分类号： TN4 单位代码： 10335

学 号： 21931061



硕士学位论文



**中文论文题目：** **面向语音关键词识别的低功耗 神经网络处理器**

**英文论文题目： Low Power Neural Network Processor**

**for Keywords Spotting**

申请人姓名： 杨树园

指导教师： 沈海斌 教授

合作导师：

专业名称： 电子科学与技术

研究方向： 智能系统与芯片

所在学院： 信息与电子工程学院

**论文提交日期 2022年1月**

**面向语音关键词识别的低功耗神经网络处理器**

****

**论文作者签名:**

**指导教师签名:**

论文评阅人1：

评阅人2：

评阅人3：

评阅人4：

评阅人5：

答辩委员会主席：

委员1：

委员2：

委员3：

委员4：

委员5：

答辩日期：

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名： 签字日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解  **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名： 导师签名：

签字日期： 年 月 日 签字日期： 年 月 日

# 摘要

人工智能物联网（Artificial Intelligence of Things，AIoT）在终端物联网设备上结合了人工智能算法，使设备更加智能，提升了用户和设备之间的交互体验。在物联网应用中，存在大量使用神经网络进行序列信号处理的任务，由于物联网设备通常由电池供电，因此对功耗具有严格的要求，此外准确度直接反映了设备的响应能力，延时会影响用户和设备交互的速度，因此高准确度和低延时对于好的用户体验来说十分重要。存内计算技术在存储器中嵌入了计算的功能，大大减少了存储器访问的次数，具有高能效和高吞吐量的优势，非常适合用来实现神经网络的运行。但存内计算技术由于权重固定等特性，存在不够灵活的缺点，如何基于存内计算实现相对灵活的神经网络处理器架构还是一个挑战。

文章首先设计了面向神经网络量化模型的动态缩放方法，充分利用低位宽ADC（[Analog-Digital](javascript:;) [Converter](javascript:;)）的量程，实现了宽范围的动态缩放因子调整，提高了存内计算核执行神经网络推理计算的准确性。然后文章设计了基于存内计算的流式处理器架构，并设计了专用指令集，处理器可以通过配置指令运行时序卷积残差网络（Temporal Convolutional Residual Neural network, TC-Resnet）类的神经网络，实现包含语音关键词识别在内的序列信号处理任务。最后文章提出了针对TC-Resnet模型的流式预测的方法，并设计了相应的硬件执行方法，充分利用数据重用，减少了90%以上的计算量。

本文采用40nm工艺进行了电路仿真实验，实验结果显示当运行语音关键词识别任务时，可识别10个关键词，准确率达94.1%-94.4%，功耗为1.6-2.31uw，延时为172.8-184.8us，与其他达到相近识别效果的工作相比，功耗降低了1.6-12.2倍。

**关键词**：低功耗，高准确度，流式预测，动态缩放，神经网络加速器，存内计算

# Abstract

Artificial Intelligence of Things (AIoT) enables artificial intelligence algorithms on IoT devices, making the devices smarter and improving the interactive experience between users and devices. Neural networks algorithms are widely used for sequential signal processing in AIoT applications. IoT devices are usually battery-powered, which imposes strict requirements on power consumption. In addition, accuracy directly reflects the responsiveness of the device. Delay will affect the speed of interaction between the user and the device. Therefore, high accuracy and low delay are very crucial for a good user experience. The IMC(In-Memory Computing) technology embeds computing capabilities in the memory, which greatly reduces the number of memory accesses. IMC technology achieves high energy efficiency and high throughput, and is very suitable for running neural network algorithms. However , the IMC technology is not flexible enough due to the characteristic of stationary weights. To implement a relatively flexible neural network processor architecture based on IMC is still a challenge.

The dissertation first proposes a dynamic scaling method for quantized neural network models. The method makes full use of the range of ADC and achieves a wide range of dynamic scaling factor, which greatly improves the accuracy of inference based the IMC core. Then, a streaming processor architecture based on IMC is proposed, which employs a centralized IMC core with high energy efficiency and low leakage. A customized instruction set is also proposed, to make the proseccor programmable for different tasks including keywords spotting by running TC-Resnet. Finally, the dissertation proposes a streaming inference method for TC-Resnet, and design the corresponding hardware execution method to reuse data, which reduces the amount of calculation by more than 90%.

The processor can identify 10 keywords with an accuracy of 94.1%-94.4% when running keyword spotting(KWS) task, and measured results in 40nm CMOS process shows it achieves 1.6-2.31uw low power consumption and 172.8-184.8us latency per decision. The power consumption are 1.6-to-12.2× smaller than state-of-the-art accelerators with similar accuracy.

**Key Words:** Low Power, High Accuracy, Stream Inference, Dynamic Scaling, Neural Network Accelerator，In-memory Computing

# **目录**

[摘要 I](#_Toc92321167)

[Abstract II](#_Toc92321168)

[目录 IV](#_Toc92321169)

[**图目录** VI](#_Toc92321170)

[**表目录** VII](#_Toc92321171)

[**算法目录** VIII](#_Toc92321172)

[第1章 绪论 1](#_Toc92321173)

[1.1 课题背景与意义 1](#_Toc92321174)

[1.2 国内外研究现状 2](#_Toc92321175)

[1.3 研究内容与章节安排 3](#_Toc92321176)

[第2章 神经网络专用处理器技术基础 6](#_Toc92321177)

[2.1 神经网络模型 6](#_Toc92321178)

[2.1.1 神经元模型 7](#_Toc92321179)

[2.1.2 全连接网络 8](#_Toc92321180)

[2.1.3 卷积神经网络 9](#_Toc92321181)

[2.1.4 深度可分离卷积神经网络 10](#_Toc92321182)

[2.1.5 时序卷积神经网络 10](#_Toc92321183)

[2.2 神经网络模型压缩 12](#_Toc92321184)

[2.3 神经网络专用处理器架构设计 13](#_Toc92321185)

[2.3.1 基于存内计算的神经网络处理器 13](#_Toc92321186)

[2.3.2 流式预测与分批预测 14](#_Toc92321187)

[2.4 本章小结 15](#_Toc92321188)

[第3章 面向神经网络模型量化的动态缩放 17](#_Toc92321189)

[3.1 神经网络模型量化基础 17](#_Toc92321190)

[3.2 基于存内计算核的动态缩放方法设计 17](#_Toc92321191)

[3.3 动态缩放方法实验对比 23](#_Toc92321192)

[3.4 本章小结 25](#_Toc92321193)

[第4章 基于存内计算的流式神经网络处理器设计与实现 26](#_Toc92321194)

[4.1 处理器整体架构设计 26](#_Toc92321195)

[4.1.1 控制模块设计 28](#_Toc92321196)

[4.1.2 SPI模块 29](#_Toc92321197)

[4.1.3 基于Flash存储器的存内计算核 29](#_Toc92321198)

[4.1.4 数字协处理器设计 32](#_Toc92321199)

[4.1.5 定制指令集设计 32](#_Toc92321200)

[4.1.6 处理器模式介绍 37](#_Toc92321201)

[4.2 基于分块串行存内计算架构的流式预测 38](#_Toc92321202)

[4.2.1 TC-Resnet模型流式预测优化方法 40](#_Toc92321203)

[4.2.2 处理器流式预测硬件执行方法 43](#_Toc92321204)

[4.3 实验验证与分析 47](#_Toc92321205)

[4.3.1 实验环境 47](#_Toc92321206)

[4.3.2 语音关键词识别任务功耗分析 48](#_Toc92321207)

[4.3.3 与其他神经网络处理器性能对比 50](#_Toc92321208)

[4.4 本章小节 51](#_Toc92321209)

[第5章 总结与展望 52](#_Toc92321210)

[5.1 总结 52](#_Toc92321211)

[5.2 展望 52](#_Toc92321212)

[参考文献 56](#_Toc92321213)

[致谢 60](#_Toc92321214)

[在读期间取得的科研成果 61](#_Toc92321215)

**图目录**

[图 2‑1序列信号处理流程图 14](#_Toc92332267)

[图 2‑2神经元模型 15](#_Toc92332268)

[图 2‑3全连接网络 16](#_Toc92332269)

[图 2‑4卷积层结构示意图 17](#_Toc92332270)

[图 2‑5 卷积层伪代码图 18](#_Toc92332271)

[图 2‑6时序卷积残差神经网络TC-Resnet结构图 19](#_Toc92332272)

[图 2‑7一维卷积二维卷积对比图 19](#_Toc92332273)

[图 2‑8分批预测与流式预测对比 23](#_Toc92332274)

[图 3‑1神经网络模型量化 25](#_Toc92332275)

[图 3‑2单通道存内计算核图 26](#_Toc92332276)

[图 3‑3神经网络激活量化示意图 26](#_Toc92332277)

[图 3‑4 SAR ADC能耗、面积与位数关系图 27](#_Toc92332278)

[图 3‑5 ADC低量程利用率示意图 28](#_Toc92332279)

[图 3‑6参数倍增图 29](#_Toc92332280)

[图 3‑7动态调整积分时间图 30](#_Toc92332281)

[图 3‑8动态缩放前后ADC量程利用率对比图 30](#_Toc92332282)

[图 3‑9偏移量化图 31](#_Toc92332283)

[图 3‑10模型推理过程采用偏移量化图 32](#_Toc92332284)

[图 3‑11是否采用动态缩放前后准确度对比 32](#_Toc92332285)

[图 4‑1处理器整体架构图 34](#_Toc92332286)

[图 4‑2处理器模式转换图 35](#_Toc92332287)

[图 4‑3流式预测下模型转换图 36](#_Toc92332288)

[图 4‑4控制模块图示 36](#_Toc92332289)

[图 4‑5存内计算核架构图 38](#_Toc92332290)

[图 4‑6存内计算核单次计算各阶段示意图 39](#_Toc92332291)

[图 4‑7位选模块电路结构 39](#_Toc92332292)

[图 4‑8指令集 40](#_Toc92332293)

[图 4‑9流水线优化 42](#_Toc92332294)

[图 4‑10TC-Resnet8 CONV0层执行图 43](#_Toc92332295)

[图 4‑11 TC-Resnet8 CONV0相邻帧间指令执行情况及数据重用示意图 45](#_Toc92332296)

[图 4‑12Flash擦除后单元阻值分布图 45](#_Toc92332297)

[图 4‑13Flash直接编程后单元阻值分布图 46](#_Toc92332298)

[图 4‑14Flash读取、编程后单元阻值分布图 46](#_Toc92332299)

[图 4‑15 分立存内计算核架构 47](#_Toc92332300)

[图 4‑16 集中式存算核分块串行 47](#_Toc92332301)

[图 4‑17 Padding 0对数据重用的影响图 49](#_Toc92332302)

[图 4‑18残差层优化前后结构对比图 49](#_Toc92332303)

[图 4‑19残差块结构 50](#_Toc92332304)

[图 4‑20 本工作设计的TC-Resnet结构图 50](#_Toc92332305)

[图 4‑21一维卷积层流式执行图 52](#_Toc92332306)

[图 4‑22全连接层流式执行图 53](#_Toc92332307)

[图 4‑23平均池化层流式执行图 53](#_Toc92332308)

[图 4‑24 TC-Resnet流式预测图 54](#_Toc92332309)

[图 4‑25芯片版图 56](#_Toc92332310)

[图 4‑26每帧预测模式下功耗分布图 57](#_Toc92332311)

[图 4‑27每8帧预测模式下功耗分布图 57](#_Toc92332312)

**表目录**

[表格 2‑1卷积层各参数说明表 9](#_Toc92321262)

[表格 3‑1 缩放因子调整汇总表 23](#_Toc92321263)

[表格 4‑1片上存储器名称、类型、容量表 26](#_Toc92321264)

[表格 4‑2基于Flash的存内计算核各模式说明 28](#_Toc92321265)

[表格 4‑3 TC-Resnet8 CONV0层指令 35](#_Toc92321266)

[表格 4‑4 TC-Resnet准确度 43](#_Toc92321267)

[表格 4‑5分批预测与流式预测对比 46](#_Toc92321268)

[表格 4‑6每帧预测模式各模块功耗组成 48](#_Toc92321269)

[表格 4‑7每8帧预测模式各模块功耗组成 49](#_Toc92321270)

[表格 4‑8性能对比图 50](#_Toc92321271)

