在CNN加速器架构设计中，将可靠性与这些指标一起考虑是很重要的。

随着脉动阵列尺寸增加，其执行时间和AVF均降低。这是因为当一个脉动阵列包含更多的PE时，投入更多的PE能够使计算速度变得更快。空闲PE的增多也使得AVF在一定程度上所有减少。同时能耗也会逐渐降低，因为当脉动阵列中PE数量增加时，数据重用增加便会减少DRAM、全局缓存访问次数从而降低访问能耗。然而这些度量的变化程度是不一样的，所以我们需要一个通用的度量来合作地分析它们。我们定义了一个称为性能-能量-可靠性-面积的指标来量化这四个指标的组合效果。PERA的计算方法如公式（5.1）将执行周期Cyc、能耗E_{data}、AVF和脉动阵列的面积S_{PEs}相乘。较低的PERA对于CNN加速器架构是可取的。

$$\mathrm{PERA} = Cyc \times E_{data} \times AVF \times S_{PEs} \quad \cdots\cdots\cdots\cdots \quad (5.1)$$

能耗E_{data}计算依据Haichuan Yang 等人[39]的计算方法，以及文献Eyeriss [7]中的统计指标。脉动阵列的片上能耗E_{data}如公式（5.2）所示，考虑了计算能耗和不同存储器层次的能耗。计算能耗即为MAC计算的能耗，这部分能耗只和卷积核与输入特征图像大小相关。存储的能耗可以为四个部分：DRAM读取的能耗，缓冲区数据读取能耗，PE间数据传递能耗，以及寄存器访问能耗。这些部分能

耗的计算方式如公式（5.3）所示。其中的N分别代表了数据访问对应类型存储器的次数，如N_{DRAM}^{input}代表了获取输入数据需要访问DRAM的次数。参数e代表了访问对应级别存储的能耗，如e_{DRAM}代表了访问DRAM的能耗。

$$E_{data} = E_{MAC} + E_{storage} \quad \cdots\cdots\cdots\cdots \quad (5.2)$$

$$E_{storage} = e_{DRAM}\left(N_{DRAM}^{input} + N_{DRAM}^{weight}\right) + e_{Buffer}\left(N_{Buffer}^{input} + N_{Buffer}^{weight}\right) + e_{Array}\left(N_{PE}^{input} + N_{PE}^{weight} + N_{PE}^{psum}\right) + e_{RF}\left(N_{RF}^{input} + N_{RF}^{weight} + N_{RF}^{psum}\right) \quad (5.3)$$

对于能耗e如下表5.1，不同级别的存储能耗被归一化为MAC能耗[7]。可以看到一次DRAM的数据访问操作的能耗相当于一次MAC操作的200倍，PE间数据传递的能耗仅为全局缓存访问所花费能耗的$\frac{1}{3}$。这也说明了多级存储以及脉动阵列的数据共享策略都能有效降低片上数据访问能耗。

表 5.1 归一化能耗度量

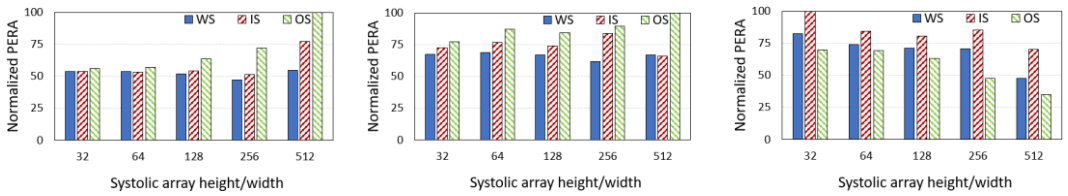
	DRAM	全局缓冲(Buffer)	PE 间传递	寄存器 RF
能耗	200×	6×	2×	1×

5.1.2 实验分析

图5.1显示了三种CNN模型的PERA结果。结果归一化到给定CNN模型的最高PERA值。对于LeNet-5和Cifar-10 CNN，通过OS数据流映射，将结果归一化为512 × 512的脉动阵列。对于VGG-16模型，则利用IS数据流映射将结果归一化为32 × 32脉动阵列。

