软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 神经形态计算——从脉冲神经网络到边缘部署*

俞诗航[1,2], 易梦军[1,3], 吴 洲[1,2], 申富饶[1,3], 赵 健[4]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](南京大学 计算机科学与技术系,江苏 南京 210023)

[3](南京大学 人工智能学院,江苏 南京 210023)

[4](南京大学 电子科学与工程学院,江苏 南京 210023)

通讯作者: 申富饶, E-mail: frshen@nju.edu.cn; 赵健, E-mail: jianzhao@nju.edu.cn

**摘 要**: 受生物神经系统启发, 神经形态计算的概念于 20 世纪 80 年代被提出, 旨在模拟生物大脑的结构和功能, 实现更高效、更具生物合理性的计算方式. 作为神经形态计算的代表模型, 脉冲神经网络因其脉冲稀疏性, 事件驱动性、生物可解释性以及硬件契合性等优势, 在资源严格受限的边缘智能任务中得到了广泛应用. 本论文针对脉冲神经网络的边缘部署情况进行了梳理和汇总, 首先从脉冲神经网络模型自身的原理出发, 论述了脉冲神经网络的高能效计算方式以及巨大的边缘部署潜力. 然后介绍了当下常见的脉冲神经网络硬件实现工具链, 并重点对脉冲神经网络在各类神经形态硬件平台的部署情况做了详细的整理与分析. 最后, 考虑到硬件故障行为已发展为当下研究中不可避免的问题, 本文对脉冲神经网络边缘部署时的故障与容错研究进行了概述. 本文从软件模型原理到硬件平台实现, 全面系统地介绍了神经形态计算的最新进展, 分析了脉冲神经网络边缘部署时遇到的困难与挑战, 并针对这些挑战给出未来可能的解决方向.

**关键词**: 神经网络;脉冲神经网络;神经形态计算;边缘智能

**中图法分类号**: TP311

中文引用格式:

英文引用格式:

## Neuromorphic Computing - From Spiking Neural Networks to Edge Deployment

YU Shi-Hang[1,2], YI Meng-Jun[1,3], WU Zhou[1,2], SHEN Fu-Rao[1,3], ZHAO Jian[4]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2] (Department of Computer Science and Technology, Nanjing 210023, China)

[3] (School of Artificial Intelligence, Nanjing 210023, China)

[4] (School of Electronic Science and Engineering, Nanjing 210023, China)

**Abstract**: Inspired by the biological nervous system, the concept of neuromorphic computing was introduced in the 1980s. It aims to mimic the structure and function of the biological brain to achieve efficient and biologically plausible computation. Spiking neural networks (SNNs), as a representative model of neuromorphic computing, have been widely used in edge intelligence tasks with strict resource constraints due to their advantages of spike sparsity, event-driven operation, biological interpretability, and hardware compatibility. This paper compiles and summarises the edge deployment of spiking neural networks. First, it discusses the energy-efficient computation of SNNs and their huge potential for edge deployment, focusing on the principles of the SNN model itself. The paper also introduces the common hardware implementation toolchain for SNNs and provides a detailed collation and analysis of SNN deployment on various types of neuromorphic hardware platforms. Finally, considering that hardware fault behavior has become an unavoidable issue in current research, an overview of

* 基金项目: 国家电网有限公司科技项目资助“数字化安全管控边缘计算装置自主可控关键技术研究及应用”(5700-202319302A-1-1-ZN)

收稿时间: 修改时间: 采用时间:

fault and fault tolerance research when deploying SNNs at the edge is also presented. This paper offers a comprehensive and systematic summary of recent advances in neuromorphic computing. It analyses the difficulties and challenges encountered in the edge deployment of SNNs, from software model principles to hardware platform implementation, and suggests possible directions for future solutions to these challenges.

**Key words**: Neural Networks; Spiking Neural Networks; Neuromorphic Computing; Edge Intelligence

近年来, 人工智能(Artificial Intelligence, AI)已经成为最重要的科学研究领域之一, AI 技术以我们过去认为不可能的方式改变了世界[1,2]. 人工神经网络(Artificial Neural Network, ANN)作为人工智能领域最著名的方法, 在计算机视觉[3,4], 自然语言处理[5,6], 智能驾驶[7,8], 医学诊断[9,10]等各个领域都取得了卓越的表现.

根据计算单元的特点, 神经网络可以分为三代. 第一代神经网络由感知机等阈值单元组成, 执行阈值运算并输出二值结果[11]. 但由于其结构过于简单, 被证明仅具备有限的功能, 只能进行线性分类, 甚至无法解决最简单的“异或”逻辑问题[12]. 为了解决线性不可分问题, 第二代神经网络是通过在计算单元的输出上应用连续激活函数构造的, 反向传播(Back Propagation, BP)机制使其能够计算一组连续的输出值, 且对于连续型前馈神经网络, 只要有足够数量的隐藏神经元, 就可以以任意精度逼近任何复杂的连续映射[13-15]. 这类拥有隐藏层和激活函数的基于联结主义的多层人工神经网络被称为第二代神经网络. 作为第二代神经网络最出色的代表, 深度神经网络(Deep Neural Network, DNN)出现于 20 世纪 80 年代中期[16], 自 2006 年以来引领了人工智能的发展[17]. 然而, 随着近年来边缘智能领域的迅速发展, 传统 ANN 暴露出许多缺点. 首先, 过去 ANN 的成功依赖于大量的训练数据和庞大的 GPU 计算资源, 以 GPT-3 为例, 该模型拥有超过 1700 亿个参数和 45TB 的训练数据[18]. 其次, 传统 ANN 在生物学上缺乏可解释性, 并且在神经元内部缺乏动态机制, 对时空信息的处理能力较弱[19]. 最后, ANN 模型的计算能效较低, 不容易在硬件尤其是便携式设备上实现. 这些缺陷促使了第三代神经网络——脉冲神经网络(Spiking Neural Network, SNN)的出现[20]. 研究表明, 在 45nm 技术节点上, 一个 ANN 神经元的乘积累加动作会消耗 4.6 pJ, 而一个脉冲神经元的累加动作仅需消耗 0.9 pJ[21], 这使得 SNN 更适合部署在低功耗边缘设备.

神经形态计算是一种受到生物神经系统启发的计算范式, 其试图基于大脑中神经元的脉冲事件, 在时间和空间分布上模仿神经元和突触的功能, 实现高能效的计算. SNN 与生物神经系统非常相似, 使用离散值(脉冲)来编码和处理数据, 提供了一种高效且低功耗的计算方案[22], 与神经形态计算的核心理念一致. 在生物神经元中, 当突触前刺激导致的膜电位变化的总和超过阈值时, 就会产生脉冲. 脉冲产生的速度和脉冲序列的时间模式携带着相关外部刺激和正在进行的运算信息. 在 SNN 中, 脉冲神经元只有当新的输入脉冲到来时才会进行处理, 这使得 SNN 在本质上更符合生物学原理, 表现出和真实神经回路中相同的有利特性, 如模拟计算能力、低功耗、快速推理、事件驱动、在线学习和大规模并行等. 这些优势使得 SNN 更加具备在边缘设备上部署的潜力[23]. 目前, SNN 是神经形态计算的核心模型, 有望充分发挥各类神经形态硬件平台的潜力, 实现“近存计算”, 打破传统冯诺依曼计算架构的瓶颈[24].

在过去的传统架构中, 所有待处理数据都被发送到数据中心进行处理和计算. 如今, 人工智能物联网(Artificial Internet of Things, AIoT)的发展和数据量的急剧增加给数据中心带来了沉重的计算负担、较长的响应延迟和过高的运行能耗. 因此, 在边缘设备上进行智能计算的需求日益增长[25]. 边缘部署的优势在于能够在接近数据源的地方进行快速的本地计算, 减少了数据传输和云端计算的延迟, 并提供实时的响应能力. 与传统 ANN 相比, SNN 在边缘设备中训练和部署时具有一些优势. 目前, 传统 ANN 的有效训练通常需要使用高能耗的 GPU, 但是在边缘设备中, 计算功耗往往会有严格的限制, 这使得复杂的 ANN 很难有效部署. 而脉冲神经元之间的突触连接并不总是活跃的, 因为神经元只能发射或不发射脉冲. 这种现象被称为 "脉冲稀疏性", 能让计算次数减少, 进而降低功耗和延迟. 实验证明, SNN 在推理过程中的能效几乎可以达到功能相同的 ANN 的 9 倍[26]. 此外, SNN 还拥有着 ANN 不具备的驱动超低功耗神经形态芯片的能力[27]. 目前, 已经有相当一部分的工作成功将 SNN 部署在神经形态硬件平台上, 并应用于边缘智能场景, 如目标检测[28-30], 文本分类

[31], 情感识别[32], 智能医疗[33-35], 机器人设计[36,37]等. 此外, 基于 SNN 中脉冲序列在时间上稀疏的特性, 可以创建事件驱动的低能耗脉冲硬件[38]. 随着 SNN 在边缘智能中扮演的角色越来越重要, 如何将 SNN 有效地部署在各类边缘硬件设备上也成为了一个值得重点关注的领域.

本文主要对神经形态计算的代表, 即 SNN 的边缘部署情况进行分析和总结, 第 1 章介绍 SNN 的基本原理, 包括脉冲神经元, 编解码算法, 训练算法和网络架构. 第 2 章介绍将 SNN 部署到神经形态硬件时常用的工具链. 第 3 章对 SNN 在各类神经形态硬件上的部署情况进行梳理, 包括现场可编程门阵列(Field Programmable Gate Array, FPGA), 神经形态芯片和非易失性存储器(Non-Volatile Memory, NVM). 第 4 章概述 SNN 在边缘硬件设备部署时面临的故障与容错问题. 第 5 章对全文进行总结并对未来研究方向进行展望. 图 1 展示了论文的总体框架图.

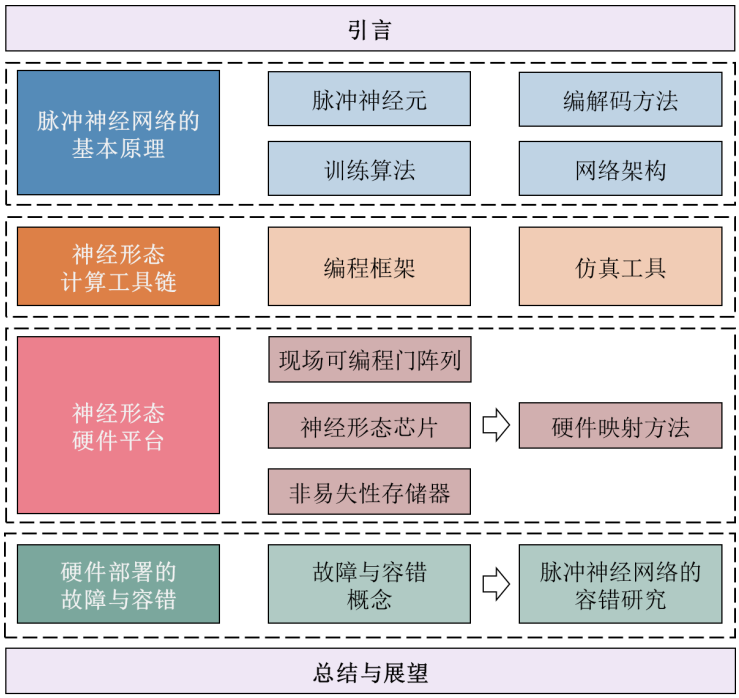


图 1 总体框架图

## 1 脉冲神经网络

本章将讲述 SNN 的基本原理, 包括常见的脉冲神经元模型、脉冲编码机制、SNN 的学习训练方式以及网络整体的拓扑结构. 同时, 还将论述 SNN 在这四个方面的硬件实现现状与问题, 并给出可能的解决方案.