# 绪论

## 课题背景与意义

人工智能物联网（Artificial Intelligence of Things,AIoT）在物联网设备上结合了人工智能技术来实现更有效的物联网操作，通过增强数据的分析过程，提升人和设备之间的交互能力。人工智能可以被用来对原始的物联网数据进行特征提取，从而提升相关任务决策的表现。轻量级神经网络算法被广泛用在物联网设备中进行序列信号处理，如语音关键词识别[1]，心电图信号处理[2]，脑电图信号处理[3]等。其中语音关键词识别检测语音信号流中关键词是否出现及出现的位置，可以用在对实时语音信号的检测，也可以用在对海量语音库中关键词的检索。目前语音关键词检测大量应用在语音控制的物联网设备中，如手机、可穿戴手表、智能家居等，语音关键词检测可以帮助人们实现对智能设备的无接触控制，带来更好的用户体验。语音关键词检测在这些设备中承担触发功能，设备在未触发状态下处于低功耗的休眠模式，语音关键词检测系统实时检测接收的语音信号中是否包含特定的关键词，当检测到特定关键词后，对设备进行唤醒。配合语音关键词检测，设备仅在被特定关键词触发后进入工作模式，从而降低了设备整体的功耗，延长了电池的寿命。由于语音关键词识别承担前端的触发功能，因此需要处于常开工作状态。

物联网设备如可穿戴手环等，通常由电池供电或者充电，一个典型的LR44型号纽扣电池的容量为190mAh/1.5V[4],为了尽量延长电池的使用寿命，神经网络加速器模块的功耗需要尽可能地低。延时影响设备与用户间的交互速度，因此低延时对好的用户体验来说十分重要。在物联网应用中使用通用处理器运行神经网络算法，在功耗、延时上无法满足要求，因此，有必要设计专用的神经网络加速器。

存内计算技术打破了冯·诺依曼架构中存储墙的限制，在存内计算中存储器承担了部分计算功能，计算可以在存储器内和临近存储器的电路中完成，从而大大减少了存储器访问的次数，减少了数据搬移的功耗[5]。存储器内计算技术具有高效能，高吞吐量的优点，适合用来加速计算量和数据量大的神经网络算法。本文结合了存内计算技术来设计低功耗、低延时的神经网络加速器。

## 国内外研究现状

最近几年的研究工作提出了一些语音关键词识别芯片[6][7][8][9][10][11][12]。语音关键词识别任务包含了功耗、准确度、延迟等多个重要的指标，需要兼顾不同性能指标的优化。其中功耗表示芯片进行实时语音关键词识别时单位时间内消耗的能量，部分工作中使用了每完成一次语音关键词识别的能耗作为指标[9][10]，功耗/能耗直接影响物联网设备中电池的寿命。延时表示设备在接收到语音信号到产生识别结果之间的时长，越低的延时表示越快的响应速度。准确度衡量了设备执行语音关键词识别任务的能力，评价准确性的最佳方法是通过标准化的基准测试和数据集。与图像识别等其他人工智能领域不同，语音关键词任务中用于实现客观比较的数据集比较匮乏。语音命令数据集(GSCD)[13]是少数广泛使用的语音关键词数据集之一，在本工作中使用不同模型在GSCD数据集上的表现来进行准确度的比较。对同一种网络而言，在一定范围内网络的参数越多，网络的准确度越高，但需要的计算量越高，对应硬件的动态功耗越高，同时也需要更多的片上存储，从而产生更大的漏电功耗，此外计算量的升高也会带来延时的增加。因此，在对不同语音关键词识别芯片的对比中，需要综合考虑不同的指标。

文献[6]中采用了全数字电路的架构，使用了深度可分离卷积神经网络，其中的深度可分离卷积层相比传统卷积层计算量减少了6倍。该工作采用面向深度可分离卷积的硬件优化方法减少了冗余计算量和存储空间，每帧计算量仅包含8736次MAC(Multiplier and Accumulation)运算，在GSCD(Google Speech Command Dataset)数据集上的准确度为94.6%，但由于采用的神经网络模型较小，因此只能识别2个关键词。该芯片工艺为28nm，在0.41V，40KHz下测试性能，神经网络加速器部分的功耗仅为0.17uw，但低频同时带来了较高的延时，[6]中的延时达到了64ms。

[7]中基于全数字电路架构，设计了面向时序卷积残差网络（temporal convolutional residual neural network, TC-Resnet）的神经网络加速器。[7]中采用了条件执行策略，对每个残差层结果送入退出分支中进行计算，若当前结果已经达到了退出标准，则直接输出预测结果，结束本次预测，从而减小预测的计算量。[7]采用22nm 工艺，可预测关键词个数为10，在GSCD数据集上准确度为93.09%，虽然该工作准确度较高，但功耗和延时也较高，其中功耗为8.2uw，每次预测延迟100ms。

[10]中基于SRAM内计算技术，设计了面向注意力循环网络（recurrent attention model, RAM）的神经网络加速器，该工作充分利用了SRAM存内计算核高能效和高吞吐量优势，具有很低的动态功耗和预测延时，单次预测的能耗为0.44uJ, 单次预测的延时为39.9us，但是该工作中片上大容量的SRAM会带来大的漏电功耗，而在物联网应用中，静态功耗的占比很高[6]。[10]采用65nm工艺，可识别的关键词个数为7个，GSCD数据集上准确度为90.38%。

此外上述提到的文献[7][8]具有可配置的特性，可以分别运行可变大小的TC-Resnet和TCN网络。文献[6][9][10][11]设计了面向语音关键词识别的专用芯片，架构固定，灵活性低。用于语音关键词识别的神经网络模型也可以进一步迁移到其他序列信号如心电图信号的处理，因此设计具有一定灵活性的神经网络处理器架构，支持不同的序列信号处理，可以进一步拓展芯片的应用场景。

综上所述，之前支持语音关键词识别的加速器的相关工作，在功耗、延时和准确度等方面还存在提升的空间，特别是在高准确率下功耗和延时的表现。

## 研究内容与章节安排

本文的研究目标是结合存内计算技术高能效，高吞吐量的优势来设计面向物联网应用的神经网络处理器，在语音关键词识别任务上达到低功耗、高准确度和低延时的表现。

基于存内计算技术，设计一个面向物联网应用的神经网络处理器，需要解决如下问题：

1. 存内计算技术中往往使用ADC进行结果的量化[5]，随着ADC的位数增加，ADC的精度增加，但功耗和面积也急剧增加，出于准确度和功耗、面积的折衷考虑，往往使用低位宽ADC。神经网络算法中参数和激活动态变化，每层激活的缩放因子具有一定的分布范围，如何充分利用ADC的量程，在低位宽ADC的基础上实现高准确度量化需要进一步研究。
2. 尽管存内计算技术具有高能效和高吞吐量的优势，但是存内计算核大小固定，计算的粒度很大，并且具有参数固定的特性。在物联网应用严格的功耗和延时限制下，基于存内计算技术设计具有一定灵活性的神经网络加速器架构还是一个挑战。

本文针对上述问题提出了相应的解决方案，主要研究工作如下：

1. 介绍了用于包含语音关键词识别在内的序列信号处理的神经网络模型，结合存内计算硬件特点，对不同的模型进行了比较，并选择了对于存内计算硬件友好的时序卷积残差神经网络模型。
2. 设计了面向神经网络量化模型的动态缩放方法，通过使用该动态缩放方法，存内计算核可以基于低位宽ADC实现宽范围的缩放因子，充分利用了低位宽ADC的量程，提高了量化的准确度。
3. 设计了基于存内计算的流式神经网络加速器架构，并设计了支持存内计算的定制指令集，处理器可以通过配置指令灵活执行TC-Resnet类拓扑结构的网络，实现包括语音关键词在内的多种序列信号处理任务；处理器通过流式预测方式执行增量计算，充分利用数据重用，减小了每帧的计算量，大幅降低了处理器的功耗和延时。

本文共包含五章，各章的主要内容如下：

第一章绪论介绍了论文的背景，阐述了课题研究的意义，然后介绍了国内外面向语音关键词的神经网络加速器的研究现状。通过分析研究现状，提出了现有的神经网络加速器存在的问题，并给出了本文解决这些问题的思路。

第二章为面向物联网应用的神经网络加速器优化技术介绍。从神经网络模型介绍、算法优化、硬件设计三个方面展开。

第三章分析了存内计算核执行激活量化的过程，并针对如何充分利用低位宽ADC量程提高量化准确度的问题，提出了动态缩放方法，通过软件仿真对该方法进行了效果验证。

第四章讲述面向基于存内计算的流式神经网络处理器设计方法及细节，处理器包括控制模块，片上存储器，张量处理单元和SPI协议控制模块，设计了支持TC-Resnet类网络的专用指令集，提出了面向TC-Resnet网络的流式预测方法及其硬件执行方法，通过充分利用数据重用，大幅降低了计算量，并展示了处理器整体的仿真实验结果。

第五章为总结与展望。首先总结论文所做的工作，然后展望了后续进一步的研究工作。

# 神经网络专用处理器技术基础

本章首先介绍了神经网络模型基础，比较了用于序列信号处理的不同网络模型优缺点，并结合存内计算硬件特点，阐述了本文选择TC-Resnet类网络作为处理器目标算法的原因；然后介绍了神经网络模型压缩相关理论；最后介绍了神经网络专用处理架构设计，其中重点介绍了基于存内计算的神经网络处理器的不同架构，最后介绍了流式预测和分批预测两种硬件执行方式的区别。

神经网络算法具有参数量大、数据量大的特点，部署在通用处理器平台上在功耗、延时上难以满足要求，因此有必要设计神经网络专用处理器

本章阐述了神经网络专用处理器技术基础，首先

神经网络模型基础，然后介绍了神经网络模型压缩方法，最后介绍了神经网络处理器架构设计，从

## 神经网络模型

常规的序列信号处理流程如图 2‑1所示，包含了3个阶段：前端传感器信号采集、预处理和神经网络分类。其中前端传感器采集原始的模拟信号，对其进行放大和滤波等操作，然后转换为数字信号；预处理对信号进行特征提取，降低数据维度；神经网络分类阶段进行进一步的特征提取，完成分类，输出判别为每个类别的概率。一些工作使用了端到端的神经网络模型，这种网络模型可以直接处理前端传感器采集到的原始信号[14]。



图 2‑1序列信号处理流程图

在序列信号处理的整个流程中，神经网络模型对于最终的识别效果有着决定性作用，并且神经网络的计算量和存储空间在整个系统中占比非常大，因此设计性能优异的神经网络模型对于序列信号处理来说十分重要，下文对神经网络基本理论进行介绍，分析了几种神经网络的优缺点，并选择了适合存内计算硬件的神经网络模型。

早期被用来进行时间序列信号处理的机器学习算法是隐马尔可夫模型（Hidden Markov Models，HMMs）[15]。近年来随着人工智能技术的不断发展，基于深度学习的模型展现出了更好的准确度和鲁棒性[16]。

[17]中使用了基于全连接层的深度神经网络进行语音关键词识别，实现了较高的准确度，但是全连接层的形式并不能够有效建立不同输入间在时间上的联系。为了解决这个问题，一些工作提出了用于时间序列信号处理的CNN[18]和RNN[19]模型。CNN使用在时间维度上的卷积来提取贯穿在时间维度上的特征。典型的RNN包含长短期记忆（Long-short Term Memory，LSTM）网络和门控循环单元（Gated Recurrent Unit，GRU）。LSTM解决了普通RNN在长序列训练过程中的梯度消失和梯度爆炸问题。GRU则是为了解决LSTM计算过于复杂而提出。最近提出的深度可分离卷积神经网络（Depthwise Separable Convolution Neural Network，DSCNN）[20]和时序卷积网络（Temporal Convolutional Network，TCN）[21]在准确度和减小计算量方面表现更加出色。下面对基础的神经元模型和上述典型模型进行介绍。

### 神经元模型

神经元模型包括输入信号、权重、偏置、求和部分和激活函数。每个神经元可以看作是处理多个输入信号，并产生单个输出信号的信号处理单元。其中输入信号X为维的向量，即。权重为维的向量，，此处的k表示第k个神经元。表示激活函数。表示第k个神经元最终的输出。



图 2‑2神经元模型

输入向量和权重向量进行向量乘法操作，并加上偏置得到，经过激活函数后得到最终的输出，如下式所示：

其中，的大小表明了对于结果的贡献程度，表示第k个神经元的偏置。和均为神经元的可训练参数，神经网络的训练即把和调整到最佳，从而提升网络的拟合能力。

激活函数。卷积层或者全连接层的运算形式本质上是乘加运算，属于线性计算。单纯地叠加线性计算最终得到的还是线性计算。神经网络算法中在每个线性计算层后面均使用了激活函数来引入非线性，提高网络的非线性拟合能力。常用的激活函数有Tanh函数，Sigmoid函数和Relu函数等，Relu函数具有运算简单，收敛速度快等优点。