如图所示，三种模型的PERA变化趋势存在显著差异。对于小型模型（如LeNet-5和CIFAR-10 CNN）， OS数据流下的PERA随着脉动阵列尺寸的增大而增大，WS下的PERA先减小后增大。IS下的PERA随脉动阵列大小的增加而波动。在WS下256×256脉动压阵列Lenet-5和CIFAR-10 CNN的PERA最低。对于大模型(VGG-16)， WS和OS下PERAs随脉动阵列尺寸的增大而减小，而IS下的趋势略有波动。OS下的512×512脉动压阵列PERA最低。这些结果表明，OS在大型CNN模型中表现最好，而WS在小型CNN模型中表现最好。这是因为与其他两种CNN模型相比，VGG-16在OS下的执行速度要比WS快得多。这样，与性能相比，OS下

的错误率增加程度比WS下更高。因此性能和可靠性的综合作用导致了相反的趋势。例如在512 × 512脉动阵列下，VGG-16 OS的AVF比WS高1.73倍，LeNet-5中这个倍率为3.83。VGG-16的OS执行时间为WS的0.44倍， LeNet-5中二者执行时间的倍率为0.48。因此执行时间和AVF对OS的综合影响在VGG-16中低于WS (0.74倍)，而在LeNet-5中高于WS (1.85倍)。正如这个例子，即使单个指标的趋势保持不变，多个指标的综合效果可能是不同的。因此，硬件设计人员需要对这些指标进行合作分析，以找到最佳的设计选择。



5.1(a) LeNet-5　　5.1(b) CIFAR-10 CNN　　5.1(c) VGG-16

图 5.1 加速器综合指标 PERA 分析

5.2 基于 ECC 保护的可靠性提升方法

通过纠错码来保护容易受到影响的位是计算机系统常常采取的方法[40][41]。其中通过ECC检验来保护系统中易受故障影响或损坏的位能够有效提升系统的可靠性。本节采用单错误校正和双错误检测（SEC-DED）ECC技术来保护脉动阵列中那些计算关键的位来提升系统整体的可靠性。如我们在4.3节中观察的那样，并不是所有的参与计算的位都能导致Top-acc错误，反之是那些指数部分的位能比较大的程度对结果造成影响。这是因为翻转这些指数位可能会使值变化很大。数值范围越大，偏离正常结果的可能性就越大。

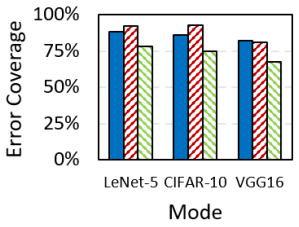
由图4.8我们还观察到在指数位附近还有少量的位同样也能引起Top-acc错误。借由SEC-DED纠错码的保护特性，通过添加一位纠错码能够增加对处于指数位附近的比特位的保护。保护过程：在数据预存储阶段脉对当前固定数据（weight—WS或是ifmap—IS）的第21位到30位生成校验码。在计算阶段，在每次的MAC操作之前，将当前层的计算数据受到校验码保护的21位到30位数据进行校验。如果没有出错，则可以正常进行计算，反之出错就会对错误数据进行修正。如果是一位出错则可以直接改正错误进行正常计算。如果是两位以上的错误

则被判定为无法矫正错误，应当终止当前计算重新执行。以下我们将展示如何使用saca-FI框架观察到的可靠性特征来实现错误保护。针对脉动阵列加速器架构，我们提出了两种错误保护机制，均可以以较小的开销实现良好的错误覆盖。