### 1.1 脉冲神经元

神经元是神经系统中的基本结构和功能单元, 生物神经元能感知环境的变化, 再将信息传递给其他神经元. 传统的人工神经元仿照生物神经元的结构和工作原理, 对所有的信息输入进行加权整合, 再通过激活函数传递给下一神经元. 脉冲神经元与之不同, 其使用离散脉冲序列进行神经元间通信, 在机制上和生物神经元是类似的, 拥有更优秀的生物合理性. 图 2 比较了脉冲神经元和人工神经元的结构与工作模式.

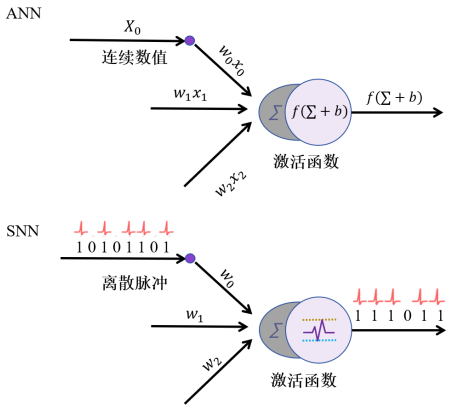


图 2 人工神经元与脉冲神经元

脉冲神经元是构成 SNN 的基本单元, 其主要功能是对脉冲序列蕴含的信息进行整合和传输, 本节将介绍当下最流行的几种脉冲神经元模型.

### 1.1.1 Hodgkin-Huxley 模型

Hodgkin-Huxley(H-H)模型是第一个描述脉冲神经元动作电位如何启动和传播的生物模型[39], 也是生物学上最合理的脉冲神经元模型, 能准确捕捉许多真实神经元的动态[40], H-H 模型给出了通过膜电位的电流的数学描述, 可以用下方公式进行计算:

$$I = C\frac{dV}{dt} + G_{Na}m^3h(V - V_{Na}) + G_kn^4(V - V_k) + G_L(V - V_L) \tag{1}$$

式中, $I$为外部电流, $C$为电路的电容; $V_{Na}$、$V_k$ 和 $V_L$ 称为逆电位; $G_{Na}$、$G_k$ 和 $G_L$ 分别是模拟钠、钾和泄漏通道电导的参数. 门控参数$n$控制钾通道, $m$和$h$控制钠通道. 这些参数可由方程(2)、(3)、(4)得到, 其中$\alpha$和$\beta$代表对应的粒子向膜内(外)移动的速率.

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \tag{2}$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n \tag{3}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h \tag{4}$$

尽管 H-H 模型具有非常优秀的生物可解释性, 但由于其模型过于复杂, 涉及的计算量过多[41], 因此并不适合用来搭建大规模的 SNN, 同时也有较高的硬件部署门槛.

### 1.1.2 Integrated & Fire 模型

虽然 H-H 模型在生物学上非常还原, 但目前的学习算法往往在更简单的脉冲神经元模型上表现更好. Integrated & Fire (IF)脉冲神经元是一种针对阈下电位的变化规律进行描述的简单模型, 它将输入脉冲信号整合到膜电位中, 如果达到定义的阈值, 则产生输出脉冲, 膜电位再恢复到静息状态. IF 模型可以用下面的公式来描述:

$$C_m\frac{dV}{dt} = I(t) \tag{5}$$

式中$C_m$为膜电容, $V$为膜电位, $I(t)$为当前电流. IF 模型忽略了神经元的时间依赖性, 计算功耗非常低, 更能适应计算资源有限的边缘硬件[42]. 而 Leaky Intergrate & Fired (LIF)模型则以漏电流的形式引入了脉冲神经元的时间依赖性[43], 也更接近实际的神经元行为, 可用下面的式子来表示:

$$\tau_m\frac{dV}{dt} = V_{rest} - V + r_mI(t) \tag{6}$$

其中$\tau_m$为膜时间常数, $r_m$为膜电阻, $V_{rest}$代表静息电位. LIF 模型在简化了动作电位过程的同时, 保留了膜电位的泄露、积累以及阈值激发这 3 个关键特征, 拥有不俗的生物神经元准确性以及模拟速度, 在模拟硬件实现中非常流行[44, 45].

1.1.3 Izhikevich 模型

尽管 LIF 模型有效降低了计算功耗, 但其还是因为过于简洁而忽略了神经元的一些性质. Izhikevich 模型是对 H-H 模型简化后的二维模型[46], 可以再现各种各样的脉冲行为, 在生物合理性和计算效率之间达到了很好的平衡. Izhikevich 模型可以用下面的数学公式描述:

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - u + I(t) \tag{6}$$

$$\frac{du}{dt} = a(bV - u) \tag{7}$$

其中, $V$代表膜电位, $u$代表发射脉冲后膜电位的恢复变量, $I$代表输入电流, $a$代表发出脉冲后膜电位的恢复速度, $b$代表恢复变量$u$受膜电位影响的大小. 通过相关参数的设定, Izhikevich 模型可以凭借极低的功耗模拟生物大脑皮层的放电模式, 进而支持大规模 SNN 的硬件部署. 目前, 已经有研究成功以 Izhikevich 脉冲神经元为基础, 实现了在神经形态硬件上的在线学习[47].

1.1.4 脉冲响应模型

脉冲响应模型(Spike Response Model, SRM)是一种生物激发的脉冲神经元, 它更精确地描述了输入脉冲对膜电位的影响. 与 LIF 模型类似, SRM 模型在其内部膜电位达到阈值时产生脉冲[48], 不过, SRM 模型包含了对于不应期的模拟, 且使用的是滤波器而非微分方程来描述神经元行为. SRM 模型的数学表达式如下:

$$V(t) = \eta(t - \hat{t}) + \int_{-\infty}^{+\infty} K(t - \hat{t}, s) I(t - s) ds \tag{8}$$

其中 $V(t)$为神经元的内部电位, $t$为最后一个神经元输出脉冲的发射时间, $\eta$代表动作电位的状态, $K$为输入脉冲的线性响应参数, $I(t)$表示刺激电流或外部电流. 相比于 LIF 模型, SRM 模型可以通过参数$K$的设定, 获得更广泛的通用性, 同时也保证了较低的计算成本. 已经有工作证明, 利用 SRM 丰富的时空特征, 可以在神经形态硬件上使用代理梯度法高能效地完成 SNN 的训练[49].

1.1.5 小结

表 1 给出了上述四种脉冲神经元模型的分析. 总的来说,为了保证部署更大规模 SNN 的能力, 寻找一种既具有优秀学习能力又具有较高生物可信度的脉冲神经元模型仍然是一个迫切需要解决的问题.

**表 1** 脉冲神经元模型的比较[41, 44]

| 模型 | 生物合理性 | 计算成本 |
|---|---|---|
| H-H | 高 | 1200 FLOPS |
| LIF | 低 | 5 FLOPS |
| Izhikevich | 中 | 13 FLOPS |
| SRM | 中 | 50 FLOPS |

### 1.2 脉冲信息的编码

SNN 与传统 ANN 最显著的区别除了神经元的不同, 还在于信息的编码和解码方式. 在 SNN 中, 数据都是以脉冲序列的形式进行传输, 因此需要将图像像素或实数这样的模拟量用二进制信号进行编码.编码机制决定了用脉冲表示模拟值时的量化或转换误差, 目前使用最广泛的编码方法是速率编码[50]和时间编码[51].

1.2.1 速率编码

速率编码使用相应记录时间内脉冲序列的发射速率来编码信息, 实际输入数字被转换成频率与输入值成正比的脉冲序列, 被视为对神经元输出的一种量化衡量. 速率编码可以进一步分为三种类型: 计数速率编码、密度速率编码和种群速率编码, 图 3 展示了速率编码的可视化案例.

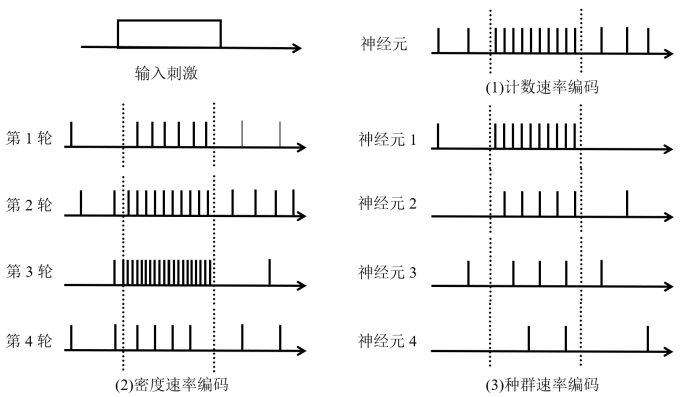


图 3 速率编码

计数速率编码是最常见的速率编码方案, 它由平均发射速率定义, 如式(9)和图 3 中(1)所示, 统计时间窗$T$, 并记录这一窗口内出现的脉冲次数$N$, 计算平均值即可得到速率$v$, 这种简单的方法也被称为频率编码. 若在时间窗$T$内等间隔分布脉冲, 那么出现在间隔内的脉冲可以很容易判断为噪声, 这种方案不仅简单而且拥有良好的鲁棒性.

$$v = \frac{N_{spike}}{T} \tag{9}$$

在密度速率编码方案中, 需要统计多轮次数据, 然后对结果取平均, 定义如式(10), 脉冲密度由一个时间窗$\Delta t$内的脉冲数量$N_{spike}$的所有迭代轮次的平均值除以迭代轮数$K$和窗口持续时间$\Delta t$得到. 图 3 中(2)统计了$K = 4$时脉冲序列的密度速率.

$$p(t) = \frac{1}{\Delta t}\frac{N_{spike}(t;t+\Delta t)}{K} \tag{10}$$

种群速率编码将多个神经元聚集在一起计算它们的脉冲数量, 能够在非常短的时间窗口内测量一群神经元的发射率, 增加了速率编码机制的可信性. 如式(11)所示, 在一个时间窗$\Delta t$内, 将种群中总脉冲数量$N_{spike}$除以神经元总数$N$和持续时间$\Delta t$即可得到发射率. 图 3 中(3)展示了$N = 4$时的种群速率编码方案.

$$A(t) = \frac{1}{\Delta t}\frac{N_{spike}(t;t+\Delta t)}{N} \tag{11}$$

1.2.2 时间编码

时间编码通过单个脉冲的相对时间对信息进行编码, 输入值被转换成具有精确时间的脉冲序列, 常用于时间序列处理. 时间编码方案包括首脉冲触发时间编码, 秩序编码, 延迟编码, 相位编码等.