### 全连接网络

全连接网络包含多层，每层由多个神经元组成，并且相邻两层间的每个神经元均存在连接。全连接网络中的第一层称为输入层，中间层称为隐藏层，最后一层称为输出层，全连接网络也被称为多层感知器。如图 2‑3所示为一个典型的3层全连接网络。每两个神经元之间的连接都对应一个权重，每个神经元包含一个偏置。



图 2‑3全连接网络

全连接网络中每层的计算形式都为一次向量矩阵乘法，如公式所示。

……

上式可整理为下式所示的矩阵运算

全连接层每一层的计算可以看作为一次向量-矩阵乘法，其中向量维度为，矩阵维度为。每个全连接层的乘加计算数量为，由于全连接层每个权重均只参与了一次计算，权重没有重用的情况，所以全连接层具有参数量大的特点。全连接层具有优异的分类性能，主要用在深度神经网络的最后几层实现分类输出。

### 卷积神经网络

卷积神经网络的核心在于卷积层的使用，不同于全连接层中相邻两层神经元全连接的特性，卷积神经网络中每个神经元均与上一层中的部分神经元进行连接，即局部连接。每个局部连接对应了一个卷积核，卷积核对输入激活进行滑窗卷积操作，实现了权值共享，从而大大减少了卷积层中的参数量。

一个典型的卷积层如图 2‑4所示，图中各符号的意义如下表所示。



图 2‑4卷积层结构示意图

表格 2‑1卷积层各参数说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 符号 | 意义 | 符号 | 意义 |
| IW | 输入特征图宽度 | OW | 输出特征图宽度 |
| IH | 输入特征图高度 | OH | 输出特征图高度 |
| IC | 输入特征图通道数 | OC | 输出特征图通道数 |
| KW | 卷积核/权重宽度 | S | 卷积步长 |
| KH | 卷积核/权重高度 |  |  |

其中输入特征图为大小的三维矩阵，权重为大小的四维矩阵，对应OC个维度为的卷积核，输出特征图为大小的三维矩阵。卷积操作的流程如图 2‑5所示。

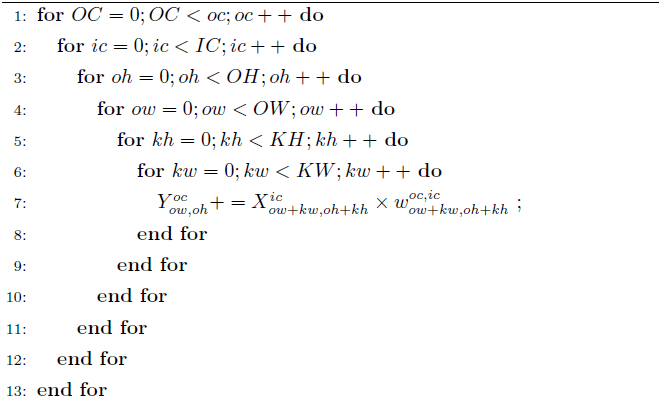


图 2‑5 卷积层伪代码图

### 深度可分离卷积神经网络

DSCNN使用深度可分离卷积层替代了传统的卷积层，每个深度可分离卷积层由一个逐通道卷积层（Depthwise convolution）和一个逐点卷积层组成（Pointwise Convolution）。逐通道卷积层分通道进行卷积，即每个通道上进行二维卷积，经过逐通道卷积层后特征图的通道数不变；逐点卷积层卷积核的宽和高均为1，对各通道之间的数据进行加权组合。相比传统的三维卷积，深度可分离卷积在运算上更加高效，大幅减少了参数量和计算量。但是由于逐通道卷积层中输入通道数IC为1，不存在输入通道上的累加，即每个输出激活的计算相当于两个维度向量的乘法，由于KW和KH的大小一般较小，常见的KW和KH均为3，而存内计算硬件计算的粒度一般较大，因此逐通道卷积层结构难以高效映射到存内计算硬件上。

### 时序卷积神经网络

时序卷积神经网络通过使用膨胀卷积的方法，在计算量没有增加的情况下，增大了感受野[21]。文献[1]中提出了带有残差层的时序卷积神经网络TC-Resnet8，使用了沿时间维度的一维卷积，并结合了残差结构，进一步提高了网络的准确度。文献[1]中提出的TC-Resnet8网络结构如图 2‑6所示,其中的参数意义同表。其中包含了1个一维时序卷积层conv0，3个残差块，1个平均池化层和1个全连接层。其中每个残差块中又分为主路径和分支路径，主路径上包含了2个一维时序卷积层conv\_1和conv\_2，分支路径上包含了1个一维时序卷积层conv\_3，主路径的结果和分支路径的结果进行元素级加后为残差块最终的输出。



图 2‑6时序卷积残差神经网络TC-Resnet结构图

文献[1]中使用的一维卷积结构相比二维卷积可以大幅减少计算量。如图 2‑7（a）所示，MFCC预处理之后的特征图维度为，其中表示时间维度，表示特征维度。传统二维卷积中，把经过MFCC预处理之后的特征图当作一个灰度图片来进行处理，如图 2‑7 (b)中的二维卷积层包含个大小的卷积核，卷积步长为1（包含padding 0），该二维卷积层中卷积核在时间维度和特征维度进行滑窗操作，输出的特征图大小为。总的MAC数为。



图 2‑7一维卷积二维卷积对比图

一维卷积中把MFCC预处理后的每帧特征向量，当作时间序列信号，如图 2‑7(c)所示，即特征图分为IW帧，每帧的特征向量维度为，整个特征图的维度为。共包含个大小的卷积核，卷积步长为1（包含padding 0），该一维卷积层中卷积核只在时间维度进行滑窗操作，输出特征图大小为，总的MAC数为。假设二维卷积层和一维卷积层具有相同的参数量，则，进一步得到二维卷积层与一维卷积层的MAC数的比例为，由此可见，使用一维卷积层后在参数量相同的情况下，相比二维卷积层减少了IC倍的计算量。

文献[2]和文献[3]分别把TC-Resnet网络结构用于心电信号处理和脑电信号处理中，并取得了优异的性能表现，展现出了TC-Resnet网络在序列信号处理中的巨大潜力。

由于RNN需要复杂的激活函数，DSCNN中的逐通道卷积不适合使用存内计算硬件实现，而TC-Resnet具有准确度高，结构相对简单的特点，本文选用TC-Resnet网络来实现语音关键词识别等序列信号处理任务。

## 神经网络模型压缩

最近的研究表明，在神经网络的推理计算中，可以在保证网络准确度的前提下，把全精度浮点数表示的激活和权重量化为低位宽[22][23]。经过低位宽量化后，可以显著降低神经网络的存储空间，减小对带宽的需求，同时降低了计算的功耗，且大大提升了硬件执行推理计算的速度。量化可以分为线性量化和非线性量化[24]。常规的线性量化过程可以使用下式表示

其中q为浮点数表示的量化前的原始值，为舍入函数，r为量化后的值，Z为浮点数0量化后的值，Z为0时称为对称线性量化，Z不为0时称为非对称线性量化[24]。对于线性量化，量化后的比特数越低，网络的准确度下降幅度越大，保证网络准确度没有明显下降的最低位宽要求在8比特附近[25]。使用混合精度量化的方式，对各层的参数和数据分别进行细粒度的量化，可以进一步在量化位宽和网络精度上达到更好的平衡。此外，对网络进行二值化量化使得每个参数仅由1比特表示，并且可以把乘法计算转换为位运算，极大减小网络的存储空间和计算量，但由于量化位宽只有1比特，因此会带来较大的信息损失，会对神经网络的准确度产生较大影响[26]。

## 神经网络专用处理器架构设计

目前可用于神经网络加速的硬件平台包括中央处理器CPU、图形处理器GPU、现场可编程逻辑门阵列FPGA和专用集成电路ASIC。

中央处理器CPU包含大容量的缓存和复杂的逻辑控制单元，算术逻辑单元（Arithmetic and Logic Unit，ALU）的个数少，适合执行逻辑复杂的计算，通用性高，不适合来做逻辑简单，但数据量大，并行度高的计算。

图形处理器GPU包含数量众多的ALU，逻辑控制单元相对较少，适合用来做逻辑相对简单，但计算并行度高的计算。

现场可编程逻辑门阵列FPGA可以根据开发者不同需求，通过编程实现不同的电路结构，灵活度高，开发周期短，可以及时适应不断变化的神经网络模型，但FPGA价格昂贵，性能相对专用集成电路ASIC还有不小的差距。

专用集成电路ASIC基于某种特定的需求对电路进行定制化设计，其电路架构等可以根据算法的特点进行定制，从而可以达到极高的性能。面向物联网应用的神经网络加速器有着严格的功耗和延时要求，采用ASIC的设计在该应用场景下具有明显优势。

面向物联网应用的神经网络加速器对功耗和延时非常敏感，往往采用轻量级网络，因此与典型加速器相比，此类型加速器中网络的参数往往全部存储在片上，从而避免了访问片外存储器带来的高额代价[27]。

### 基于存内计算的神经网络处理器

在传统冯诺依曼架构中，运算器和存储器分离，完成一次运算需要把对应的操作数从存储器中读出，送入运算器执行计算，这其中存储器的访问消耗了相当大部分的能量。存内计算技术在存储器中嵌入了计算的功能，大大减少了存储器访问的次数[5][28][29]。

存内计算技术虽然具有高能效和高吞吐量的优势，但是由于单个存算核的计算粒度大，而且存内计算具有权重固定的特性，因此基于存内计算设计的神经网络加速器往往不够灵活。如文献[9]和文献[10]中基于存内计算设计了专用的语音关键词识别芯片。最近一些工作对针对存内计算的灵活性问题进行了研究。

文献[30]和文献[31]中对存内计算核设计了专用指令集，通过指令把大部分向量运算映射到存内计算核上执行，并配合使用标量处理器来执行算法中逻辑相对复杂的部分。文献[32]和文献[33]中结合片上网络技术（Network on Chip，NOC）。文献[32]和文献[33]中具有大规模的存内计算核阵列，存内计算核位于NOC的节点上，不同存算核之间的数据流动通过NOC中的路由实现。但是文献[30][31][32][33]主要出于通用性的角度进行设计，NOC和配合通用处理器的方案对于追求超低功耗的物联网应用来说存在功耗太大的问题。

存内计算核设计。根据存内计算核所采用的存储器类型，可以分为2类，一类基于易失性存储器如动态随机存取存取存储器（Dynamic Random Access Memory，DRAM）和静态随机存取存储器（Static Random-Access Memory，SRAM）。这类存内计算核静态漏电功耗较高，导致整体的计算。另一类基于非易失性存储器如闪存（Flash Memory）、可变电阻式存储器（Resistive Random Access Memory，RRAM）、相变存储器（Phase Change Memory，PCM ），这类存内计算核需要先把参数写入到存储器中，非易失性可以保证掉电后数据不会丢失，漏电功耗低，适合用在需要长待机的物联网设备中[5]。其中Flash存储器具有工艺成熟度高等优点，本处理器选择嵌入了一块基于Flash存储器的存内计算核。

### 流式预测与分批预测

序列信号处理可以使用流式预测或者分批预测的方式执行[27][34][35]。

在分批预测中，多个相邻帧的特征向量打包为一批，一批特征向量即对应了一个完整的样本，神经网络加速器每次对一个完整的样本进行计算，然后输出该样本的分类结果。由于无法提前预知序列信号流中关键信息出现的位置，因此在实时的序列信号处理过程中，分批预测需要对包含重叠信息的批次进行处理。极限情况下分批预测中神经网络加速器每帧都对一个完整的样本进行计算并输出预测结果，但是相邻帧的2个样本之间只有一帧特征向量是不同的，因此相邻2次分批计算中存在大量可以复用的结果。

流式预测充分重用了相邻样本间的可复用的结果，执行增量计算，每帧不再处理整个样本，而是只处理当前帧的样本与前一帧样本不同的部分。与分批预测的方法相比，流式预测大大减少了实时序列信号处理中神经网络每帧的计算量，从而大幅减小了整体的功耗和每帧预测的延时。

如图 2‑8所示，以CNN为例，在分批预测中加速器每次对整个样本进行计算如图 2‑8（a）所示，对整个输入特征图进行卷积；在流式预测中，之前可重用的结果被缓存下来，每帧该卷积层只进行一个卷积窗口的计算[27],如图 2‑8（b）所示。图中的dense表示全连接层等结构，在该结构中，分批预测核流式预测均对整个输入特征图进行计算。