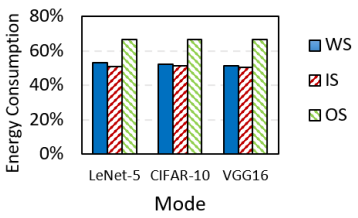
5.2.1 保护更为脆弱的寄存器

在第4.2.2节中，我们发现在权重寄存器和输入寄存器中发生的错误对脉动阵列CNN加速器的可靠性影响程度要大于部分和寄存器。根据这一特征，我们使用SEC-DED ECC来保护PE中权重寄存器和输入寄存器的数据。保护策略的有效性如图5.2(a)所示，与完全保护（即用SEC-DED ECC保护所有三种类型的寄存器）相比，三种模型的WS和IS数据流的平均错误覆盖率分别为85.4%和88.9%。这三种寄存器在OS数据流下都有相近的可靠性，保护权重和输入寄存器的错误覆盖率仅为73.6%。这是因为在当前层的计算过程中，部分和寄存器中发生的错误不会直接影响其他PE。它会在下一层执行期间开始将错误作为输入传播。

评估错误保护的能量消耗时，我们只考虑ECC能量，因为在CNN加速器中其余部分保持不变。ECC能量的计算方法分别为SEC-DED ECC的异或操作次数乘以生成过程和检查过程的次数。写入寄存器时产生ECC，读取寄存器时检查ECC。图5.2显示了归一化能耗，与完全保护相比，WS为52.2%，IS为50.9%，OS为66.7%。这表明在这些模型中保护权重和输入寄存器具有更多收益。



5.2(a) 错误覆盖率



5.2(b) 归一化能耗

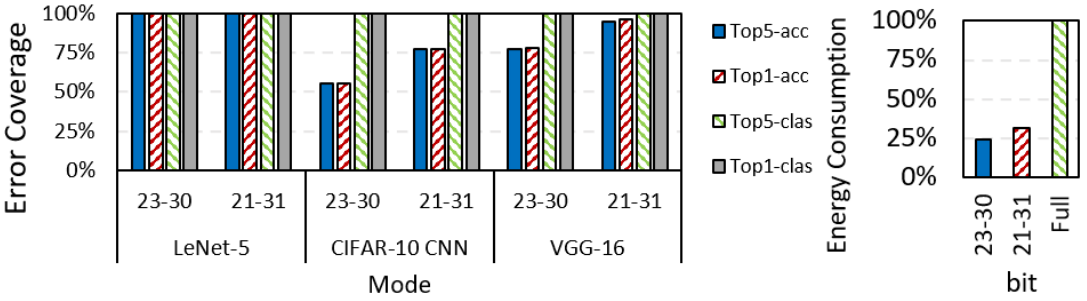
图 5.2 保护更为脆弱的寄存器

5.2.2 保护更为脆弱的比特位

在第4.3节，我们观察到故障发生在指数位（即23-30位）在大多数情况下会导致分类错误。不同于5.2.1节的方案，我们需要为PE中全部类型的寄存器数据

保护指定的比特位区间，同样使用校验SEC-DED ECC校验码来进行校验与保护。从图5.3中可以看出，对三种模型的指数进行保护可以保证100%的正确分类(Top-clas)，LeNet-5可以保证100%的准确率（Top-acc）， CIFAR-10 CNN可以保证55.6%的准确率，VGG-16可以保证77.3%的准确率。

由于在32位数据表示中有8个指数，因此保护它们需要5个SEC-DED ECC位。而5位的SEC-DED ECC最多可以保护11位。为了充分利用纠错码的保护能力，我们将其扩展为保护11位（21-31），通过增加对符号位和两个高位尾数位的保护。这样CIFAR-10 CNN和VGG-16的Top5-acc分别提高到77.2%和94.9%。图5.3(b)为这三种保护方式的归一化能耗，与完全保护相比保护指数位的能耗为24.1%，保护21-31位的能耗为31.5%。这表明了，当只考虑分类结果时，只保护指数位是最节能的方法；当考虑分类精度时，增加符号位和尾数位高位的保护会带来很好的分类精度提升。