首脉冲触发时间(Time-To-First-Spike, TTFS)编码是最基础的时间编码方案, 它通过刺激开始时刻和神经元第一个脉冲到达时刻之间的时间差来编码信息. TTFS 方案中每个神经元仅使用第一个脉冲, 而忽略其他脉冲, 牺牲了一定计算精度从而获得更低的功耗. 例如, 图 4 中(1) 里的每个神经元的刺激强度越大, 那么其第一个脉冲到达的时间$\Delta t$就越短.

秩序编码(Rank-Order Coding, ROC)使用神经元发射脉冲的相对顺序作为编码, 在这种情况下, 神经元发射的确切时延并不重要, 重要的是每个神经元的排列顺序, 编码较大模拟值的输入神经元比编码较小模拟值的输入神经元更早被激活. 例如, 图 4 中(2)里 4 个神经元发射脉冲的相对排序为④①②③, 排序越靠前的神经元会编码更大的模拟值.

延迟编码如图 4 中(3)所示, 信息被嵌入到神经元组峰值之间的相对时间差$\Delta t_i$中,较大的模拟值意味着延迟较小的脉冲, 其拥有更早的发射时间和更大的刺激强度.

相位编码受大脑区域内发生的背景振荡启发, 在脉冲和背景振荡之间的相对时间差中编码信息, 如果周期之间没有变化, 相位模式会周期性重复. 如图 4 中(4)所示, 每个神经元根据参考信号$\Delta t_i$发射脉冲, 并对数据进行类似于 TTFS 的编码.

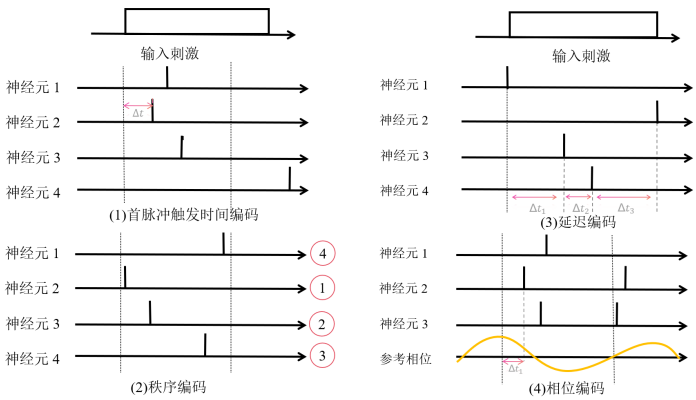


图 4 时间编码

1.2.3 边缘部署时的脉冲编码方案对比

在使用速率编码的 SNN 中, 神经元的整合和发射行为几乎与 ReLU 激活函数相匹配[52], 后者常作为 ANN 中使用的激活函数. 因此, 由 ANN 训练的权重可直接用于 SNN, 无需额外的训练信息, 直接将训练好的 ANN 模型转换为 SNN 模型进行评估[53]. 而时间编码往往需要基于脉冲的训练机制来训练脉冲之间的时间间隔以及网络参数, 限制了 SNN 的探索范围, 适用于浅层网络.

尽管基于速率编码的 ANN-to-SNN 转换法可达到与 ANN 接近的精度, 在复杂任务中得到了广泛应用, 但是通过发射速率传输模拟值需要大量的脉冲, 从而导致推理延迟和能耗的增加, 而且速率编码只关注时间窗内的脉冲数, 忽略了码间干扰, 不能充分利用脉冲序列中包含的时空信息, 因此效率不高.

相比之下, 基于时间编码的 SNN 可以在边缘部署时获得更高的能效, 因为时间编码大大减少了脉冲的数量. 无论输入数据的强度如何, TTFS 编码都只使用一个脉冲, 因此就脉冲数量而言, TTFS 方案有最低的功耗[54]. 然而, 许多研究都使用了复杂的突触函数, 导致在边缘设备上很难用时间编码实现[55-57].

表 2 简单给出了速率编码和时间编码在边缘部署时的性能比较. 总的来说, 速率编码以神经元的发射速率对信息进行编码, 不依赖于每个脉冲事件的精确时间, 也不需要额外的训练信息, 方法相对简单, 有更好的鲁棒性和抗扰性. 时间编码通过神经元发出脉冲的精确定时进行编码, 可以提供更大的信息容量、更快的反应时间和更高的传输速度, 有更低的功耗和更快的计算速度, 但是结构更为复杂, 且拥有更高的训练门槛.

**表 2** 速率编码和时间编码在硬件实现中的性能比较

| 编码方案 | 复杂度 | 鲁棒性 | 信息量 | 功耗 | 速度 | 反应时间 | 训练难度 |
|---|---|---|---|---|---|---|---|
| 速率编码 | 低 | 高 | 少 | 高 | 慢 | 长 | 低 |
| 时间编码 | 高 | 低 | 多 | 低 | 快 | 短 | 高 |

值得一提的是, 速率编码和时间编码拥有各自的独特优势, 组合这些方案可能对系统性能产生巨大影响, 这种组合方案被称为混合编码. 混合方式可以是神经编码方案在网络层之间变化[58], 也可以是神经元在编码方案之间切换[59], 文献[60]提出了一种多层编码方案, 脉冲序列根据时间尺度在不同编码方案的不同通道中表示信息. 目前为止, 脉冲编码的混合方案还没有明确的主题定义, 也没有成体系的硬件实现, 未来还需要进一步的研究.

另一方面, 由于没有一种适用于所有神经形态应用的通用编码方法, 因此在实际环境下所使用的编码方案也视情况而定. 为了解决多种编码方法的兼容问题, 文献[61]提出了第一个基于硬件的脉冲编码器, 支持多种脉冲编码方案和运行时配置, 并允许集成到各类神经形态硬件中, 为 SNN 在边缘部署时的脉冲编码方案选择提供了新的思路.

### 1.3 脉冲神经网络的训练

在深度学习领域, 基于梯度的误差反向传播训练算法[16]是当前 DNN 优化理论的核心, 在实际场景中得到

了广泛应用. 然而, 在 SNN 中, 学习是一项艰巨的任务, 由于脉冲事件的不可微性, SNN 无法直接应用传统的 BP 训练算法进行学习, 也没有公认的核心训练算法. 目前, 训练 SNN 主要有三种策略:无监督学习、监督学习和 ANN-to-SNN 转换法.

1.3.1 无监督学习

无监督学习是指从未标记数据中识别模式的训练算法, 此时输入数据没有相应的标签, 需要训练算法自行学习无标签数据中隐藏的规律或分布关系. SNN 的无监督学习基于 Hebb 规则[62], 该规则会使网络的突触连接适应神经元接收到的数据. 脉冲时间相关的可塑性(Spike-Timing Dependent Plasticity, STDP)算法[63]是 Hebb 规则的一种实现, 基于 STDP 的学习规则根据各自脉冲时间的相关程度, 修改连接一对突触前和突触后神经元的突触权重[64]. 其中, 对神经元发射脉冲有贡献的突触应该得到加强, 对那些没有贡献或以消极方式贡献的突触应该被削弱[65]. 最常见的 STDP 规则由式(12)描述:

$$\Delta\omega = \begin{cases} +A_+ \exp\left(\frac{-\Delta t}{\tau}\right) & \Delta t > 0 \\ -A_- \exp\left(\frac{+\Delta t}{\tau}\right) & \Delta t \le 0 \end{cases} \tag{12}$$
$$\Delta t = t_{post} - t_{pre}$$

其中 $\omega$为突触权重, $\tau$为时间常数, $A_+$和$A_-$为表示增强和抑制强度的参数.

近年来, 由于 STDP 的生物启发特性不仅适合离线学习和推理, 还能以无监督方式学习输入模式的结构, 所以有非常多的工作将 STDP 规则用于片上学习[66-71]. 文献[66]提出了一种具有生物合理性的基于在线 STDP 学习规则的 SNN 模型, 其神经元和突触活动被数字化, 更利于硬件实现, 所提出的模型在 55nm CMOS 工艺下, 硬件资源和功耗分别减少了 40.7%和 36.3%.

然而值得注意的是, SNN 的权重需要大量的片上存储空间进行保存, 同时, STDP 规则的实现过程中涉及大量耗时耗能的内存访问, 这些限制都是亟需解决的问题, 需要提高硬件面积利用率和能效. 为了解决 STDP 硬件实现的存储问题, 目前已经提出了一些方法, 如查找表、分段线性和以 2 为底的指数近似[72]. 文献[67]提出了一种专用于 SNN 的 CMOS 突触和神经元, 以生物启发的方式执行 STDP 学习, 性能评估结果表明, 与传统 CMOS 突触相比, 所提出的突触在能耗和面积上分别减少了 94%和 43%. 此外, 也有工作将 STDP 与存内计算的理念相结合, 实现了硬件友好的 SNN 和高能效 STDP 学习方法, 最终达到了 0.47 nJ/像素的学习效率和 70.38 TOPS/W 的学习能效[68].

1.3.2 有监督学习

有监督学习是指从带有标记的数据中识别模式的训练算法, 此时网络接收输入数据, 并根据学习到的分布对输入进行预测, 然后将预测标签与真实标签进行比较, 以确定误差, 最终基于损失函数相对于参数的梯度更新网络参数. 然而, 由于脉冲神经元不可微的特性, 在 SNN 中直接实现基于梯度的反向误差传播训练方法十分具有挑战性.

为了规避脉冲神经元动力方程不可微的问题, 一些研究将不连续导数近似为连续函数的代理梯度并用于训练端到端反向传播的 SNN[49,73-75]. 代理梯度法的原理如图 5 所示, 其思想是用连续伪导数逼近脉冲函数的不连续梯度. $x_1$和$x_2$表示两个输入神经元的泊松脉冲序列, 他们以$w_{1,1}$和$w_{1,2}$的权重连接到 LIF 神经元, $u$代表膜电位, $\lambda$为泄漏系数, $o$为 LIF 神经元的脉冲输出. LIF 神经元的输出$o$对其输入$x$的导数是不连续的, 因此可由伪导数近似, 以便通过反向传播实现信用分配. 文献[49]使用代理梯度法来替代脉冲序列的导数, 并将权重二值化, 高效地实现了支持神经形态硬件的 SNN 训练算法, 训练得到的 SNN 模型在 N-MNIST, DVS-CIFAR10, DvsGesture 和 N-TIDIGITS18 数据集上分别获得了 99.52%, 62.1%, 97.57%和 90.35%的识别准确率.