图 2‑8分批预测与流式预测对比

由于神经网络模型在训练过程中通常采用分批计算的方式，之前的工作[1]中主要考虑网络的准确度和分批计算的计算量，没有对使用流式预测的方式把模型部署在硬件上进行进一步的探索，而且并不是所有的网络模型都支持流式预测。文献[34]中提出了一个优化工具，可以把一些网络部署在Keras框架上进行流式预测。其中RNN是支持流式预测的一个典型模型，在RNN中使用了内部状态来保存之前时间的信息。文献[6]中在ASIC平台上采用全数字电路架构实现了DSCNN的流式预测，每帧的计算量减少了94.3%，得益于每帧计算量减少，文献[6]中可以进一步降低频率和电压，从而进一步降低功耗。但是基于存内计算的加速器，如文献[10][11]中，仍使用了分批预测的方式，因此基于存内计算的流式预测加速器架构值得进一步研究。

## 本章小结

本章首先介绍了神经网络模型基础，比较了用于序列信号处理的不同网络模型优缺点，并结合存内计算硬件特点，阐述了本文选择TC-Resnet类网络作为处理器目标算法的原因；然后介绍了神经网络模型压缩相关理论；最后介绍了神经网络专用处理架构设计，其中重点介绍了基于存内计算的神经网络处理器的不同架构，最后介绍了流式预测和分批预测两种硬件执行方式的区别。

# 面向神经网络模型量化的动态缩放

## 神经网络模型量化基础

本节主要介绍本设计中采用的线性量化方式，并介绍在模型推理过程中硬件需要执行的量化操作。采用文献[25][26]中的线性量化方式，可以实现对神经网络模型低位宽量化，本设计中对权重和激活均采用了6bit量化。如图 3‑1所示，模型量化后得到低位宽参数和每层激活的缩放因子。在对模型的量化过程中，出于便于硬件实现的考虑，限定每层的缩放因子为2的幂次，即，其中n为自然数，缩放因子表示对神经网络中激活量化的尺度。由于神经网络模型各层的数据分布存在差异，因此量化后神经网络模型各层的缩放因子在一定范围内分布。



图 3‑1神经网络模型量化

如图 3‑1所示，使用量化后模型执行推理过程中，参数直接以低位宽参与计算，神经网络每层的原始输出激活为高位宽，需要根据缩放因子从高位宽量化到低位宽，然后作为下一层的输入，本章主要集中在激活量化的硬件实现。

## 基于存内计算核的动态缩放方法设计

本设计中把神经网络中的卷积层/全连接层均映射到存内计算核上执行，其中一个单通道的存内计算核如图 3‑2所示，该存内计算核每次执行向量和向量的乘法并进行量化操作，输出结果为标量，如下公式所示，其中均为256维向量，为6bit无符号数，为6bit有符号数。

原始输出激活的最大位宽理论上为，如下公式所示(减1表示由于采用了Relu激活函数，所以输出结果的值全为非负数，减去1比特符号位占用的空间)：

需要重新量化为6bit，然后作为下一层的输入。根据缩放因子，量化过程为对19bit的原始输出激活取中间的，对低n进行四舍五入，对高进行饱和操作，如图 3‑3所示。饱和操作是指若高中存在1，表示要量化的值超出了6bit所能表示的最大范围，则最终的量化结果取6bit表示的最大值，即。



图 3‑2单通道存内计算核图



图 3‑3神经网络激活量化示意图

本设计中采用的存内计算核使用了ADC来进行原始输出激活的量化，对应公式中的量化函数。由于存内计算核计算速度为0.42M，且对功耗具有严格的要求，因此选用了低功耗、中高精度、中低速的逐次逼近式模数转换器（Successive-approximation register ADC，SAR ADC）。ADC的位数越高，量化的范围越宽，量化的准确度越高，但同时ADC的功耗和面积越大。SAR ADC的单次模数转换的能耗如公式所示，其中n表示ADC的位数，表示单位电容，表示参考电压。

根据公式，SAR ADC随位宽升高，功耗成指数增加，如图 3‑4所示。

SAR ADC面积和位宽的关系如公式所示，其中n表示ADC的位数，表示单位电容的面积。在本设计中，SAR ADC的面积占比较大，且出于版图对称性角度的考虑，增加SAR ADC的面积会显著增加处理器的整体面积。

根据公式，SAR ADC随位宽升高，面积成指数增加，如图所示。

图 3‑4 SAR ADC能耗、面积与位数关系图

之前的工作[18][19][20]集中在对存内计算核阵列的实现，没有对存内计算核执行激活量化的过程进行深入讨论，其中存内计算核的功能为向量矩阵乘法，向量和矩阵中每个元素的位宽均为8比特，输出结果也量化到8bit，因此存内计算核中选用了8bit ADC。在文献[18][19][20]中进行测试时，使用了典型的数据分布情况，即输入激活/参数的值均比较大，在这种情况下，向量矩阵乘的结果比较大，使用8比特ADC对结果的高8比特进行量化具有很高的量程利用率，ADC量化的准确度高。

由于本工作中模型量化为6bit，因此理论上至少需要采用6bit ADC，即对19bit结果的高6bit进行量化，提供的缩放因子。

由于神经网络每层的网络大小不同，以及每层的数据分布不同，因此经过模型量化后每层结果的缩放因子S在一定范围内波动。当比较小的时候，使用6bit ADC进行量化时，ADC的量程利用率比较低。如图 3‑5所示，紫色部分表示ADC量化的部分，红色阴影部分表示某层结果的有效位数实际分布的部分，（紫色部分和红色阴影重叠的部分表示ADC量程实际利用到的部分），ADC量化的准确度低。



图 3‑5 ADC低量程利用率示意图

理论上可以通过增大ADC位宽来提高量化的准确度，具体操作为配合使用多路选择器MUX从ADC 的高位宽结果中选出6bit，并进行四舍五入和饱和操作。但由于ADC功耗随位宽呈指数增加，因此直接使用该方法会显著增加整体功耗。使用该方法在6bit ADC的基础上把ADC位宽增加bit，可以提供一个的缩放因子。

本设计提出了一种动态缩放因子调整的方法，在尽可能减少ADC位宽的基础上，保证高ADC量程利用率来实现每层激活的量化，提高了低位宽ADC量化的准确性。该动态缩放因子调整方法包含3个可选步骤，分别为：

（1）调整ADC的量程。理论上ADC的满量程应该对应输入数据X和权重W全为最大值时的输出结果最大值，但是神经网络中的参数和激活的数据分布具有稀疏性，因此在神经网络中输出结果的最大值，所以ADC的量程可以不对应理论上的最大值，而是对应一个相对较小的值，其中和均为2的正整数次幂，，。ADC的量程由参考电压决定，可以通过调整实现量程的调节。该方法提供了一个的缩放因子，如下公式所示。由于ADC的量程太小会导致模数转换的误差增高，因此只能在有限的范围内进行调整。

。

（2）参数倍增。在存内计算核的参数存储设计上，本工作对6bit有符号参数使用差分编码的方式存储在Flash存储器的12个cell中，如图 3‑6所示。考虑到6bit有符号参数的范围为，因此实际占用了10个cell（和），空余了2个cell（和），提供了参数倍增的空间。如图 3‑6所示，该12个cell中原来存储的参数值为32，经过参数倍增后，存储的参数变为64。在模型量化后已经决定了每层结果的缩放因子，当该层缩放因子太大的情况下，可以在对Flash存储进行编程操作的时候写入倍增后的权重，相当于把数据的有效位数向高位移动，即把图 3‑6中红色阴影部分左移。选择是否倍增存储的参数可以提供的量化因子如下公式所示。使用更多的cell来存储6bit有符号参数可以使有更大的调整范围，但也会使得Flash利用率下降，导致Flash可容纳的参数个数下降，出于调整范围和利用率的综合考虑，本设计中选用了12个cell来存储差分编码的6bit有符号参数。

。



图 3‑6参数倍增图

（3）动态调整存内计算核的积分时间。ADC的输入电压值（积分电路的输出）和积分时间成正比，积分时间越长，ADC的输入电压越大。存内计算核控制电路产生存内计算核的控制波形，该波形决定了存内计算核每次的积分时间，如图 3‑7所示，积分时间由S2/S5的下降沿和S1的下降沿之间的S2/S5的脉冲宽度决定，通过控制S2/S5的下降沿的位置，可以实现积分时间的控制，在本设计中积分时间设置了3档调节空间，即30ns，60ns和120ns。“IMC”指令可以对存内计算核控制单元进行调控，实现对不同档位积分时间的切换。通过“IMC”指令进行积分时间的切换可以提供一个的缩放因子，如下公式所示。



图 3‑7动态调整积分时间图

通过上述3个步骤，最终可达到的缩放因子调整范围为，从而实现了宽范围缩放因子的调整，保证在一定范围内分布的缩放因子都可以100%在ADC的量程范围之内，如图 3‑8所示，通过动态缩放因子调整，可以使得结果的有效位数全部落在ADC量程中。



图 3‑8动态缩放前后ADC量程利用率对比图

各步骤优先级问题：步骤3会增大存内计算核的功耗，综合各方法的功耗等考虑，3个步骤的优先级为1-2-3。通过使用动态缩放方法，在ADC位宽为的情况下，可以提供的缩放因子调节范围为，如下表所示。

表格 3‑1 缩放因子调整汇总表

|  |  |  |
| --- | --- | --- |
| 缩放因子调整步骤 | 调整的范围 | 优先级 |
| 增大ADC位宽，配合MUX选出6bit |  | 1 |
| 参数倍增 |  | 2 |
| 调整积分时间 |  | 3 |
| 整体 |  | - |

## 动态缩放方法实验对比

实验使用了TC-Resnet8在GSCD数据集上进行模型准确度测试，本设计中采用的TC-Resnet8模型量化后缩放因子。实验测试了（1）采用动态缩放方法的情况下准确度和ADC位宽的关系；（2）不采用动态缩放方法的情况下准确度和ADC位宽的关系。



图 3‑9偏移量化图

在执行推理过程中，常规量化方法为根据正确的缩放因子S（该缩放因子在模型量化后已决定）进行量化，即取中间6bit，对低位进行四舍五入，对高位进行饱和操作。在测试中，为了模拟由于ADC量程利用率不足带来的误差，采用偏移量化方法，偏移量化方法模拟了使用ADC进行量化时，不同量程利用率产生的影响，即在ADC量程利用率低时，只对高位进行量化，低位补零。偏移量化方法分为3个步骤：（1）根据使用的ADC位宽和是否使用缩放因子调整方法，决定，如图 3‑9所示，，然后根据offset产生新的缩放因子，，使用对结果进行量化。（2）对于量化后的结果乘以offset，即对低位补零。（3）对乘offset之后的结果取低6比特，高位进行饱和操作。测试过程中，把对激活的常规量化方法替换为偏移量化方式，如图 3‑10所示。



图 3‑10模型推理过程采用偏移量化图

实验结果如图 3‑11所示:

图 3‑11是否采用动态缩放前后准确度对比

GSCD数据集为12分类，且每一类样本在总样本中均匀分布。当ADC量程利用率为0进行量化时，ADC的量化结果完全失效，输出为全0，此时模型的推理结果固定判别为12分类中的某一类。所以在ADC量化完全失效的情况下，预测准确度为。

在无动态缩放方法的情况下，当ADC位宽小于8时，ADC的量化结果完全失效，此时模型的准确度均为8.3%；在ADC位宽大于等于8后，随着ADC位宽增大，ADC的量程利用率逐渐升高，模型准确率不断提升，在ADC位数达到15比特后，模型准确度稳定在最高的94.2%。

在采用动态缩放方法的情况下，当ADC位宽小于3时，ADC的量化结果完全失效，因此模型的准确度均为8.3%，在ADC位宽大于等于3后，随着ADC位宽增大，ADC的量程利用率逐渐升高，模型准确率不断提升，在ADC位数达到10比特后，模型准确度稳定在最高的94.2%。

由于ADC的动态功耗和面积随着位数的增加成指数增加，特别是本设计中ADC的面积占比较大，增加ADC位数会带来整体面积显著增大，因此需要在ADC位数尽可能低的情况下达到最高的准确度。在使用动态缩放方法的情况下，ADC位数为8时准确度为94.1%，已经接近最高，出于准确度、功耗和面积的综合考虑，因此本设计选用了“8比特ADC+动态缩放方法”。“8比特ADC+无动态缩放方法”的情况下模型准确度为8.3%，“8比特ADC+采用动态缩放方法”下准确度为94.1%，即在ADC位宽为8的条件下，采用动态缩放方法后，模型准确率提升了85.8%。

## 本章小结

本章首先介绍了神经网络模型量化的基础知识，对量化后模型执行推理过程中的激活量化进行了说明；然后介绍了存内计算核中使用ADC进行激活量化在准确度、功耗和面积上面临的问题，并提出了动态缩放的解决思路，设计相应的存内计算核控制电路，最后进行了软件仿真，实验结果表明，通过动态缩放方法，大幅提高了低位宽ADC进行激活量化的准确度，在使用8bit ADC的情况下，准确度提高了85.8%。