5.3(a) 错误覆盖率　　5.3(b) 归一化能耗

图 5.3 保护更为脆弱的比特位

5.3 本章小节

首先我们综合分析CNN加速器性能、能量、可靠性与面积的指标PERA并提出了权衡的见解。随后我们提出了两种用ECC技术保护脉动阵列加速器来提升推理可靠性的方法：1）通过实验观察我们发现发生在浮点数指数部分的故障对加速器的推理精度影响最大，因此采用ECC校验技术保护PE寄存器中的所有数据的指数部分来提升可靠性。2）由于加速器中的权重寄存器以及输入寄存器对于推理精度影响最大，故我们提出了保护这两种类型的寄存器来提升脉动阵列加速器的可靠性方案。我们进一步分析了两种方案的能耗、错误覆盖情况。

通过实验我们得出同时保护权重寄存器和输入寄存器可以在WS数据流模式下达到85.4%和88.9%的错误覆盖率。对于OS数据流来说这种保护策略优势较小。这样保护的平均开销为完全保护开销的56.6%。保护更容易受到错误影响的指数位可以为三类模型带来100%的Top5-clas分类精度。对于Top5-acc的分类精度来说能达到平均77.63%的错误覆盖率。而将保护范围扩展到符号位和高尾数位能够使得错误覆盖率提升到平均90.7%。这样的保护策略的能耗仅占全保护策略的31.3%。实验表明上述思路是可以以较低的能耗开销来有效降低脉动阵列加速器的误分类率，提高推理结果的精度。

第 6 章 总结与展望

6.1 工作总结

深度学习的兴起带动了卷积神经网络、深度学习应用程序在大型服务器、边缘配套设备的广泛应用与部署。这为设备的算力、面积、能耗开销等提出了新的需求。CNN加速器设计的复杂性来自于需要同时处理高维卷积中的数百个卷积核和通道，这涉及大量的数据移动。找到一个支持并行处理的数据流，以最小的数据移动成本，在不影响准确性的情况下实现节能CNN处理至关重要。除此之外当前的CNN加速器设计依然缺乏足够的可靠性支持，这阻碍了CNNs等相关技术推广的同时也埋下了诸多隐患。因此本文立足于一种通用的加速器设计模式——脉动阵列构建了微体系结构可靠性分析框架saca-FI。根据saca-FI的可靠性分析，能够帮助设计者发掘出脉动阵列内部所隐含的体系结构级别的可靠性特性。此外我们还根据可靠性分析结果提出了相应的改进策略。

利用周期精确的微体系结构级别的脉动阵列加速器分析框架saca-FI，我们首先发现OS数据流比其他两个数据流更脆弱。卷积层的可靠性与PE利用率密切相关，而全连接层的可靠性通常较高。随后我们比较了寄存器类型的可靠性特征，发现在WS和IS数据流下，包含固定数据的寄存器的错误率最高，而在OS数据流下，三种寄存器的可靠性相似。然后我们分析了位翻转方向的影响，发现0→1故障导致了所有模型输出误差的平均55%以上。我们进一步研究了故障比特位置的影响，并观察到故障比特位所在位置对错误输出的贡献不均匀。然后，我们探讨了脉动阵列尺寸的影响，我们发现错误率随着脉动阵列尺寸的减小而增加。之后我们发现错误率随着错误数量的增加（多个错误）以及错误影响的持续时间增加（持久性故障）而显著增加。之后我们使用了一种灵活高效的体系结构脆弱因子分析模型saca-AVF，将这它的预测结果与故障注入AVF进行了对比分析，结果表明两者具有高度一致性的结论。我们还依据比特位的重要程度提出了添加调整系数的方法，使得两个工具的AVF能够进行数值比较。

为了提升脉动阵列加速器的可靠性，我们提出了一个综合分析指标PERA，对脉动阵列的性能-能量-可靠性-面积等设计要素进行了分析与权衡。随后采用

SEC-DED校验码来保护更容易受到错误影响的寄存器位。并提出了两种保护策略：1）保护脆弱性更高类型的寄存器（权重寄存器和输入寄存器）。2）保护更容易受到故障影响的数据比特位（浮点数的指数位）。通过实验证明，这两种保护方法均在可接受的能耗开销下有效提升了神经网络加速器的分类结果精度以及分类准确性。