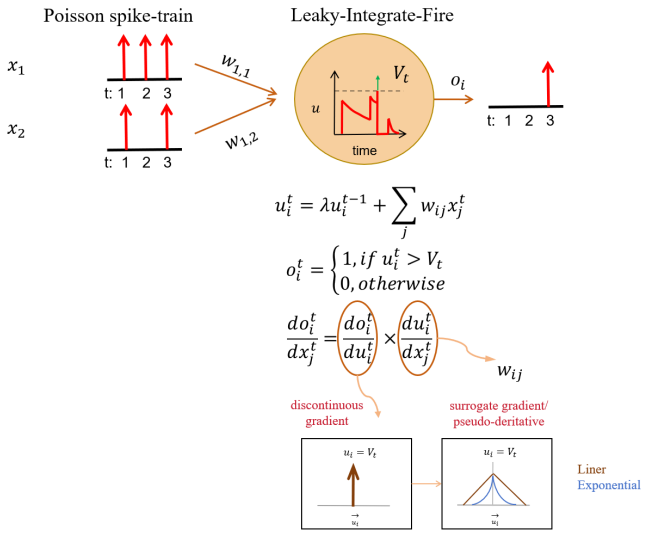


图 5　代理梯度训练方法[76]

除了代理梯度法外, SNN 的有监督训练还包括反向误差传播训练法. 最早使用反向传播误差训练 SNN 的算法之一是 SpikeProp[55], 该模型使用三层结构成功地应用于分类问题. 文献[77]借助 SNN 中的脉冲时序来估计梯度分量, 从而实现与传统 ANN 中采用的随机梯度下降过程类似的学习, 适合在神经形态硬件中实现.

在 SNN 的反向误差传播训练法中, 基于脉冲的梯度下降法是一种广泛被使用的方法. 该方法通过时间反向传播(Back-Propagation Through Time, BPTT)执行信用分配, 结合脉冲序列中的时间信息实现更低的推理延迟[78]. 由于 SNN 在前向传递中需要在多个时间步长上进行计算, 因此可以通过及时展开网络并执行 BPTT 来计算梯度. 如图 6 所示, BPTT 根据输入序列的长度展开 SNN, 并利用链式规则将计算出的最终误差反向传播到整个输入序列. 文献[79]提出了一种使用 BPTT 进行训练的 SNN, 消除了梯度近似的需要, 最终在 MNIST 测试集上达到了 97.58%的准确率.

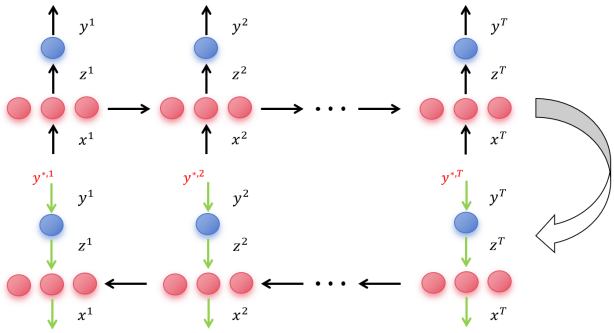


图 6　时间反向传播训练方法

尽管 BPTT 为训练 SNN 带来了新的方案[79,80], 但其复杂的时空动态和巨大的内存消耗导致这类 SNN 在边缘部署时的门槛很高[81]. 同时, 这种训练方法在生物合理性方面也带来了一些问题, 且损害了 SNN 的在线处理能力. 最近, 几种受生物启发的在线训练算法[82,83]一定程度上解决了 BPTT 的缺陷. 这些算法将 BPTT 计算出的梯度因式分解为瞬时学习信号$L^T$与神经元合格性轨迹$e^1$的时间乘积之和. 这两个信号在每个时间步长$t$上结合在一起, 可以得到训练参数的更新, 无需通过时间向后传播信息, 如图 7 所示.

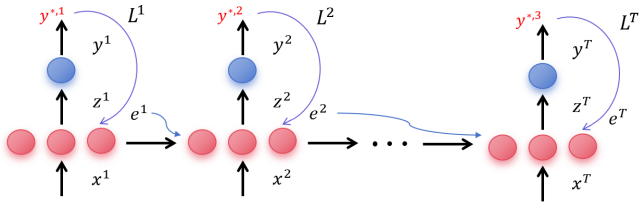


图 7 生物启发式 BPTT 训练方法

需要指出的是，与 ANN 相比，只有少数算法可以直接训练大规模的深度 SNN. 同时，DNN 训练中存在的梯度消失、资源开销大、甚至不收敛等问题一样会出现在 SNN 的训练中. 最后，对 SNN 直接训练时需要额外参考的时间维度带来的巨大内存开销也是在边缘部署时不可忽略的问题.

1.3.3 将 ANN 转化为 SNN

尽管有许多研究完成了 SNN 的无监督学习或者有监督学习，但一些事实表明，采用无监督学习得到的 SNN 在复杂数据集上存在次优精度问题[84.85]，而监督学习也因为脉冲序列的不连续性和不可微性，很难扩展到更深层的 SNN，当网络规模更大时，也存在误差反向传播训练时的梯度消失或爆炸问题.

ANN-to-SNN 转换法从拥有成熟训练方法的 ANN 中学习，快速得到性能良好的 SNN[86-89]. 在 ANN-to-SNN 转换法中，因为 ReLU 神经元在功能上等同于没有任何泄漏和折射期的 IF 脉冲神经元[90]，所以通常会选择 ReLU 神经元作为待转换 ANN 的基本神经元. 在转换过程中，首先训练 ReLU 神经元组成的 ANN，训练完成后，用 ANN 的权重初始化具有 IF 神经元和等结构的 SNN，图 8 展示了转换过程中 ReLU 神经元映射为 IF 脉冲神经元的原理. 为了确保在 ANN-to-SNN 转换法的过程中分类准确性损失尽可能小，就必须适当选择神经元阈值与突触权重的比率. 因此，该方法的大部分研究工作都集中在选择阈值平衡的合适算法，或者对网络的不同层进行权重归一化，以实现近乎无损失的 ANN-to-SNN 转换[91]. 文献[92]采用 ANN-to-SNN 转换法，在 FPGA 平台上实现了 SNN 模型. 该模型在几乎没有精度损失的情况下，具有每瓦 8841.7 帧的高能效.

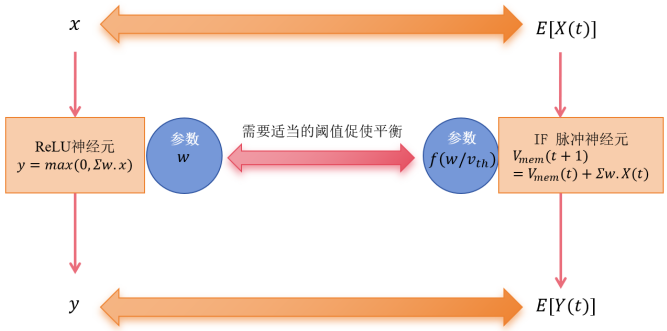


图 8 将 ANN 神经元转化为 SNN 神经元

采用 ANN-to-SNN 转换法有几个好处. 第一，在大规模网络中模拟精确的脉冲动作可能会在计算上很昂贵，特别是在资源严格受限的边缘硬件设备中. 因此，ANN-to-SNN 转换法可以让 SNN 避免高代价的大规模训练，同时与转换前的 ANN 相比，得到的 SNN 精度损失很小[93]. 第二，神经网络的训练过程可以在 ANN 上进行，从而使用算力资源更充足的 GPU 等设备，无需在线学习，降低边缘部署的硬件要求. 最后，对 ANN 进行训练的方法已经经过了长久的发展，技术方面更成熟，这能为转换得到的 SNN 提供更高的性能上限.

当然，这种转换训练法也存在一些缺陷，比如可能会忽略 SNN 特有的一些生物合理性，以及在转换过程中或多或少地损失一些精度[38]，此外，现有的 ANN-to-SNN 转换法也存在模拟周期长的问题[94].

最新工作表明，生物学上合理的 SNN 的性能在很大程度上取决于模型参数和神经动态. 文献[95]探讨了生

物时间常数对 SNN 中信息传输的影响, 并通过适当选择时间常数, 提高了 SNN 训练的收敛速度. 与之前的工作相比, 该工作采用欧拉数值方法的并行实现, 达到了在训练(推理)速度上超过 300 倍(240 倍)的提升, 并且在能耗上减少了 180 倍(250 倍). 可以预见的是, 生物合理性与性能的有机结合将是未来 SNN 训练算法的长久目标, 此外, 也可以进一步探索与权重量化、结构剪枝等压缩算法的结合, 进一步发挥 SNN 的能效优势[96].

### 1.4 脉冲神经网络的拓扑

SNN 的拓扑结构直接反映了神经元和突触之间的连接方式. 根据训练时网络拓扑是否发生变化, 可以将 SNN 的拓扑分为静态拓扑和动态拓扑. 静态拓扑中脉冲神经元数量和突触连接方式在训练时一直保持不变, 改变的只有突触之间的连接权重. 常见的静态结构包括前馈网络结构和循环网络结构[97]. 动态拓扑中脉冲神经元数量和突触连接方式在训练时会动态变化, 典型代表为进化脉冲神经网络(Evolutionary Spiking Neural Network, ESNN). ESNN 的设计思想来源于生物学的互联进化系统, 它能够以自适应、自组织、在线连续的方式动态改变系统的结构和功能.

#### 1.4.1 静态拓扑 SNN

在 SNN 中, 最常见的静态拓扑是前馈结构和循环结构. 前馈结构一般由卷积层和全连接层的组合构成,没有任何自连接或循环. 循环结构中, 神经元不但可以接受其他神经元的信息, 也可以接受自身的信息, 形成具有环路的拓扑结构. 采用这两种静态拓扑的 SNN 分别被称作卷积脉冲神经网络(Convolutional Spiking Neural Network, CSNN)和循环脉冲神经网络(Recurrent Spiking Neural Network, RSNN).

CSNN 将卷积神经网络(Convolutional Neural Network, CNN)的概念与 SNN 结合起来, 在 CSNN 中, 卷积层和池化层等经典的 CNN 层次结构被重新设计为适用于脉冲信号的形式. 脉冲信号在神经元之间通过突触连接进行传递, 并在神经元接收到足够数量的脉冲时触发激活. CSNN 的结构如图 9 所示, 输入为脉冲序列, 经过多次卷积-池化采样后, 最终将处理的信息交给分类器进行预测.与 CNN 相比, CSNN 可以更好地模拟生物神经系统的工作方式, 在处理时间相关信息和事件驱动任务方面更具潜力, 同时能够提高边缘部署的能效.

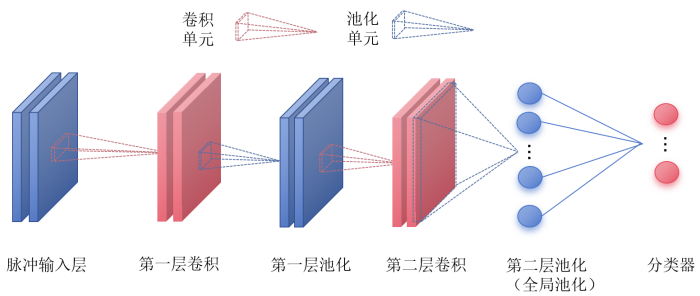


图 9 卷积脉冲神经网络的结构

在循环网络中, 拓扑结构具有反馈连接,神经元的输出以时间延迟作为输入路由返回. 因此, 输出是当前输入和神经元过去状态的函数. SNN 中隐含着这种关系, 因为膜电位取决于输入和前一个时间步的电位, 如图 10 所示, 等效膜电位$V(t)$充当了过去输入的记忆, 最终的神经元输出$o(t)$由输入$x(t)$与上一个膜电位$V(t-1)$共同决定.