# 基于存内计算的流式神经网络处理器设计与实现

本章将首先介绍处理器的整体架构，依次对处理器的各个模块进行说明，然后对处理器执行流式预测的细节进行阐述，最后与已有的面向物联网应用的神经网络加速器进行对比分析。

## 处理器整体架构设计

处理器整体架构框图如图 4‑1所示，主要包含5个部分：（1）张量处理单元，可以分为2部分，存内计算核和数字协处理器，其中数字协处理器包含向量加/减法器，向量-常数除法器；（2）片上存储器，包括用来存储指令的指令存储器（RF\_INST），用来存储中间结果的数据存储器（RF1），以及分别用来存储2种不同配置信息的存储器RFC0和RFC1；（3）控制模块，包括处理器整体的有限状态机FSM，译码器和存内计算核控制电路；（4）串行外围设备接口SPI（Serial Peripheral interface）模块;（5）芯片输入输出端口。



图 4‑1处理器整体架构图

表格 4‑1片上存储器名称、类型、容量表

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 存储器类型 | 存储数据类型 | 存储器容量 |
| 存内计算核 | Flash | 神经网络参数 |  |
| RF1 | Regfile | 中间结果 |  |
| RF\_INST | Regfile | 指令 |  |
| RFC0 | Flip-flop | 存内计算指令的配置信息 |  |
| RFC1 | Flip-flop | 指令组的配置信息 |  |
| RF0 | Flip-flop | 神经网络输入激活 | 1536b |

处理器采用了单指令多数据（SIMD）的设计，单条指令控制的数据位宽为96比特，表示一个维度的向量，向量中每个元素的位宽都是6比特。处理器包含了3种模式，分别为配置模式、工作模式和休眠模式，模式之间的切换由状态机控制，各模式切换如图 4‑2所示。其中配置模式包括了4个子模式，分别为重置Flash模式、读Flash模式、写Flash模式和初始化模式。其中重置Flash模式实现对Flash存储器的全局擦除操作；读Flash模式依次读出Flash存储器中每个单元的阻值；写Flash模式完成对Flash存储器所有单元的编程操作；初始化模式完成片上寄存器组RF\_INST、RFC0、RFC1和RF1中数据的初始化。工作模式中处理器的流水线开始运行。在休眠模式中，处理器流水线停止运行，对张量处理单元和SPI 控制模块进行power gating操作，并把片上的4块寄存器组置入保持（Retention）模式，从而尽可能减少芯片的漏电功耗。由于采用了非易失性存储器Flash存储网络参数，因此在休眠模式中，处理器的漏电功耗主要由4块处于Retention模式的寄存器组产生。



图 4‑2处理器模式转换图

处理器以流式预测的方式进行工作，在对序列信号的处理过程中，每帧接收输入的特征向量，在工作模式运行神经网络算法对其进行处理，产生当前帧的推理结果。由于处理器的吞吐量很大，因此每帧神经网络的计算可以很快完成，即每帧工作模式持续的时间很短，剩下的绝大部分时间处理器从工作模式切换到休眠模式，等待下一帧特征向量的输入，如图 4‑3所示。



图 4‑3流式预测下模型转换图

### 控制模块设计

控制模块的结构如图 4‑4所示，包含有限状态机，译码器和存内计算核控制电路。有限状态机控制处理器不同状态之间的切换。译码器在工作模式控制模块中的译码器从RF\_INST中读取指令，对指令进行译码，生成各个模块的控制信号。存内计算核控制电路负责产生存内计算核在不同模式的控制信号。



图 4‑4控制模块图示

在本设计中，基于Flash存储器的存内计算核共包含4种模式，分别为擦除模式，读模式，写模式和计算模式，各模式说明如下表所示。为了减小芯片片上资源，部分控制电路在片外FPGA上实现。片外FPGA和芯片通过SPI协议进行通信。

表格 4‑2基于Flash的存内计算核各模式说明

|  |  |
| --- | --- |
| 模式名称 | 功能 |
| 擦除模式 | Flash整块擦除为低阻态 |
| 读模式 | 读出Flash指定的16个单元的阻值 |
| 写模式 | 对Flash指定地址进行有选择地写 |
| 计算模式 | 执行向量-矩阵乘法 |

### SPI模块

SPI模块包含SPI 从（Slave）模块和仲裁模块。SPI从模块负责和片外FPGA上的主（Master）模块进行数据传输。仲裁模块用于区分SPI从模块收发的不同数据类型。为了区分不同的数据类型，对不同类型的数据设置了帧头，仲裁模块根据SPI传输的二进制码元中的帧头来区分SPI协议传输的不同数据类型。

### 基于Flash存储器的存内计算核

存内计算核的架构如图 4‑5所示，其详细的工作原理在文献[20]中进行了阐述，本工作中主要对外围的寄存器、MUX2、位选电路等部分进行了设计，并设计了存内计算核的分块策略。存内计算核由5个部分组成：（1）1536比特的触发器RF0；（2）256个6选1 MUX2；（3）192个10选1 MUX3；（4）64KB Flash存储器阵列；（5）计算电路，包括16个整流电路，16个积分电路，16个10比特逐次逼近型模数转换器（SAR ADC）和16个位选电路。RF0中的1536个单比特触发器分为256组，每组包含6个触发器，每组触发器的输出端口和1个6选1 MUX2连接。256个6选1 MUX2的输出分别连接到Flash存储器阵列的256根字线上。64KB Flash存储器阵列的尺寸为，包含256根字线和2048根位线。Flash存储器阵列在位线维度上分为了10块，每块的大小为，所有的块共享计算电路。每块Flash存储器阵列在位线维度上又分为了16组，每组包含12列。同一根位线上的一组字线对应了12个Flash存储单元，一个6比特有符号参数使用差分编码的方式存储在12个Flash存储单元中。192个10选1 MUX3的1920个输入端口分别连接10块存储阵列包含的1920根字线，用来从10块Flash存储阵列中选出其中的一块，即从1920根位线中选出192根位线，然后接入计算电路。



图 4‑5存内计算核架构图

存内计算核支持的计算形式为向量-矩阵乘法，如公式xx所示。

其中向量X为存内计算核单次计算的输入向量，，向量X对应神经网络算法每层的输入激活；矩阵W对应神经网络算法中的参数，，矩阵W存储在10块Flash存储器阵列的其中一块中；向量表示向量X和矩阵W相乘并经过Relu操作后的原始输出向量，，其中的位宽由下式决定(减1表示由于采用了Relu激活函数，所以输出结果的值全为非负数，减去1比特符号位占用的空间)：

向量Y表示对向量进行量化后的最终输出结果，，对应神经网络算法中的输出激活；；为量化函数，量化的尺度由缩放因子S决定，S为量化的缩放因子，，n为自然数，表示舍去低n位，取中间6位，并对高位进行饱和操作。

Flash存储器阵列共分为10块，通过192个MUX3的选通作用，每次只选定其中一块来完成向量-矩阵乘。存内计算核每完成一次向量-矩阵乘法共包含8个阶段，如图 4‑6所示，每个阶段时长为300ns。RF0中的256个6比特触发器组依次存储向量X中每个元素。经过256个MUX2的选通作用，的最低有效位（Least Significant Bit，LSB）到最高有效位（Most Significant Bit，MSB）分6个阶段依次输入到Flash存储器阵列的字线上，对应下图中的BIT0-BIT5阶段，在每个阶段都会进行电容重置、电荷积分和电荷重分配。第7个阶段执行模数转换，对应下图ADC阶段，16个ADC并行完成模数转换，每个ADC输出10bit的转换结果。第8个阶段为下图中的BITselect阶段，其中16个位选模块并行从每个ADC的10bit结果中选出6bit，共96bit，对应存内计算核单次向量-矩阵乘的最终输出向量Y。



图 4‑6存内计算核单次计算各阶段示意图

在调用存内计算核完成一次向量-矩阵乘法操作前需要首先完成数据准备，即把输入向量X先写入RF0中，把权重W先存到Flash存储器阵列某一块中。完成数据准备后，存内计算核控制电路产生存内计算核需要的控制信号，执行该次向量-矩阵乘操作。



图 4‑7位选模块电路结构

位选电路的结构如图 4‑7所示，每个位选电路接收10bit的ADC输出结果A，然后根据信号右移一定位数得到低7bit，对应要选出的6bit数据S和要舍掉的最高有效位，然后完成对低位的四舍五入操作，即在S的基础上加入，并对高位进行饱和操作，即判断舍去的高位中是否包含1，若存在1，则饱和为6’b111111。

### 数字协处理器设计

数字协处理器包含了向量加/减法器和向量-常数除法器。其中向量加/减法器包含了16个13bit的行波进位加/减法器，16个加/减法器并行执行从而完成一次向量加/减法操作。在面向序列信号处理的神经网络算法中平均池化是一种常见的结构，选用13bit的高位宽加/减器可以保证在平均池化层的多帧（不超过128帧）特征累加过程中不会因为溢出而带来精度损失。向量-常数除法器包含了16个移位除法器，16个移位除法器并行执行从而完成一次向量-常数除法操作。每个移位除法器通过移位和舍入实现除数为2的整数次幂的除法操作，同时在训练神经网络时，其平均池化层中的除数也近似到了最临近的2的整数次幂，从而降低硬件执行除法的功耗，减少除法器的面积。

### 定制指令集设计



图 4‑8指令集

处理器的指令集如上表所示，指令共包含14bit，可分为4个字段，其中13-10比特表示指令的操作码，9-6比特存储指令访问RFC1寄存器组的地址，5-2比特表示指令访问RFC0/RF0/RF1寄存器组的地址，1-0比特表示指令的类型。指令在处理器的初始化模式中被预先加载到指令寄存器组RF\_INST中，在工作模式中译码器读取指令，对其译码并控制各模块的执行。指令共包含4级流水线，分别为“取指(IF)、译码(ID)、执行(EX)、写回(WB)”。

指令配合2个配置寄存器组RFC0和RFC1共同完成处理器的控制。各指令执行任务如下所述：

1. IMC指令，存内计算指令，调用存内计算核完成一次向量-矩阵乘法操作，结果存入RF1指定地址。由于IMC指令需要配合大量的控制信息，包括：积分时间的控制信息、Bit select module的选择信息、Flash存储器块的选择信息和偏置选择信息。14bit无法容纳IMC指令需要的所有信息，因此增加了配置寄存器组RFC0来配合存储这些信息。IMC指令中包含RFC0的地址字段，在IMC指令运行过程中会读取RFC0对应地址来得到所有的控制信息。
2. Vector add指令，向量加法指令，调用向量加/减法器完成2个向量的加法操作，结果暂存在psum寄存器中；
3. Vector sub指令，向量减法指令，调用向量加/减法器完成2个向量的减法操作，结果暂存在psum寄存器中；
4. Vector div指令，向量除法指令，调用向量除法器完成向量-常数除法操作，结果写入RF1指定地址；
5. Input to RF1指令，把接收到的SPI输入数据写入RF1指定地址中。
6. From RF1 to RF0指令，从RF1指定地址读出，写入RF0指定地址中。
7. From RF1 to psum指令，从RF1指定地址读出，写入psum寄存器中。
8. Write 0 to RF0，向RF0指定地址写入0向量。
9. Reset RF0，把RF0全部重置为0。
10. do-nothing，空指令。
11. stand-by，休眠指令，控制处理器进入休眠状态。

流水线优化。由于存内计算核的执行过程中需要30ns或者30ns倍数长的脉冲进行控制，因此我们从片外输入了频率为100/3MHz的时钟。如前文所述，存内计算核每完成一次向量-矩阵乘，需要2400ns，因此对应IMC指令的EX级需要2400ns。除IMC指令的EX级外，IMC指令的其它级和其它指令的所有级的关键路径延时都在30ns之内。如果把指令每一级都设置为2400ns，那么可以实现一个相对简单的4级流水线，但是这样会带来高延时。我们对流水线进行了优化，把流水线每级的时间设置为30ns，这种情况下IMC指令的EX阶段需要80级才可以完成。当IMC指令的EX阶段被执行时，流水线中的其他指令被阻塞，直到完成IMC指令的EX阶段后，被阻塞的其它指令恢复正常执行。由于流式预测中，神经网络算法具有串行执行的特性，因此生成的指令也存在前后 的数据依赖，即在IMC指令的EX级执行过程中，无法通过乱序执行的方式提前执行IMC指令后面的指令来提高吞吐量。经过流水线优化后，整体的延时大幅下降。由于IMC指令在编译后的指令中占20%左右，因此指令执行的时间缩短了80%左右。对流水线优化后，电路的频率提升，会带来时钟网络上功耗的增加，因此需要进行细粒度的时钟门控，特别针对流水线阻塞期间除存内计算核外的其他模块，本次设计中仍采用了流水线优化前的方案。