6.2 工作展望

本文基于我们提出的周期精确的微体系结构级别故障注入框架saca-FI，对脉动阵列的可靠性进行了综合性的分析。此外，基于实验中的观测结论，结合ECC技术提出了提升加速器推理精度与正确性的方法。在接下来的研究中，我们会进一步扩展故障注入框架saca-FI，使它能够支持更多的数据类型、模型类型，以及不同的数据流。并继续探究提升脉动阵列加速器应对瞬时故障与持久性故障的可靠性提升策略。

参考文献

[1] SZE V, CHEN Y H, YANG T J, et al. Efficient Processing of Deep Neural Networks: A Tutorial and Survey[J]. Proceedings of the IEEE, 2017, 105(12).

[2] K. Simonyan, A. Zisserman. Very deep convolutional networks for large scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[3] JIANG P, FU H, TAO H, et al. Parallelized Convolutional Recurrent Neural Network with Spectral Features for Speech Emotion Recognition[J]. IEEE Access, 2019, 7.

[4] GONG Y, XIAO Z, TAN X, et al. Context-Aware Convolutional Neural Network for Object Detection in VHR Remote Sensing Imagery[J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(1).

[5] N. P. Jouppi, C. Young, N. Patil, et al. In-datacenter performance analysis of a tensor processing unit[C]. ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), 2017, pp. 1–12.

[6] N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, et al. Ten lessons from three generations shaped google’s tpuv4i[C]. ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021, pp. 1–14.

[7] Chen Y H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks[J]. ACM SIGARCH computer architecture news, 2016, 44(3): 367-379.

[8] S. S. Banerjee, S. Jha, J. Cyriac, et al. Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data[C]. 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018, pp. 586– 597.

[9] K. D. Julian, J. Lopez, J. S. Brush, et al. Policy compression for aircraft collision avoidance systems[C]. IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016, pp. 1–10.

[10] R. Baumann. Radiation-induced soft errors in advanced semiconductor technologies[J]. IEEE Transactions on Device and Materials Reliability 5 (3) 2005, 305–316.

[11] Li G, Hari S K S, Sullivan M, et al. Understanding error propagation in Deep Learning

Neural Network (DNN) accelerators and applications[C]. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2017.

[12] C. Schorn, A. Guntoro, G. Ascheid. Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators[C]. Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 979–984.

[13] C. Schorn, A. Guntoro, G. Ascheid. An efficient bit-flip resilience optimization method for deep neural networks[C]. in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2019, pp. 1507–1512.

[14] L. Ping, J. Tan, K. Yan. Sern: Modeling and analyzing the soft error reliability of convolutional neural networks[C]. Proceedings of the 2020 on Great Lakes Symposium on VLSI, GLSVLSI '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 445-450.

[15] M.-C. Hsueh, T. Tsai, R. Iyer. Fault injection techniques and tools[J]. Computer 30 (4) 1997, 75–82.

[16] Z. Chen, N. Narayanan, B. Fang, et al. Tensorfi: A flexible fault injection framework for tensorflow applications[C]. IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2020, pp. 426–435.

[17] F. F. dos Santos, S. K. S. Hari, P. M. Basso, et al. Demystifying gpu reliability: comparing and combining beam experiments, fault simulation, and profiling[C]. IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2021, pp. 289–298.

[18] F. Benevenuti, F. Libano, V. Pouget, et al. Comparative analysis of inference errors in a neural network implemented in sram-based fpga induced by neutron irradiation and fault injection methods[C]. 31st Symposium on Integrated Circuits and Systems Design (SBCCI), 2018, pp. 1–6.

[19] R. L. Rech Junior, S. Malde, C. Cazzaniga, et al. High energy and thermal neutron sensitivity of google tensor processing units[J]. IEEE Transactions on Nuclear Science 69 (3) 2022, 567–575.