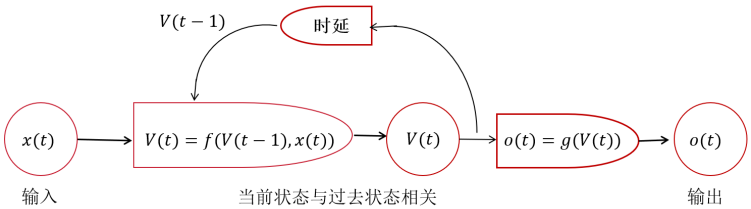

图 10 脉冲神经网络的隐式循环

RSNN 的拓扑如图 11 所示. 脉冲神经元之间的连接具有循环结构, 结合了脉冲神经元的时间编码特性和循环神经网络(Recurrent Neural Network, RNN)的时间依赖处理能力.

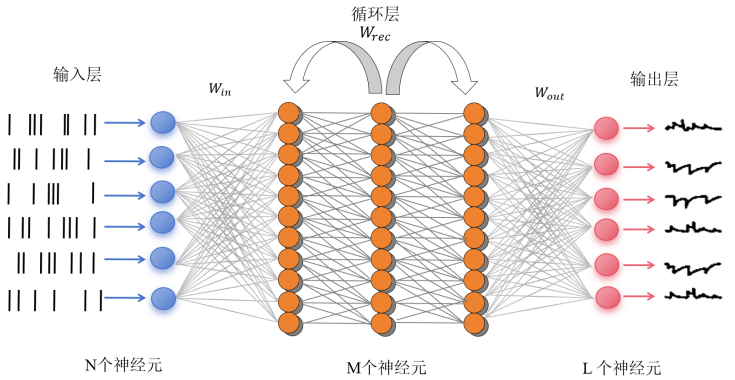


图 11 循环脉冲神经网络的结构

与 RNN 相比, RSNN 吸收了 SNN 高能效的优势, 更容易满足边缘设备的功耗要求. 然而, 传统的 RNN 存在梯度消失和梯度爆炸等问题, 从而恶化整体学习[98], RSNN 也不例外. 尽管在过去的几年中, 已经有一些工作尝试以生物似然的方式训练 RSNN 模型[99-102], 然而这一领域的研究非常有限, 目前, 对 RSNN 的训练仍存在不小的挑战.

总的来说, 静态拓扑 SNN 可以看作是传统 ANN(CNN 和 RNN)与 SNN 的结合. 这类 SNN 既继承了传统结构在图像数据与序列数据处理上拥有的优势, 也吸收了脉冲序列作为传递的信息时在功耗方面的优化, 更利于边缘部署. 然而, 这种结构无法彻底解决传统 ANN 固有的缺陷, 如模型过于复杂时训练难度增加, 可解释性弱, 以及对于硬件故障的容错能力差等.

#### 1.4.2 动态拓扑 SNN

进化算法(Evolutionary Algorithm, EA)是一种基于种群的元启发式算法, 可用于直接优化网络拓扑结构和模型超参数, 或优化突触权重和延迟[103,104]. ESNN 是一种结合了 SNN 和进化算法的神经网络模型. 在 ESNN 中, 输入数据被编码成脉冲序列并传递到网络中,利用脉冲网络的进化结构动态生成新的脉冲神经元, 并将其添加到相应的神经元储备类别中. 目前, 常用的进化算法有差分进化、语法进化、和谐搜索算法和粒子群优化等[105-107].

除了采用进化算法外, 对神经元模型进行改进, 让脉冲神经元获得不断自我优化的能力, 也可以得到 ESNN. 文献[108]在传统 LIF 脉冲神经元模型的基础上提出了进化脉冲神经元和进一步改进的模型——自适应进化脉冲神经元, 所提出的神经元模型能根据突触前的输入和随后的神经元发射事件, 自适应地调整突触前的输入电流. 实验证明, 所提出的进化模型还能降低网络发射活动的速率, 使网络对输入的响应速度更快, 但触发的脉冲信号和计算操作更少, 有助于在神经形态硬件上实现 SNN 模型的低功耗计算.

目前, ESNN 在神经形态硬件上的实现很受欢迎, 因为这类动态拓扑的 SNN 不依赖于模型的特定特征,可以灵活地发挥所选择硬件平台的特有优势, 适用于各种设备和应用程序. 然而, 目前对于较为复杂的模型或应用程序, ESNN 也存在收敛速度较慢的问题, 未来还需要不断优化 ESNN 的训练算法.

## 2 神经形态计算工具链

本章将对 SNN 边缘部署的工具链进行介绍, 包括软件编程框架和仿真工具. 不论何时, 开发工具都在一个领域的研究中扮演着重要的角色. 作为算法和物理设备之间的桥梁, 高效且完善的工具链能简化开发过程,

加速应用程序的部署, 促进合作和知识共享, 为一个领域的研究提供更便捷的条件.

### 2.1 软件编程框架

编程框架是一种提供了特定功能和结构的软件工具集合, 使开发者不必过多关注底层的技术细节, 只需专注于解决问题和实现功能. 在深度学习领域中, TensorFlow[109]、PyTorch[110]和 Caffe[111]这些耳熟能详的框架使得构建和训练神经网络变得简单而快速, 具有广泛的应用场景. 然而, 这些传统框架的原生版本并没有提供专门的神经形态计算支持, 通常需要添加一些第三方库和扩展才能支持 SNN 模型的开发和训练. 随着神经形态计算及 SNN 的快速发展, 为了让研究者们的想法更好地被实现, 设计相应的编程框架十分有必要, 下面简单介绍几种现有的 SNN 专用编程框架.

BindsNET[112]是一个用于快速构建和模拟 SNN 的开源 Python 软件包. 它支持 STDP 训练算法, 允许研究人员在 CPU 或 GPU 上测试软件原型, 然后将模型部署到专用硬件上. BindsNET 的缺点在于它无法实现复杂的网络, 并且缺乏对延迟突触的支持.

SpikingJelly[113]是一个基于 PyTorch 的开源库, 专门用于构建和训练 SNN. SpikingJelly 支持各种常见的 SNN 训练方法, 比如代理梯度法, STDP, ANN-to-SNN 转换法等等. 此外, 通过实现 Tempotron 神经元[114], SpikingJelly 还可以支持事件驱动的 SNN.

SpykeTorch[115]是基于 PyTorch 的 SNN 开源高速编程框架, 采用时间编码方案, 并支持 STDP 和带有奖励调节的 STDP 学习规则[116]. SpykeTorch 具有很强的通用性, 所有的计算都基于张量, 完全由 PyTorch 函数实现, 在 CPU、GPU 或多 GPU 平台上具备即时优化能力.

目前, 这些 SNN 专用的编程框架都处于积极开发迭代的过程中, 还存在可扩展性方面的问题, 且不够稳定. 在未来, 扩大这些编程框架的软硬件兼容性并形成稳定版本至关重要. 表 3 简单总结了目前 SNN 专用编程框架.

**表 3** SNN 专用编程框架小结

| 框架名称 | 神经元模型 | 编码方案 | 事件驱动 | 训练方法 |
|---|---|---|---|---|
| BindsNET | IF, Izhikevich | 秩序编码 | 不支持 | STDP |
| SpikingJelly | LIF | 频率编码, 延迟编码, 相位编码 | 支持 | ANN-to-SNN, STDP |
| SpykeTorch | LIF | 首脉冲触发时间编码, 秩序编码 | 不支持 | STDP |

### 2.2 仿真工具

在将 SNN 模型部署到硬件设备前, 通常需要进行软件模拟仿真. 通过软件仿真, 可以验证硬件设计的功能是否按预期工作, 在实际部署前发现和解决问题. 此外, 许多软件模拟器还提供性能和时序分析, 并评估硬件实现的功耗和资源利用情况, 有助于达到更节能和高效的硬件实现. 选择 SNN 仿真模拟器有一些标准[117], 它理应是开放的, 且易于调试和运行, 并能支持各种硬件. 下面介绍一些常见的 SNN 模拟器.

NEST[118]支持具有不同生物学原理的神经元和突触模型, 提供了 50 多种神经元模型和 10 多种突触模型, 包括 Izhikevich 和 H-H 等典型的脉冲神经元模型, 以及 STDP 学习规则及其各类变体, 同时允许在仿真过程中随时检查和修改神经元和突触状态. NEST 由 C++实现, 并提供一个名为 PyNEST 的 Python 接口, 可以有效地从单核电脑扩展到多节点超级计算机.

Nengo[119]是一款专注于神经认知仿真的模拟器, 支持包括 SNN 在内的各类神经网络模型. Nengo 由 Python 实现, 提供了一种灵活的方式来构建和模拟 SNN 模型, 支持 TensorFlow 框架扩展, 允许用户定义神经元类型、学习规则和优化方法, 将大脑的认知功能映射到神经网络中.

NeMo[120]是一款充分发挥 GPU 的通用计算能力以仿真大规模 SNN 的模拟器. NeMo 实现了 Izhikevich 脉冲神经元模型, 能够在不牺牲计算效率的情况下真实地模拟各种脉冲动态, 也能在并行流处理器上模拟具有传导延迟的 SNN, 同时减少内存带宽需求.

GeNN[121]是用于 SNN 仿真的软件工具包. GeNN 通过代码生成将神经元仿真的不同描述转换为统一的

C++代码, 得到与英伟达 GPU 兼容的可执行文件, 从而在 CPU 或 GPU 上加速 SNN 仿真.

Brian2[122]是使用 Python 编写的一款 SNN 开源模拟器. 它高度灵活、易于扩展, 方便用户使用微分方程定义脉冲神经元和突触模型, 允许高效地模拟 SNN. Brian2 使用自己的通用描述方法, 无需修改核心代码即可轻松生成不同平台的代码, 具备强大的可扩展性.

CARLsim6[123]是 CARL-sim SNN 仿真平台的第 6 个版本, 该平台是首批利用 CUDA GPU 模拟大脑巨大并行处理能力的开源仿真系统之一. 最新的 CARLsim6 支持当下的各类主流计算机和 GPU, 增加了可扩展性, 并纳入了长期和短期突触可塑性, 能进行快速学习、网络重新连接和神经活动调节.

表 4 统计了主流 SNN 仿真软件的功能.