指令集中数据冒险，如当前一条“IMC”指令与后一条“From RF1 to RF0”指令间存在数据依赖时。“IMC”指令在WB级把执行的结果写回到RF1中，“From RF1 to RF0”指令在ID级读取RF1，若前后2条指令存在数据依赖，即访问RF1的地址相同，则会出现数据冒险。我们在控制模块中增加了旁路检测电路，当检测当有数据冒险时，后一条“From RF1 to RF0”指令直接从流水线寄存器中读取“IMC”指令EX级的结果，从而避免了流水线阻塞。



图 4‑9流水线优化

指令组设计。处理器运行神经网络算法对序列信号进行实时处理，每帧工作模式中均调用相同的指令。流式预测中，相邻帧之前存在数据复用，为了充分利用数据重用，相邻帧的指令对寄存器组RF1的访问地址需要一定的偏移。我们引入了指令组的设计，把特定的多条指令合并为一个指令组。每个指令组中包含3种指令：头指令、体指令和尾指令，通常由前一层对RF1进行写操作的指令和下一层对RF1进行读操作的指令构成一个指令组。每个指令组在编译后的指令中处于相邻位置。指令组设计配合公共计数器和配置寄存器组RFC1，在指令执行过程中自动记录地址偏移。在执行头指令时，使用从RFC1寄存器组中读出的地址去读RF1（读取RFC1的地址从头指令的RFC1地址地段中获得），并使用公共计数器对该地址加1，写入临时寄存器中；在执行体指令时，使用临时寄存器中记录的地址去访问RF1，并在执行过程中使用公共计数器对该地址加1，然后写入临时寄存器中；在执行尾指令时，使用临时寄存器中记录的地址读RF1，并在执行过程中使用公共计数器对该地址加1，然后把写回RFC1的指定地址中（写RFC1的地址从尾指令的RFC1地址地段中获得）。每层需要保存的中间结果保存在RF1中，每层在RF1中分配的地址个数（即存储空间大小）由该层网络大小决定，如公式所示。

下面以TC-Resnet8中的CONV0层的执行过程为例，阐述指令组的工作机制。Conv0层的映射如图 4‑10所示。Conv0在RF1中分配的地址个数为6，即从地址0到地址5。



图 4‑10TC-Resnet8 CONV0层执行图

如表所示，CONV0层经过编译后，共生成7条指令。

表格 4‑3 TC-Resnet8 CONV0层指令

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| layer | Instruction | Num | Type | Nth | (N+1)th |
| Init | Input to RF1 | 1 | header | 0 | 2 |
| Input to RF1 | 2 | body | 1 | 3 |
| Conv0 | From RF1 to RF0 | 3 | body | 2 | 4 |
| From RF1 to RF0 | 4 | body | 3 | 5 |
| From RF1 to RF0 | 5 | body | 4 | 0 |
| From RF1 to RF0 | 6 | body | 5 | 1 |
| From RF1 to RF0 | 7 | body | 0 | 2 |
| From RF1 to RF0 | 8 | tail | 1 | 3 |
| IMC | 9 | header | 6 | 7 |
| … | … | … | … | … | … |

表中init层表示每帧通过SPI协议把特征向量输入片上的过程，init层的前2条指令与Conv0层的前6条指令形成一个指令组，即表中阴影部分表示的8条指令。第n帧该指令组的执行过程中，头指令“Input to RF1”首先从RFC1中读出当前输入向量要写入RF1的地址，即地址0，然后通过公共计数器对该地址加1变为2写入临时寄存器中；init阶段的第二条“Input to RF1”为体指令，该指令把临时寄存器中存储的地址1作为输入向量要写入RF1的地址，然后通过公共计数器对该地址加1变为2写入临时寄存器中；conv0层的第1条“from RF1 to RF0”指令把临时寄存器中的地址2作为读取RF1的地址，然后通过公共计数器对该地址加1变为3写入临时寄存器中；conv0层的第2-5条“from RF1 to RF0”指令均为体指令，其操作过程与第1条“from RF1 to RF0”指令类似；conv0层的第6条“from RF1 to RF0”指令为尾指令，该指令把临时寄存器中的地址1作为读取RF1的地址，然后通过公共计数器对该地址加1变为2写回RFC1中。第(n+1)帧该指令组的执行过程中，头指令“Input to RF1”首先从RFC1中读出当前输入向量要写入RF1的地址，即地址2，后续过程与第n帧的执行过程类似。第n帧和第（n+1）帧该指令组数据流动如图 4‑11所示，其中存储在RF1中的蓝色部分表示相邻帧间重用的数据。至此，通过引入指令组的机制，相邻帧间的地址偏移可以在指令执行过程中自动记录下来，从而保证每帧可以对指令进行循环调用，在充分对相邻帧的数据进行重用的同时，大大减少了指令的存储空间。



图 4‑11 TC-Resnet8 CONV0相邻帧间指令执行情况及数据重用示意图

### 处理器模式介绍

前文提到处理器配置模式共包括了4个子模式，分别为重置Flash模式、读Flash模式、写Flash模式和初始化模式。每次对Flash存储器存储的参数进行更新，都需要依次经过重置模式，读模式和写模式。其中读模式可以把Flash存储器每个单元的阻值读出来，根据读模式读出的每个单元的阻值，可以对权重的差分编码进行优化，即根据读模式读出的阻值来优化写模式要写入的值，减少由于Flash阻值分布带来的误差，提高存内计算核计算的准确度。

根据Flash存储器的特性，对单元进行擦除后，单元为低阻态，对应数字1，低阻状态下由于器件阻值分布特性会产生误差；对单元进行编程后，单元为高阻态，对应数字0，高阻状态下由于单元的阻值非常大，因此几乎不产生误差。6bit有符号权重采用差分编码的方式存储在Flash的12个单元里，如图 4‑12所示，12个cell对应的阻值从左到右分别为，每个单元左下角的数字表示实际的阻值大小。每12个cell表示的理论数字值V由公式所示。



图 4‑12Flash擦除后单元阻值分布图

由于Flash的单元在低阻状态下存在阻值分布问题[20]，即单元为低阻时并不是对应标准的1，而是在1左右浮动，因此直接把二进制差分编码的权重写入Flash单元中，会存在误差，高位由于位的权重更大因此高位的阻值分布带来的影响更大。

如某12个cell在擦除模式中均被初始化为低阻，对应数值1，但实际的阻值分布如图 4‑12所示。若该12个单元中需要存储的数值为9，数值9本身的二进制编码为1001，因此只需要把除和外的10个单元编程到高阻（对应数值0），如图 4‑13所示。



图 4‑13Flash直接编程后单元阻值分布图

但是由于阻值分布的原因，实际上表示的值，误差为0.98；主要是由于为表示的高位的阻值误差较大造成的，若在编程之前已经通过读取知道了每个单元低阻状态下的误差，那么可以对原始的二进制差分编码进行优化，如把外的11个单元写为0，如图 4‑14所示，这样表示的值，误差为0。



图 4‑14Flash读取、编程后单元阻值分布图

## 基于分块串行存内计算架构的流式预测

之前基于存内计算的神经网络加速器设计，出于增大加速器吞吐量的考虑，加速器一般包含多个分立的存内计算核，每个存内计算核映射神经网络的一层参数，数据在各个存算核之间流动，各层串行/流水线操作，如文献[12]中使用了9个分立的存算核执行LSTM算法，其结构如下图 4‑15所示。

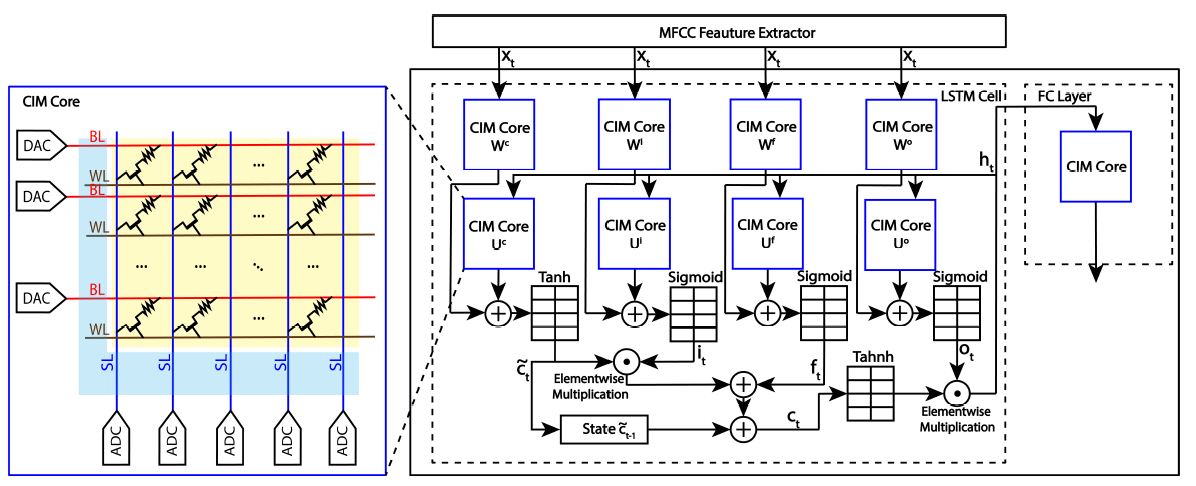


图 4‑15 分立存内计算核架构

在流式预测中，每帧各层的计算具有数据依赖，需要依次完成从第一层到最后一层的计算，多个分立存内计算核的架构难以发挥并行计算的优势。本文根据流式计算中各层串行执行的特点，采用集中式的存内计算核，对存内计算核进行分块。通过增加选通电路使得集中式存内计算核可以分块串行工作，实现对存储器阵列的外围电路的资源共用，如下图 4‑16所示。



图 4‑16 集中式存算核分块串行

文献[6]和文献[14]中对不同神经网络模型进行流式预测的方法进行了研究，实现了CNN、RNN等模型的流式预测。本文进一步对TC-Resnet网络的流式预测进行了研究，提出了TC-Resnet中残差层和平均池化层结构的流式预测方法，并基于分块串行存内计算核设计了处理器执行流式预测的方法。

### TC-Resnet模型流式预测优化方法

如上文所述，在使用神经网络模型对序列信号的实时处理过程中，相邻2帧的输入特征图只有1列特征向量是不同的，可以采用流式预测方式，执行增量计算而不是重新计算整个特征图，从而大大减少每帧的计算量，减少推理的延时和功耗。但是并不是所有的神经网络模型都适合来进行流式预测，如文献[1]中提出的原始的TC-Resnet模型，本工作对该网络模型中阻碍流式预测的结构进行了分析和优化。

在包含卷积结构的神经网络模型设计中，大量使用了padding 0的结构，即在输入特征图的边缘进行0填充，使用padding 0提高了神经网络捕捉特征图边缘信息的能力，并且使得神经网络中特征图维度的变化更有规律，有利于深层网络的设计。在有padding 0的情况下，与输入特征图相比，步长为1的卷积层输出特征图宽和高保持不变，步长为2的卷积层输出特征图宽和高减小一半。但在流式预测中，包含padding的卷积窗口的结果不能被重用。如图 4‑17TC-Resnet8的卷积层CONV0中，假设第n帧对编号为1-98帧的特征向量组成的特征图进行卷积，第(n+1)帧对编号为2-99帧的特征向量组成的特征图进行卷积。在存在padding 0的情况下，第n帧需要计算包含padding 0的卷积窗口为（0，0，1）、（0，1，2）、（97，98，0）和（98，0，0）；第（n+1）帧需要计算包含padding 0的卷积窗口为（0，0，2）、（0，2，3）、（98，99，0）和（99，0，0），这4个卷积窗口均不能重用，此外第（n+1）帧还需要对新的卷积窗口（97，98，99）进行卷积，因此第（n+1）帧进行增量计算需要执行5个卷积窗口。此外处理器不仅需要缓存第97帧和98帧特征向量，还需要缓存第2帧和第3帧，从而增加了缓存容量，并使得控制逻辑更加复杂。在去掉padding 0后，CONV0层执行增量计算只需要计算（97，98，99）一个卷积窗口，只需要缓存第97帧和98帧特征向量，从而减少了缓存空间和计算量，简化了控制逻辑。