[20] R. L. Rech, P. Rech. Reliability of google's tensor processing units for embedded applications[C]. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022, pp. 376–381.

[21] P. Pandey, P. Basu, K. Chakraborty, et al. Greentpu: Improving timing error resilience of a near-threshold tensor processing unit[C]. 56th ACM/IEEE Design Automation Conference (DAC), IEEE, 2019, pp. 1–6.

[22] J. J. Zhang, T. Gu, K. Basu, S. Garg. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator[C]. IEEE 36th VLSI Test Symposium (VTS), IEEE, 2018, pp. 1–6.

[23] J. J. Zhang, K. Basu, S. Garg. Fault-tolerant systolic array based accelerators for deep neural network execution[J]. IEEE Design & Test 36 (5) 2019, 44–53.

[24] K. Cho, I. Lee, H. Lim, et al. Efficient systolic-array redundancy architecture for offline/online repair[J]. Electronics 9 (2) 2020.

[25] B. Reagen, U. Gupta, L. Pentecost, et al. Ares: A framework for quantifying the resilience of deep neural networks[C]. 55th ACM/ESDA/IEEE Design Automation Conference (DAC), IEEE, 2018, pp. 1–6.

[26] G. Papadimitriou, D. Gizopoulos. Demystifying the system vulnerability stack: Transient fault effects across the layers[C]. ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021, pp. 902–915.

[27] S. Albawi, T. A. Mohammed, S. Al-Zawi. Understanding of a convolutional neural network[C]. International Conference on Engineering and Technology (ICET), IEEE, 2017, pp. 1–6.

[28] A. Samajdar, J. M. Joseph, Y. Zhu, et al. A systematic methodology for characterizing scalability of dnn accelerators using Scale Sim[C]. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2020, pp. 58– 68.

[29] S. Mittal. A survey on modeling and improving reliability of dnn algorithms and accelerators[J]. Journal of Systems Architecture 104 2020, 101689.

[30] J. Wei, A. Thomas, G. Li, K. Pattabiraman. Quantifying the accuracy of high-level fault injection techniques for hardware faults[C]. 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014, pp. 375–382.

[31] R. Leveugle, A. Calvez, P. Maistri, et al. Statistical fault injection: Quantified error and confidence[C]. Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 502–506.

[32] Y. Lecun, L. Bottou, Y. Bengio, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE 86 (11) 1998, 2278–2324.

[33] K. Simonyan, A. Zisserman. Very deep convolutional networks for largescale image recognition[J]. arXiv preprint arXiv:1409.1556 2014.

[34] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images[J]. Tech. rep. Citeseer 2009.

[35] O. Russakovsky, J. Deng, H. Su, et al. Imagenet large scale visual recognition challenge[J]. International journal of computer vision 115 (3) 2015, 211–252.

[36] MUKHERJEE S S, WEAVER C, EMER J, et al. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor [C]. Proceedings of the Annual International Symposium on Microarchitecture, MICRO.

[37] B. Sangchoolie, K. Pattabiraman, J. Karlsson. An empirical study of the impact of single and multiple bit-flip errors in programs[J]. IEEE Transactions on Dependable and Secure Computing 19 (3) 2022, 1988–2006.

[38] 平丽琪. 面向卷积神经网络的软错误可靠性分析及提升研究[D]. 吉林大学, 2021.

[39] Yang H, Zhu Y, Liu J. Energy-constrained compression for deep neural networks via weighted sparse projection and layer input masking[J]. arXiv preprint arXiv:1806.04321, 2018.

[40] Alameldeen A R, Wagner I, Chishti Z, et al. Energy-efficient cache design using variable-strength error-correcting codes[C]. Proceedings - International Symposium on Computer Architecture.

[41] C. Lunardi, F. Previlon, D. Kaeli, et al. On the Efficacy of ECC and the Benefits of FinFET Transistor Layout for GPU Reliability[J]. IEEE Transactions on Nuclear Science, vol. 65, no. 8, pp. 1843–1850, 2018.