**表 4** SNN 仿真软件功能统计

| 仿真工具 | 开源 | GPU 支持 | 编程语言 | 特点 |
|---|---|---|---|---|
| NEST | 是 | 否 | C++<br>Python | • 适合动态关注神经系统的结构<br>• 仿真速度快<br>• 可利用本地网络中的多核计算机提高内存效率 |
| Nengo | 是 | 是 | Python | • 支持 CPU 多线程执行<br>• 拥有仿真后端, 如 Nengo FPGA、Nengo Loihi、Nengo SpiNNaker 和 Nengo OpenCL<br>• 专注于 SNN 的高级神经行为 |
| NeMo | 是 | 是 | C++ | • 支持 CPU 多线程执行<br>• 可模拟不同的神经元模型和硬件配置<br>• 可在支持 CUDA 的 GPU 上运行 |
| Brian2 | 是 | 是 | Python | • 支持所有主流操作系统<br>• 可在英伟达 GPU 上运行 Brian 2 脚本<br>• 可将性能提升数十倍至数百倍 |
| CARLsim6 | 是 | 是 | C++<br>Python | • 提供易于使用的编程界面<br>• 为 SNN 提供方便的参数调节<br>• 提供生物可信的 C++实现, 在灵活性和性能之间取得平衡 |

## 3 神经形态硬件平台

随着物联网(Internet of Things, IoT)时代任务的日益复杂, 以及需要部署的网络规模的不断增大, 在功率、能量和计算资源受限的边缘设备中训练和部署这些 DNN 已成为一项艰巨的任务[124]. 相较于高能耗的 ANN, SNN 凭借其理论上高效的事件驱动计算特性, 已成为高能效的部署方案. 然而, 由于传统的深度学习平台无法彻底发挥出 SNN 高能效的潜力,神经形态硬件平台的出现为 SNN 的边缘部署打开了一扇窗户. 神经形态硬件设计的最初灵感来自于建立与人脑等效的电子系统, 以模仿人脑的计算能力[125]. 随着摩尔定律即将结束, 与登纳德缩放[126]相关的功率需求不断增加, 以及 CPU 和内存之间的低带宽(冯·诺伊曼瓶颈), 神经形态硬件平台受到了越来越多的关注, 并被视为执行神经形态计算的可行解决方案. 由于其固有的并行性, 低能量开销和更小的物理空间占用, 神经形态硬件可以在资源受限的嵌入式系统以及 IoT 中的边缘设备上高效实现机器学习任务[127].

不同于传统的冯·诺依曼架构, 神经形态硬件通常采用去中心化的可扩展架构, 通过路由网络交换中间结果数据, 允许多计算核心同时工作. 同时, 在神经形态硬件平台中, 外部存储器将不复存在, 取而代之的是存储单元与计算单元的高度贴合, 从而带来极高的并行性和访存效率. 神经形态硬件旨在最大限度地降低能耗和成本, 同时尽可能保持高精度.由于这些特性与 SNN 的理念十分契合, 神经形态硬件可以看作是 SNN 及其应用部署的理想硬件平台.

下面将详细介绍当下流行的适合部署 SNN 的神经形态硬件平台以及一些硬件映射方法.

### 3.1 现场可编程门阵列

FPGA 是一种可编程逻辑器件, 它可以根据用户的需求进行可重构的硬件设计. 相比于传统的冯·诺依曼

架构处理器(如 CPU 和 GPU), FPGA 通常具有更低的功耗[128], 这对于在嵌入式设备或低功耗环境中部署神经网络模型非常有利. 与专用集成电路(Application Specific Integrated Circuit, ASIC)相比, 由于 ASIC 的设计、验证和制造的高非重复性工程成本和时间难以跟上神经网络模型改进的步伐[129], 而作为可配置工具的 FPGA, 由于其灵活性, 可以根据具体的神经网络结构和算法进行定制化设计, 保证了更短的开发时间, 因此更适合当今快速发展的神经形态计算领域.

尽管 FPGA 本身并非专门设计用于神经形态计算, 但它具有高度的灵活性和不俗的并行计算能力, 同时具备高能效和位级操作的特点, 因此在一定程度上被认为是可以模拟生物神经网络自然可塑性的神经形态硬件平台. 研究表明, 在现代 FPGA 平台上部署时, SNN 可以凭借事件驱动的特性和更高的内存组织效率, 达到比 ANN 更低的能源需求这一预期[130]. 结合 SNN 在边缘设备部署时的明显优势, 目前的研究热点之一是开发基于 FPGA 的 SNN 加速器, 以实现高性能神经形态计算, 将人工智能带向边缘计算场景[131].

在信号处理领域, SNN 相较于 ANN 具备更强的时空信息的处理能力, 脉冲神经元事件驱动的特点也使得 SNN 的计算更高效. 文献[132]基于 FINN[133]框架高效利用了 FPGA 的并行资源, 并结合 SNN 在序列数据处理上的优势, 开发了一种流式 SNN 架构 S2N2, 在射频信号识别任务上使内存利用率提高了三个数量级以上.

在语音识别领域, 文献[134]在 10 块 Cyclone 系列 FPGA 上设计了一种基于 SNN 的数字感知系统. 该系统采用全并行流水线方案, 最高频率可达 107.28 MHz, 最大吞吐量为 5364 Mbps, 能效为 845.85 μJ, 高效地完成了听觉感知任务. 在 TIMIT 数据集上的实验表明, 在明显的噪声条件下(信噪比为 20 dB), 该系统的语音识别准确率高达 85.75%, 并且随着信噪比的下降, 准确率的衰减幅度也很小, 是目前最先进的基于 SNN 的听觉感知系统. 文献[135]提出了一种基于 SNN 的与动物运动相关的声音识别系统. 通过将 SNN 部署在 FPGA 平台上, 该神经形态听觉系统产生了类似于生物耳蜗的脉冲输出的表示. 即使在声音中存在相同强度的白噪声时, 其基于信噪比的检测系统也能达到 91%以上的准确率.

在智慧医疗领域, 文献[136]提出了一个神经形态系统, 将神经记录头戴式设备与 SNN 处理核心集成在 XEM7360 FPGA 上, 用于处理颅内脑电图, 并展示了如何可靠地检测高频振荡, 从而实现了最先进的准确性、敏感性和特异性. 该工作也是利用混合信号神经形态计算技术实时识别颅内脑电图中相关特征的首次可行性研究.

在机器人领域, 文献[36]基于 FPGA 实现了具有可重构连接性的 SNN, 并应用于 Khepera 机器人的避障任务. 文献[137]在 Spartan-6 FPGA 上建立了一个神经形态框架, 使双足、四足和六足的机器人能够自由运动, 可以与多足机器人的 CPG 运动机制相结合.

在图像识别领域, 基于 FPGA 实现的 SNN 得到了大量的应用. 文献[138]在 Xilinx SPARTAN-6 FPGA 上实现了一种同时利用基于事件和基于帧的处理方式的混合神经网络, 兼具 ANN 与 SNN 的特点, 在每帧仅需消耗 7 uJ 的情况下, 最终在 MNIST 数据集上获得了 97%的准确率. 文献[139]在 Xilinx Zynq ZCU102 上高效部署了 SNN 模型. 该工作使用两种并行方法来提高数据重用率, 最终在 MNIST 数据集上达到了 98.94%的准确率, 功耗比 GPU 实现方法低 22 倍, 速度比 CPU 实现方法快 41 倍. 文献[140]采用事件驱动 STDP 规则进行训练, 从输入模式中学习各种特征, 并以无监督方式进行分类, 在 Stratix III FPGA 平台上实现了高能效的 SNN, 最终在 MNIST 数据集上的准确率为 93%.

表 5 统计了部分基于 FPGA 实现的 SNN 模型在 MNIST 数据集的表现.

**表 5** 基于 FPGA 实现的 SNN 在 MNIST 数据集的表现情况

| 作者 | 年份 | FPGA 平台 | SNN 模型/算法 | 准确率(%) | 推理时间/ 工作频率 | 功耗 |
|---|---|---|---|---|---|---|
| Neil[141] | 2014 | Xilinx Spartan-6 | 事件驱动 SNN | 92 | 0.53 s/张 | 1.2 μJ/张 |
| Wang Q[142] | 2017 | Xilinx Virtex-6 | STDP | 89.1 | 8.4 s/张 | 1.76 μJ/张 |
| Mostafa[51] | 2017 | Xilinx Spartan6-LX150 | 前馈 SNN | 96.98 | N/A | N/A |
| Zhang C M[143] | 2019 | Terasic DE2-115 | 前馈 SNN | 96.26 | 100 MHz | 293 mW |

| | | | | | | |
|---|---|---|---|---|---|---|
| Zhang J[144] | 2019 | Xilinx VC707 FPGA | 事件驱动 SNN | 98 | 1.1ms/张 | 360 mW |
| Abderrahmane[145] | 2019 | Altera Cyclone V | 前馈 SNN | 98.15 | 50 MHz | N/A |
| Kuang[138] | 2019 | Altera Stratix III | STDP | 93 | N/A | N/A |
| Guo[146] | 2019 | Xilinx V7 690T | ANN-to-SNN | 98.98 | 100 MHz | 745 mW |
| Losh[147] | 2019 | Xilinx Zynq XC7Z010 | 前馈 SHiNe | 97.70 | 125 MHz | 161 mW |
| Ju[139] | 2020 | Xilinx Zynq ZCU102 | ANN-to-SNN | 98.94 | 6.11 ms | 4.6 W |
| Han[148] | 2020 | Xilinx ZC706 | 前馈事件驱动 SNN | 97.06 | 200 MHz | 477 mW |
| Fang[149] | 2020 | Xilinx ZCU102 | 基于脉冲的梯度下降 | 99.2 | 7.53 ms | 4.5 W |
| Wang S Q[150] | 2020 | Xilinx XCVU440 | CSNN | 99.16 | 200 MHz | 1.5625 TOPS |
| Aung[151] | 2021 | UltraScale+ VCU118 | CSNN | 99.14 | 500 MHz | 5.64 kFPS/W |
| Li[152] | 2021 | Xilinx Virtex-7 | STDP | 92.93 | 3.15 ms/张 | 5.04 mJ/张 |
| Zheng[153] | 2021 | Xilinx ZCU102 | TSTDP | 90.53 | 200 MHz | 782 mW |
| Gerlinghoff[154] | 2021 | Xilinx XCKU3P | CSNN | 99.1 | 294 μs | 3.4 W |
| Zhang J[155] | 2021 | Xilinx Virtex-7 | BP-STDP | 95.3 | 0.27 ms/张 | 0.34 mJ/张 |
| Pancha[53] | 2022 | Xilinx ZCU102 | ANN-to-SNN | 99.3 | 200 MHz | 32.7 kFPS/W |
| Liu Y[156] | 2022 | Xilinx Kintex-7 | 事件驱动 SNN | 97.70 | 4.17 ms/张 | 2.23 mJ/张 |
| Ye[157] | 2022 | Xilinx Kintex-7 | CSNN | 99.10 | 1.21 ms/张 | 1.19 mJ/张 |
| Chen[158] | 2022 | Xilinx XC7Z045 | CSNN | 98.5 | 22.6 GSOP/s | 19.3 GSOP/W |
| Sommer[159] | 2022 | Xilinx XCZU7EV | CSNN | 98.3 | 0.04 ms | 2.1 W |
| Liu H[160] | 2023 | Zynq XA7Z020 | CSNN | 99.00 | 0.27 ms | 0.28 W |
| Wang Z[161] | 2023 | Xilinx KCU115 | 前馈 SNN | 99.4 | 65.7 GSOP/s | 41.7 GSOP/W |
| Li J[162] | 2023 | Xilinx XCZU3EG | CSNN | 98.12 | 5.53 TOP/s | 2.55 W |

截止目前, 在 FPGA 上部署 SNN 仍存在许多的挑战与困难. 其中最大的问题出现在内存方面. 因为当部署大规模 SNN 时, 它需要大量的神经元和突触来缓冲大权重矩阵. 考虑到商用 FPGA 设备中的资源数量, 瓶颈往往出现在块存储器(Block Random Access Memory, BRAM)上, 这导致网络中的神经元数量受到限制. 另一方面, 硬件上的实现面临着准确性和成本或资源使用之间的权衡. 实现高准确率虽然性能优异, 但功耗较大, 如何在边缘环境中获得最合适的性能与功耗配置将一直是该领域需要面对的问题. 当使用 FPGA 实现 SNN 时, 另一个挑战是确定适当的模型和模型组合, 以及部署在逻辑阵列上的互连和架构问题, 以便网络可以进行训练和片上学习. 此外, 与 CPU 和 GPU 相比, FPGA 在算力上具有明显劣势, 因此在 FPGA 上实现 SNN 非常耗时. 最后, 由于 FPGA 的可编程性相对较低[163], 缺乏诸如 TensorFlow, PyTorch 等软件开发框架的支持, 因此需要开发人员对硬件细节有更深入的了解, 开发难度较高.