图 4‑17 Padding 0对数据重用的影响图

接下来是对残差层进行优化，在去掉padding后，在步长为1的情况下，特征图经过一个卷积核宽度大于1的卷积层，宽度会减小；经过一个卷积核宽度为1的卷积层，特征图的宽度不变。由于残差层主路径上的卷积核宽度往往大于1，而分支路径上的卷积核宽度往往为1，因此主路径和分支路径的输出特征图维度不同，无法完成元素级相加，如下图所示。为了解决维度不同无法相加的问题，我们使用了2种方法，一种为去掉分支层，另一种是对卷积核的宽度进行调整，使得残差层主路径和分支路径的输出特征图维度相同，如图 4‑18所示。我们在TC-Resnet8上进行了实验，方案1的准确度为93.9%，方案2的准确度为94.8%，均为模型量化前准确度，即采用方案2可以提升0.9%的准确度。最终通过去掉padding 0，并且对卷积核的宽度进行调整，实现了对原始TC-Resnet模型的流式预测优化。



图 4‑18残差层优化前后结构对比图

通过使用模型流式预测优化方法，本工作中在文献[1]中提出的TC-Resnet结构的基础上，重新设计了适用于本处理器的轻量级TC-Resnet，并在语音关键词识别和心电信号处理任务上进行了测试。如图 4‑19（a）（b）分别为所设计的卷积步长为1和2的两种残差块结构。



图 4‑19残差块结构

基于上述残差块结构所设计的TC-Resnet网络模型结构如下图 4‑20所示。其中图（a）、（b）分别为用于语音关键词识别的步长为1和步长为2的TC-Resnet模型，图（c）表示用于心电图信号处理的步长为2的TC-Resnet模型。



图 4‑20 本工作设计的TC-Resnet结构图

上述3个网络训练和量化后的准确度如下表所示：

表格 4‑4 TC-Resnet准确度

|  |  |  |  |
| --- | --- | --- | --- |
| 网络 | 数据集 | 量化前准确度 | 量化后准确度 |
| TC\_Resnet8\_S1 | GSCD | 94.5% | 94.1% |
| TC\_Resnet8\_S2 | GSCD | 94.8% | 94.4% |
| TC\_Resnet8\_S2’ | ECG5000 | 93.6% | - |

本设计中采用TC\_Resnet8\_S1和TC\_Resnet8\_S2作为测试用例对处理器进行了性能测试。

### 处理器流式预测硬件执行方法

一维卷积层的流式执行。如图 4‑21所示，卷积层中每个卷积窗口的计算都可以转换为一次向量矩阵乘法。每个卷积窗口的输入激活转换为一个维度为的向量X；OC个卷积核转换为一个维度的矩阵W。此外每个卷积窗口的计算需要在输入激活和卷积核卷积结果（上加入偏置b，如公式所示。由于偏置b的位宽一般大于激活和权重的位宽，为了方便进行统一化的存储和计算，在模型量化中把每个偏置分解为，其中和的位宽和量化后的激活和权重一致。进一步把激活和权重的向量矩阵乘法与偏置与的乘法进行整合，则每个卷积窗口的计算最终转换为一次向量和矩阵乘法。在流式预测中，在步长为1的情况下，卷积层缓存之前帧输入的（KW-1）列输入激活进行复用，同时卷积层每帧会得到一列新的输入激活，如图中输入特征图中红色填充的第（IW+1）列所示，该列和缓存的（KW-1）列组成一个新的卷积窗口，如图 4‑21中输入特征图上的红色虚线框所示。卷积层每帧对该新的卷积窗口进行计算，通过转换为向量-矩阵乘法映射到存内计算核上执行，其中向量映射到RF0中，矩阵映射到Flash阵列的某个子块中，然后存内计算核启动计算得到每帧的输出，如图 4‑21中输出特征图中红色填充的第（OW+1）列所示。不考虑偏置的情况下，分批预测下卷积层每帧的计算量为，流式预测下卷积层每帧的计算量为，减少了OW倍。



图 4‑21一维卷积层流式执行图

全连接层的流式执行。全连接层本身激活和权重的计算形式为向量-矩阵乘，如下图 4‑22所示，在此基础上对偏置的计算也进行整合，如上公式所示，从而把全连接层一次完整的计算也转换为一次向量矩阵乘。全连接层每帧会输入新的输入特征图，每帧均转换为向量-矩阵乘映射到存内计算核上执行，映射方式和卷积层相同，从而每帧都产生新的输出特征图，即每帧生成推理结果。不考虑偏置的情况下，分批预测和流式预测下全连接层每帧的计算量均为。



图 4‑22全连接层流式执行图

平均池化层的流式执行。在常规的分批预测中，平均池化层对整个输入特征图在IW维度上进行累加得到一个维度的和向量，然后该向量和常数IW进行向量-常数除法操作得到池化结果。在流式预测中，对和向量以及之前IW帧的输入激活进行缓存和复用，每帧会得到一列新的输入激活，如下图 4‑23输入特征图中的红色填充的第（IW+1）列所示，每帧在和向量的基础上减去保存的第1列的输入激活，然后加上第（IW+1）列的输入激活，再进行向量-常数除法操作，得到当前帧的池化结果。池化层的计算由数字协处理器完成。分批预测下平均池化层每帧的加/减法数为，流式预测下池化层每帧的加/减法数为，加法/减法数量减少了倍，除法数保持不变。



图 4‑23平均池化层流式执行图

残差层的流式执行，残差层中包含了3个一维卷积层，每个一维卷积层均单独采用前文所述的流式方法执行。

前文介绍了一维卷积层、残差层、全连接层和平均池化层的流式执行方法，下图 4‑24以TC-Resnet8为例，展示了完整的网络流式预测的方法，下图中每层每帧只计算红色虚线框所示的部分计算。



图 4‑24 TC-Resnet流式预测图

在TC-Resnet上计算量减少了90%以上，如下表所示（其中TC-Resnet\_S2表示步长为2的TC-Resnet网络，TC-Resnet\_S1表示步长全为1的TC-Resnet网络）。

表格 4‑5分批预测与流式预测对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | 分批预测MAC数 | 流式预测MAC数 | 减少比例 |
| TC-Resnet8\_S2 | 0.64M | 50K | 90.6% |
| TC-Resnet8\_S1 | 2.16M | 26.2K | 98.8% |

前面介绍了神经网络不同层流式预测下的硬件执行方式，下面介绍从整个网络在硬件上的执行流程。卷积层和全连接层的计算均转换为了向量-矩阵乘法，转换后的参数矩阵大小为需要存储在存内计算核中。存内计算核分为了10块，每块可容纳的参数矩阵大小为。整个网络的映射过程从第一层到最后一层依次进行，每个卷积层/全连接层都根据参数矩阵的大小在存内计算核中占用一定的空间。为了提高存内计算核的利用率，可以把多个维度较小的参数矩阵存储在一块存内计算核中；对于维度较大的参数矩阵，则使用多块存内计算核进行存储。假设网络存在n层（不包括平均池化层），并映射到了n个Flash子块中，分别为，在整个网络的流式预测过程中，依次选通串行计算。

## 实验验证与分析

上文介绍了基于存内计算的流式神经网络处理器的电路设计，本节对处理器进行电路仿真，介绍了本文采用的实验环境以及验证和功耗分析使用的方法，分析了各个模块的功耗分布，最后与其它支持语音关键词识别的神经网络加速器进行了对比。

### 实验环境

本文设计的流式神经网络处理器采用40nm工艺实现。本设计为数模混合电路，本文主要集中在对数字部分设计和实现。其中模拟部分主要由Flash存储器、整流电路、积分电路和ADC组成。对于数字部分，首先根据处理器运行的算法需求进行RTL设计，然后使用功能仿真工具（VCS）进行RTL波形仿真，验证RTL功能正确；然后使用综合工具（Design Compiler，DC）进行门级网表的综合，并使用形式验证工具（Formality）对综合的正确性进行验证；接下来使用（Prime Time，PT）对门级网表进行了仿真，得到了描述门级网表翻转率的波形文件；最后使用功耗分析工具（Prime Time PX，PTPX）对门级网表进行功耗分析。对于模拟部分的存内计算核电路，使用仿真工具（Virtuoso）进行功耗分析。

40nm工艺中提供了高阈值（High Voltage Threshold，HVT）电压、常规阈值（Regular Voltage Threshold，HVT）电压、低阈值（Low Voltage Threshold，HVT）电压等选项。随着阈值电压的降低，电路器件的延时越低，漏电功耗越高。由于本设计中存内计算核需要最小变化粒度为30ns的控制信号，因此整个芯片最快的时钟频率为100/3MHz，该时钟从片外引入。由于频率较低，因此本芯片对于电路器件延时的要求较低，但是由于应用场景的限制，本设计要求功耗特别是静态漏电功耗尽可能低，因此本设计中选用了高阈值电压。本设计在TT 25℃ 1.1V工艺角下进行性能测试。

处理器的版图如图 4‑25所示。



图 4‑25芯片版图

### 语音关键词识别任务功耗分析

当运行语音关键词识别任务时，我们测试了2种工作模式：每帧预测模式和每8帧预测模式。每帧预测模式运行的网络模型为卷积步长全为1的TC-Resnet8\_S1，每帧（10ms）均产生一次预测结果；每8帧预测模式运行的网络模型为包含3个卷积步长为2的卷积层的TC-Resnet8\_S2，由于步长的累积，每8帧（）产生一次预测结果。

在每帧预测模式下，TC-Resnet8中每一层在每帧均运行一次,该模式下的功耗分布如表所示，总功耗为2.31uw。

表格 4‑6每帧预测模式各模块功耗组成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 电路类型 | 功耗/uw | | | 占比 |
| 动态功耗 | 静态功耗 | 总计 |
| 组合逻辑 | 0.327 | 0.093 | 0.421 | 18.2% |
| 时序逻辑 | 0.03 | 0.218 | 0.249 | 10.8% |
| 存内计算核 | 0.181 | 0.13 | 0.312 | 13.5% |
| 存储器 | 0.357 | 0.974 | 1.331 | 57.5% |
| 整体 | 0.896 | 1.416 | 2.312 | 100.00% |



图 4‑26每帧预测模式下功耗分布图

每8帧预测模式运行的网络模型为包含卷积步长为2的TC-Resnet8\_S2。8帧预测模式下，TC-Resnet8中由于存在步长为2的卷积层，由于步长的累加，各层之间运行的频率会出现差异，层数越靠后的层运行的频率越低。CONV0层每帧运行1次，RES1层每2帧运行一次，RES2层每4帧运行一次，RES3层、平均池化层和全连接层每8帧运行一次。每8帧预测模式的功耗分布如表所示，总功耗为1.6uw。

表格 4‑7每8帧预测模式各模块功耗组成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 电路类型 | 功耗/uw | | | 占比 |
| 动态功耗 | 静态功耗 | 总计 |
| 组合逻辑 | 0.111 | 0.092 | 0.202 | 12.7% |
| 时序逻辑 | 0.01 | 0.216 | 0.226 | 14.1% |
| 存内计算核 | 0.072 | 0.044 | 0.116 | 7.3% |
| 存储器 | 0.142 | 0.915 | 1.057 | 65.9% |
| 整体 | 0.335 | 1.267 | 1.602 | 100.00% |



图 4‑27每8帧预测模式下功耗分布图

与每帧预测模式相比，每8帧预测模式中每层运行的频率更低，因此动态功耗降低了62.6%，同时工作模式持续的时间更短，休眠模式持续的时间更长，因此静态功耗也有部分下降。但是由于每8帧预测模式中预测的频率更低，因此在实时关键词检测过程中，可能会出现错词率升高的情况。

### 与其他神经网络处理器性能对比

当运行语音关键词识别任务时，和其他神经网络处理器的性能对比如下表所示。出于公平对比的考虑，我们把其他工作的性能量化到了40nm，1.1V，量化的方式如表所示。同时表格中中的功耗仅包含神经网络处理器部分（对于文献[6][4]已经减去了预处理MFCC部分的功耗）。我们使用每帧预测模式下的性能和其他的工作进行对比。

表格 4‑8性能对比表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | [6]ISSCC 2020 | [4]TCAII 2021 | [8]TVLSI 2021 | [7]TCAD 2020 | 本设计 |
| 工艺 | 28 | 40 | 65 | 22 | 40 |
| 算法 | DSCNN | LSTM | TCN | TC-Resnet | TC-Resnet |
| 数据集 | GSCD | | | | |
| 识别关键词个数 | 2 | 10 | 10 | 10 | 10 |
| 准确度 | 94.60% | 90.60% | 93.30% | 93.09% | 94.10% |
| 位宽 | 8/1 hybrid | 8 | 8 | 8/4 hybrid | 6 |
| 片上存储器容量 | 2KB regfile | 9.28KB SRAM | 128KB SRAM | 74.125KB SRAM | 64KB FLASH 2.9KB regfile |
| 频率 | 40KHz | 400KHz | 31-27kHz | 250KHz | 30MHz |
| 电压(V) | 0.41 | 0.6 | 1.2 | 0.8 | 1.1 |
| 面积(mm2) | 0.23 | 0.16 | - | 0.2 | 2.11 |
| 归一化面积(mm2)a | 0.47 | 0.16 | - | 0.66 |
| 延时(us) | 64000 | 32000 | 128000 | 100000 | 184.8 |
| 功耗(uw) | 0.17 | 1.84 | 7.15 | 8.2 | 2.32 |
| 归一化功耗(uw)b | 1.75 | 6.18 | 3.7 | 28.19 |
| 可配置性 | No | No | TCN | TC-Resnet | TC-Resnet |
| a:归一化面积=面积 \* (工艺/40nm)2  b:归一化功耗=功耗 \* (工艺/40nm) \* (电压/1.1)2 | | | | | |