针对这些问题, 可以总结未来在 FPGA 平台上部署 SNN 的研究主题:

I) 近似计算. 考虑到 FPGA 平台在资源方面的严格限制, 进一步压缩 SNN 模型中权重和参数的内存消耗是非常有必要的. 近似计算方法是未来可以应用在基于 FPGA 的 SNN 加速器的技术, 文献[164]提出的近似乘法器减少了 25% ~53%的 LUT 总使用量. 此外, 量化技术在 SNN 加速器上也具有十足的潜力.

II) 自动模型生成. 网络架构搜索(Neural Architecture Search, NAS)是近期研究的热点. 文献[165]探讨了 NAS 在 FPGA 上的应用, 这也可能是在 FPGA 平台上高效部署 SNN 的一种潜在的研究和应用思路.

III) 在 FPGA 平台上提供 SNN 的开发框架. 为了降低 FPGA 开发的门槛, HLS(High-Level Synthesis)高层次综合工具[166]应运而生, 它提供了一种抽象层, 将高级语言代码转化为硬件级别的描述, 自动生成硬件电路, 使开发者能够使用熟悉的高级语言进行硬件设计, 大大简化了 FPGA 的设计流程, 能有效降低 SNN 模型在 FPGA 平台部署的门槛.

### 3.2 神经形态芯片

由于 SNN 在硬件电路实现时具有超低能耗的优势, 在过去十年中, 各类以 SNN 为原型的神经形态芯片不断涌现. 与 ANN 缺乏生物可解释性不同, 神经形态芯片从大脑的结构和功能中汲取灵感, 其主体是一个类似于人脑神经网络的近似 SNN. 与传统计算机取指-执行的循环工作方式不同, 神经形态芯片遵循并行工作和分布式处理机制, 完成学习、记忆、推理等认知任务[167]. 许多计算核心在芯片中同时工作, 模拟生物神经元的动力学行为, 通过路由网络完成和其他计算核心的数据和与控制交互, 呈现去中心化的操作模式, 具有极高的并行性和内存访问效率, 突破了 CPU 和 GPU 这类通用处理器在执行深度学习任务时面临的功耗与内存瓶颈. 图 12 展示了两种常见的神经形态芯片路由结构.

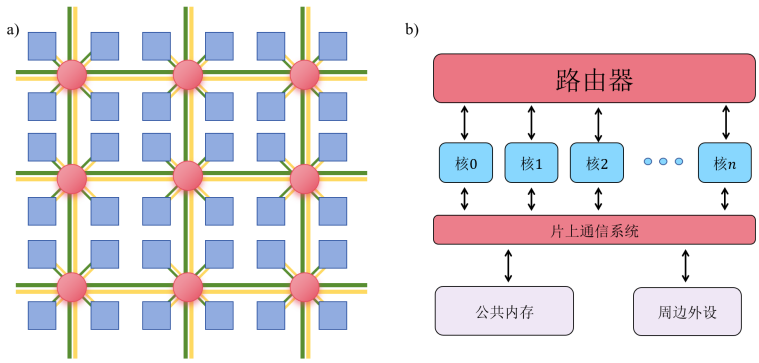


图 12 a)网状路由结构. b)公共路由结构

现有的神经形态芯片根据电路实现技术的不同, 可以分为数模混合神经形态芯片和全数字神经形态芯片.

#### 3.2.1 数模混合神经形态芯片

模拟电路使用物理过程来模拟人工神经元的某些计算功能. 这种方法的优势在于通过系统的自然动态可以非常高效地实现显式数学运算成本可能很高的操作[168]. 此外, 实值物理变量的精度几乎可以达到无限, 为电路功能提供了非常高的上限. 然而在大规模神经形态芯片中, 由于路由网络部分存在跨核和跨芯片的长程数据传输的需求, 难以用模拟电路精确完成, 因此, 在混合实现的神经形态芯片中, 往往用数字电路实现网络连接, 而模拟电路则致力于复现神经元动力学.

Neurogrid[169]是一种数模混合神经形态芯片, 它灵活地利用神经元离子通道的动态特性和晶体管阈下区域的电学特性之间的相似性来设计神经元回路和突触连接. Neurogrid 的硬件系统由 16 个神经网格芯片的 PCB 组成, 每个芯片有 256 × 256 个模拟神经元, 采用 180 纳米 CMOS 技术制造, 能够对具有数十亿个突触连接和 100 万个神经元的大脑进行实时生物模拟.

BrainScaleS[170]采用 8 英寸硅晶片, 片上集成了 352 个计算芯片, 采用超阈值模拟电特性进行神经元动力学仿真. BrainScale 系统实现了自适应指 IF 神经元模型和突触, 并支持 STDP 规则的在线学习, 通过 FPGA 模块进行计算核心之间的路由通信, 可以实现远超生物大脑的运行速度, 常用于加速具有精确生物神经行为的类脑神经网络的时间模拟.

ROLLS[171]只有 256 个神经元和 128000 个突触, 采用亚阈值模拟电路实现神经元和突触动力学, 常用于模拟生物神经系统的物理活动、研究计算神经科学模型和构建类脑计算系统. 在硬件架构上, ROLLS 芯片将内存和计算共置, 包括一个可配置的突触电路阵列, 其中的脉冲神经元可产生符合生物特征的响应特性, 从而表现出各种真实行为.

DYNAPs[172]是采用超阈值模拟电特性实现的数模混合神经形态硬件平台, 包含 9 块芯片, 单芯片含 4 个计算核, 每个计算核拥有 256 个神经元. 该系统采用两级路由方案, 芯片间采用 2D 网格路由拓扑, 片内的计算核心则通过树状路由进行通信, 结合了网格拓扑的低带宽和树状拓扑的低延迟优点, 拥有很高的内存效率.

数模混合神经形态芯片的优势在于它可以将数字和模拟电路集成在同一芯片上, 减少了电路的面积、功耗和成本, 在一些仿生学应用实现了全面的性能优势, 非常适合实时研究系统与环境的交互问题.

3.2.2 全数字神经形态芯片

虽然模拟电路可以相对容易地复现较为复杂的神经元动力学, 但它也面临着抗干扰能力差、可编程性不足以及难以仿真等问题. 在数字电路中, 神经元的所有变量都用比特位表示, 变量的精度取决于表示变量的比特数, 这种可配置的数据精度极大地影响了基本操作的能耗和变量存储的内存要求. 与模拟电路相比, 数字电路设计的最大优势在于变量的精度是可控和有保证的. 此外, 数字电路硬件的设计也可以采用最先进的芯片设计和制造技术, 拥有较为成熟的工艺流程. 目前, 基于数字电路的稳定性和可靠性, 全数字神经形态芯片在工业界得到了更广泛的应用. 纯数字电路实现的芯片可以构建超低功耗的大规模神经形态系统, 并精确再现 SNN 模型.

TrueNorth[173]是 IBM 公司于 2014 年推出的全数字神经形态芯片, 包含 54 亿个晶体管和 4096 个神经突触内核, 面积仅为 430 $mm^2$, 可达到 100 万个神经元和 2.56 亿个突触的计算规模. TrueNorth 支持 LIF 神经元模型及其诸多变体, 采用事件驱动的异步同步电路混合设计方案, 不依赖全局时钟协调工作, 不会因芯片故障而影响整体工作, 具有良好的可扩展性和可靠性. 事件驱动的特性使得该芯片工作功耗极低, 在模拟百万神经元规模的 SNN 时也仅有至多百毫瓦的功耗, 可实际应用于图像识别和语音处理等领域. 尽管 TrueNorth 为神经形态芯片的设计带来了突破性进展, 但其仅支持 SNN 的推理, 不支持片上学习, 需要额外的训练设备进行参数更新, 还存在一定发展空间.

SpiNNaker[174]是一个大型数字神经形态硬件平台, 由 48 个芯片组成. 单个芯片含有 18 个 ARM 处理核心和 128MB 的 DRAM 存储器, 每个 ARM 核可以仿真近 1000 个神经元. PyNN 接口[175]的存在以及 ARM 核心的高度灵活性使 SpiNNaker 电路板允许自由编程, 支持包括 LIF、Izhikevich 与 H-H 等脉冲神经元模型和在线的突触学习, 能够模拟大规模 SNN 模型.

Loihi[176]是英特尔设计的全数字神经形态芯片, 采用 14 纳米工艺技术制造, 包含 20.7 亿个晶体管, 128 个内核, 单核包含最多 1024 个神经元和 16MB 突触容量, 可实现多达 13 万个神经元和 1.3 亿个突触. 此外, 每个内核中嵌入的学习引擎可实现片上学习, 支持多种神经形态可塑性、复杂的神经元模型和信息编码协议, 能模拟各类 SNN 模型, 具有非常高的灵活性. Loihi 同时支持稀疏网络压缩、内核间多播通信、可变突触格式和基于种群的分层互连等机制, 拥有远超传统处理器的计算能效. 在 Loihi 芯片的基础上, 英特尔还推出了 Pohoiki Springs[177]. 该系统是英特尔最大的机架式神经形态系统, 集成了 768 个 Loihi 芯片, 拥有 1 亿个神经元和 990 亿个突触, 运行功耗低于 500 瓦, 整体神经容量与小型哺乳动物相当.

Tianjic[178]是清华大学类脑计算研究中心设计的异构融合神经形态计算芯片, 尺寸为 14.44 $mm^2$, 采用 28nm 工艺, 由 150 多个计算核心组成, 可满足约 4 万个神经元和 1 千万个突触的计算. Tianjic 芯片的最大特点是将计算科学和神经科学两个不同的人工智能研究方向进行了整合, 既能支持现有的机器学习算法, 也能支持神经形态计算范式, 很好地融合了 ANN 与 SNN 各自的优势.