在准确度对比上，[6]中准确度为94.6%，但是仅能识别2个关键词；[4][8][7]和本工作均可以识别10个关键词，其中[4]的准确度为90.6%，[8][7]的准确度在93%以上，本工作的准确度为94.1%。

由于准确度和识别的关键词个数反应了运行的神经网络模型的大小，越小的神经网络模型计算量越少，同时需要的片上存储空间越小，从而带来显著的功耗下降。出于公平比较的考虑，本设计和准确度接近的工作进行了功耗对比，即识别10个关键词以上，并且准确度在90%以上的工作[4][8][7]。与[4][8][7]相比，本设计分别实现了1.6倍、2.7倍和12.2倍的功耗减小。

此外[6]和[4]均为语音关键词检测专用架构设计，灵活性低。[8]和[7]具有可配置性，可以分别运行TCN类和TC-Resnet类的网络。本工作设计了定制指令集，可以运行TC-Resnet类的网络，实现语音关键词识别等序列信号处理功能。

## 本章小节

本章介绍了处理器的整体架构设计，包括控制模块、SPI模块、存内计算核等模块的设计，以及专用指令集设计。然后介绍了本设计中采用的流式预测优化，通过执行增量计算，减少了90%以上的冗余计算。最后进行了实验分析，和其他具有相近准确度的工作相比，本工作实现了1.6倍以上的功耗减小。

# 总结与展望

## 总结

神经网络算法广泛应用在物联网场景中，物联网设备受限于电池等因素，对功耗有着严格的要求，此外延时和准确度等性能对用户体验十分重要。本文设计了一种基于存内计算的流式神经网络处理器，当运行语音关键词识别任务时，可识别10个关键词，准确率达94.1%-94.4%，功耗为1.6-2.32uw，延时为172.8-184.8us，与其他达到相近识别效果的工作相比（识别关键词数大于等于7个，准确度在90%以上），功耗降低了1.6倍以上。

本文的主要工作包括：

1）设计了基于存内计算核的动态缩放因子调节方法，配合使用多条调整方法，可以实现动态的缩放因子调整。

2）设计了基于存内计算的流式神经网络处理器架构，存内计算核和数字协处理器共同完成神经网络中的算术操作。

1. 设计了专用指令集，支持一维卷积层、全连接层、平均池化层等操作。配合专用指令集设计了神经网络的映射方案，目前处理器支持对TC-Resnet类网络的运行，可以实现包含语音关键词识别在内的序列信号处理任务。
2. 进行了流式预测优化，处理器每帧执行增量计算，充分利用数据重用，减少了90%以上的计算量。

## 展望

（1）降低存储器的漏电功耗。面向物联网应用的超低功耗神经网络加速器中往往运行轻量级网络，因此动态功耗占比相对较低，相比其他场景的神经网络加速器，需要更加关注静态功耗的优化。本文中使用了非易性Flash存储器进行参数的存储，使用了寄存器组进行中间结果的存储，尽管寄存器组绝大部分时间被置入了保持模式，但仍然产生了较大的漏电功耗。下一步可以对片上存储器进行优化，如将片上寄存器组替换为非易失性存储器，或者对寄存器组进行定制化设计。

（2）神经网络中的参数具有一定的冗余，可以使用剪枝等网络压缩方法进一步减少参数量和计算量，配合稀疏编码减少存储空间。由于存内计算核支持规则的向量-矩阵乘法，不够灵活，无法支持细粒度的剪枝方法，因此本设计中使用了粗粒度的结构化剪枝。使用细粒度的剪枝可以进一步提高网络的稀疏度，可以进一步优化存内计算核的结构使其支持细粒度剪枝方法。

（3）本工作中设计的指令集目前成功实现了对TC-Resnet类网络的支持，并在语音关键词识别任务上进行了实验，对于其他的网络形式，以及其他的序列信号如脑电信号、步态信号等可以做进一步的探索，挖掘更多的应用场景。

# 参考文献

1. Choi S, Seo S, Shin B, et al. Temporal convolution for real-time keyword spotting on mobile devices[J]. arXiv preprint arXiv:1904.03814, 2019.
2. Ingolfsson T M, Hersche X W M, Burrello A, et al. Ecg-tcn: Wearable cardiac arrhythmia detection with a temporal convolutional network[C]//2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2021: 1-4.
3. Ingolfsson T M, Hersche M, Wang X, et al. EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain–Machine Interfaces[C]//2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2020: 2958-2965.
4. Chong Y S, Goh W L, Nambiar V P, et al. A 2.5 uW KWS Engine with Pruned LSTM and Embedded MFCC for IoT Applications[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021.
5. Zhang S, Huang K, Shen H. A robust 8-bit non-volatile computing-in-memory core for low-power parallel MAC operations[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(6): 1867-1880.
6. W. Shan et al., “14.1 a 510nW 0.41 V low-memory low-computation keyword-spotting chip using serial FFT-based MFCC and binarized depthwise separable convolutional neural network in 28nm CMOS,” in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, Feb. 2020, pp. 230–232.
7. Bernardo P P, Gerum C, Frischknecht A, et al. Ultratrail: A configurable ultralow-power tc-resnet ai accelerator for efficient keyword spotting[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 4240-4251.
8. Giraldo J S P, Jain V, Verhelst M. Efficient Execution of Temporal Convolutional Networks for Embedded Keyword Spotting[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021.
9. Guo R, Liu Y, Zheng S, et al. A 5.1 pJ/neuron 127.3 us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS[C]//2019 Symposium on VLSI Circuits. IEEE, 2019: C120-C121.
10. Dbouk H, Gonugondla S K, Sakr C, et al. A 0.44-J/dec, 39.9-s/dec, Recurrent Attention In-Memory Processor for Keyword Spotting[J]. IEEE Journal of Solid-State Circuits, 2020.
11. Zhou C, Redondo F G, Büchel J, et al. AnalogNets: ML-HW Co-Design of Noise-robust TinyML Models and Always-On Analog Compute-in-Memory Accelerator[J]. arXiv preprint arXiv:2111.06503, 2021.
12. Schaefer C J S, Horeni M, Taheri P, et al. LSTMs for Keyword Spotting with ReRAM-based Compute-In-Memory Architectures[C]//2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2021: 1-5.
13. Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv:1804.03209.Retrieved from <https://arxiv.org/abs/1804.03209>.
14. Mittermaier S, Kürzinger L, Waschneck B, et al. Small-footprint keyword spotting on raw audio data with sinc-convolutions[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 7454-7458.
15. Rose R C, Paul D B. A hidden Markov model based keyword recognition system[C]//International Conference on Acoustics, Speech, and Signal Processing. IEEE, 1990: 129-132.
16. Baljekar P, Lehman J F, Singh R. Online word-spotting in continuous speech with recurrent neural networks[C]//2014 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2014: 536-541.
17. Chen G, Parada C, Heigold G. Small-footprint keyword spotting using deep neural networks[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 4087-4091.
18. Sainath T, Parada C. Convolutional neural networks for small-footprint keyword spotting[J]. 2015.
19. Fernández S, Graves A, Schmidhuber J. An application of recurrent neural networks to discriminative keyword spotting[C]//International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2007: 220-229.
20. Zhang Y, Suda N, Lai L, et al. Hello edge: Keyword spotting on microcontrollers[J]. arXiv preprint arXiv:1711.07128, 2017.
21. Coucke A, Chlieh M, Gisselbrecht T, et al. Efficient keyword spotting using dilated convolutions and gating[C]//ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019: 6351-6355.
22. Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.
23. Zhou A, Yao A, Guo Y, et al. Incremental network quantization: Towards lossless cnns with low-precision weights[J]. arXiv preprint arXiv:1702.03044, 2017.
24. Szymon Migacz. 8-bit Inference with TensorRT. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit- inference-with-tensorrt.pdf
25. VIVIENNE SZE, Yu-Hsin Chen, Tien-Ju Yang and Joel S, Efficient Processing pf Deep Neural Networks: A Tutorial and Survey[C], in Proceedings of the IEEE, 2017: 2295-2329.
26. Qin H, Gong R, Liu X, et al. Binary neural networks: A survey[J]. Pattern Recognition, 2020, 105: 107281.
27. Giraldo, J. S. P., and Marian Verhelst. "Hardware acceleration for embedded keyword spotting: Tutorial and survey." ACM Transactions on Embedded Computing Systems (TECS) 20.6 (2021): 1-25.
28. Zhang Y, Huang K, Xiao R, et al. An 8-bit In Resistive Memory Computing Core with Regulated Passive Neuron and Bit Line Weight Mapping[J]. arXiv preprint arXiv:2008.11669, 2020.
29. Xiao R, Huang K, Zhang Y, et al. A Low Power In-Memory Multiplication andAccumulation Array with Modified Radix-4 Inputand Canonical Signed Digit Weights[J]. arXiv preprint arXiv:2101.02419, 2021.
30. Wang J, Wang X, Eckert C, et al. A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing[J]. IEEE Journal of Solid-State Circuits, 2019, 55(1): 76-86.
31. Noel J P, Pezzin M, Gauchi R, et al. A 35.6 TOPS/W/ 3-stage pipelined computational SRAM with adjustable form factor for highly data-centric applications[J]. IEEE Solid-State Circuits Letters, 2020, 3: 286-289.
32. Zhou K, He Y, Xiao R, et al. A Customized NoC Architecture to Enable Highly Localized Computing-On-the-Move DNN Dataflow[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021.
33. Jia H, Ozatay M, Tang Y, et al. 15.1 A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing[C]//2021 IEEE International Solid-State Circuits Conference (ISSCC). IEEE, 2021, 64: 236-238.
34. Rybakov O, Kononenko N, Subrahmanya N, et al. Streaming keyword spotting on mobile devices[J]. arXiv preprint arXiv:2005.06720, 2020.
35. Li Y, Tagliasacchi M, Rybakov O, et al. Real-Time Speech Frequency Bandwidth Extension[C]//ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021: 691-695. （STREAMING）
36. Liu C C, Chang S J, Huang G Y, et al. A 10-bit 50-MS/s SAR ADC with a monotonic capacitor switching procedure[J]. IEEE Journal of Solid-State Circuits, 2010, 45(4): 731-740.

# 致谢

三年的硕士研究生生涯转瞬即逝，还记得初入实验室做毕业设计时的陌生，不知不觉已经走过了三年，到了要说再见的时候。在此，感谢一路上细心指导的老师，感谢给我帮助和支持的家人和朋友们，是你们给我力量，伴我前行。

首先感谢沈海斌老师和黄科杰老师在科研工作上的悉心指导。沈老师开阔的视野，精益求精的态度、渊博的学识让我受益良多，感谢沈老师在我碰到困难时的帮助。黄老师在神经网络加速器上具有前瞻性见解，他大胆创新的精神激励和感染了我，感谢一直以来的指导和帮助。

其次，感谢朱朝阳、周资群、郑瑞沣、章烨炜等师兄的帮助，感谢同届的刘一铭、厉军城、赵智洋、陆凯晨、肖蕊、李鑫的陪伴和鼓励，以及陈杨、王文威、李永根、顾峻川等师弟的支持和配合。感谢室友胡永睿、肖晓辉、张泽琪，谢谢你们的理解和包容。很高兴与你们在浙大相识。

最后，感谢我的家人。感谢父亲和母亲对我一直以来的鼓励和关怀，你们的理解和支持是我前行路上最坚强的后盾。

# 在读期间取得的科研成果

期刊论文

1. Zhu C, Huang K, Yang S, et al. An efficient hardware accelerator for structured sparse convolutional neural networks on FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(9): 1953-1965.

已受理的发明专利

1. 黄科杰，杨树园，沈海斌，一种低功耗存储器内计算处理器架构（发明专利，申请号：202110265116.8）
2. 黄科杰，杨树园，陆凯晨，沈海斌，一种语音关键词检测专用芯片（发明专利，申请号：202110111358.1）