Darwin[179]是浙江大学研究的全数字神经形态芯片, 采用 55 纳米工艺, 包含 576 个计算内核, 每个内核可实现约 256 个神经元和 1000 万个突触的计算. Darwin 芯片具有高度的可配置性, 支持神经元、突触及突触延迟的自主配置, 适用于手势识别、图像识别、语音识别、脑电识别等多种应用.

全数字实现的电路具有很强的可编程性与灵活性, 支持 SNN 模型各种参数的配置, 大大缩短了开发时间, 而且不受电源、热噪声或器件不匹配的影响. 在片上通信方面, 数电实现也能提供相当的稳定性和可靠性. 然而, 全数字实现的电路会带来更多的硅片面积占用以及单位功能功耗.

3.2.3 神经形态芯片的现状总结与发展趋势

对于上述两小节介绍的数模混合神经形态芯片与全数字神经形态芯片, 表 6 进行了总结.

**表 6** 神经形态芯片总结

| 名称 | 类型 | 核心 | 神经元 | 突触 | 工艺(nm) | 面积(mm²) | 能耗 | 片上学习 |
|---|---|---|---|---|---|---|---|---|
| Neurogrid[169] | 数模混合 | 1 | 65k | 100M | 180 | 168 | 2.7W | 否 |
| BrainScaleS[170] | 数模混合 | 352 | 512(单核) | 130k(单核) | 180 | 50(单核) | 174pJ/SOP | 是 |
| ROLLS[171] | 数模混合 | 1 | 256 | 128k | 180 | 51.4 | 4mW | 是 |
| DYNAPs[172] | 数模混合 | 4 | 1k | 64k | 180 | 43.79 | 17pJ/SOP | 否 |
| TrueNorth[173] | 数字电路 | 4096 | 1M | 256M | 28 | 430 | 65mW(单核) | 否 |
| SpiNNaker[174] | 数字电路 | 18 | 1k(单核) | 1M(单核) | 130 | 102 | 1W(单核) | 是 |
| Loihi[176] | 数字电路 | 128 | 131040 | 130M | 14 | 60 | 45W | 是 |
| Tianjic[178] | 数字电路 | 156 | ~40000 | ~10M | 28 | 14.44 | 937mW | 否 |
| Darwin[179] | 数字电路 | 576 | 32768 | 1B | 180 | 25 | 0.84 mW/MHz | 否 |

目前主流的神经形态芯片通常采用数模混合电路或全数字电路实现, 在架构设计上通过片上网络(Network on Chip, NoC)[180]连接多个神经形态计算核心, 形成大规模集成芯片. 虽然这两种方法都利用先进的技术模拟出了上亿神经元规模的 SNN, 但是在制造工艺上, 由于 CMOS 电路受限于二维连接和有限的互连金属及路由协议, 复现真正的三维生物大脑结构仍然存在巨大困难, 神经元之间的大规模突触连接和突触自适应的物理实现是目前主流架构的最大瓶颈. 为了克服传统 CMOS 工艺存在的问题, 选择具有理想性能的忆阻器正成为部署大规模 SNN 的解决方案之一[181].

除了前两节提到的两种电路实现方式外, 基于随机计算实现的神经形态芯片也是一种理想的部署 SNN 的硬件方案. 随机计算[182]最早于 20 世纪 60 年代提出, 用于简化复杂的二进制计算单元, 运算时将二进制数转换成概率编码的数字脉冲码流的逻辑计算, 与 SNN 的速率编码方案十分类似[183], 能通过简单的逻辑门实现加法、减法、乘法等复杂运算. 将随机计算添加到神经形态芯片中, 可以实现毫瓦级的超低功耗和高计算效率的电路, 并大大优化芯片的面积占用, 具有良好的发展前景.

### 3.3 非易失性存储器

不论是数模混合实现还是全数字电路实现, 这两类神经形态芯片都是基于 CMOS 工艺制成的. 尽管 CMOS 技术在模拟 SNN 中处理和存储元件的逻辑相邻关系方面取得了重大进展, 然而出于前文提到的缺陷, 探索新的工艺仍有必要. 首先, 为了提高神经形态计算的效率, 需要更高的片上存储密度, 这是 CMOS 工艺很难达到的; 其次, 在 CMOS 工艺中模拟复杂的神经元和突触功能会导致高面积消耗, 不利于大规模神经形态系统的搭建. 于是, 忆阻器件进入了研究者们的视野[184]. 忆阻器与传统 CMOS 晶体管在基本功能上的主要区别在于它们是非易失性可编程模拟电阻器, 这使得研究者们能够在底层器件的电阻状态中直接模拟神经形态计算中的神经元与突触行为, 实现更具生物合理性的神经网络.

#### 3.3.1 忆阻器

忆阻器(Memristor)的概念由 chua 等人于 1971 年被提出[185], 它是电路中假设的第四种无源元件, 将磁通量的变化与流经该元件的电荷变化联系起来. 在数学上, 它等同于一个非线性电阻, 会根据电流的变化改变其电阻值, 因此, 它被称为忆阻器, 即记忆电阻器. 2008 年, 惠普实验室首次实现了忆阻器的工艺[186], 这种工艺允许开发者根据底层设备的特性, 在硬件平台上直接模拟神经元以及突触的功能, 同时支持高片上存储密度和大规模并行计算, 非常适合神经形态硬件平台的搭建.

关于忆阻器的计算原理, chua 等人[185]指出, 应有六种不同的数学关系将电流$I$、电压$V$、电荷$q$和磁通量$\phi$这四个基本电路变量对连接起来. 其中, 电荷是电流的时间积分这一关系是根据两个变量的定义确定的, 而磁通量是电动势或电压的时间积分是根据法拉第感应定律确定的. 因此, 根据其余的变量之间的关系, 应该有四个基本电路元件, 如图 13 所示, 忆阻器描述的是电荷和磁通量之间的函数关系, 即$d\phi = Mdq$.

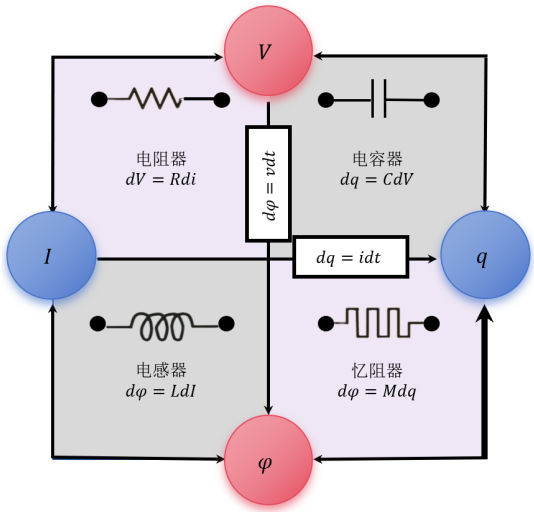


图 13　四个基本电路变量的联系

1976 年, chua 等人[187]将忆阻器的概念推广到更广泛的非线性动力学系统中, 并称之为忆阻器系统, 其描述方程为:

$$I(t)=\frac{V(t)}{R(x,V)} \tag{13}$$

$$\frac{dx}{dt}=f(x,V) \tag{14}$$

其中, $I(t)$ 是流经系统的电流, $V(t)$ 是电压的改变量, $R(x,V)$ 是记忆电阻, $x(t)\in R_m$ 是描述系统内部状态的 $m$ 维动态变量, $f:R_m\times R\to R_m$ 是非线性函数.

得益于忆阻器工艺的发展, NVM 技术在神经形态计算领域大放异彩. 由于表面积小且易于集成, 忆阻器通常用于 NVM 阵列的搭建, 如基于堆叠技术的三维阵列和横杆型二维阵列, 以实现神经形态计算所必需的高度紧凑、高能效的存内计算内核. NVM 的横杆阵列结构如图 14 所示, 在 NVM 器件中, 每个单元的电导状态都编码了相应的突触权重, 输入脉冲作为电压沿横杆阵列的行施加, 流经每个单元的电流根据器件电导加权, 并沿着每个阵列的列进行求和, 以实现点积运算. 横杆阵列还可与电阻神经元器件连接, 以实现神经元的信息处理.

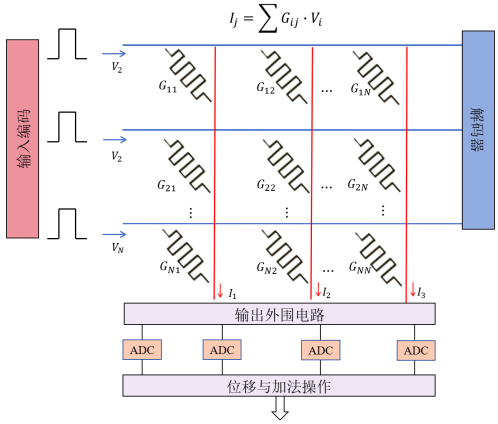


图 14　非易失性存储器的横杆阵列

除了执行高效的突触计算外, 基于 STDP 规则的无监督学习也已经在 NVM 交叉横杆中得到了实现[188-190].

如图 15 所示, 横杆阵列的一条边代表突触前神经元, 一条正交边代表突触后神经元, 连接到这些后神经元的导线上的电压代表膜电位, 根据突触前和突触后神经元中脉冲的时间来修改 NVM 单元的电导, 在两个 NVM 忆阻单元上分离长期增强(Long-Term Potentiation, LTP)和长期抑制(Long-Term Depression, LTD)功能即可实现 STDP 规则.

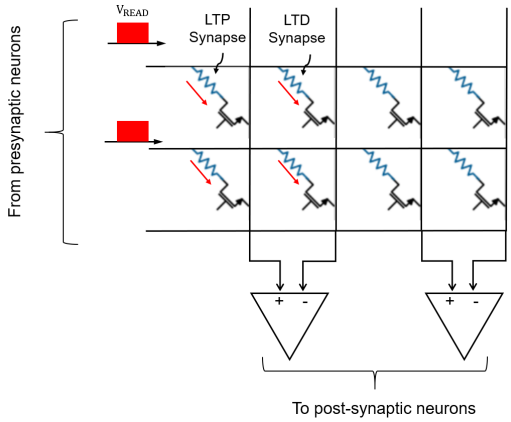


图 15 在横杆阵列上实现 STDP 规则[191]

目前, 已经有一些小型网络模型展示了基于忆阻器 NVM 阵列的高效运算[192-194], 这种横杆阵列的结构为实现大规模并行和高能效的神经形态硬件平台提供了一种有效的途径. 然而, 受限于忆阻器系统本身的集成难度和仿生物突触学习规则的限制, 基于忆阻器的神经形态计算仍处于利用器件进行原理探索与验证的阶段, 大规模的忆阻器 SNN 暂时仍未被报道[195].

3.3.2 NVM 实现工艺

目前, 主要有三种工艺来实现 NVM 技术, 分别是可变电阻式存储器(Resistive Random Access Memory, RRAM)、相变存储器(Phase Change Materials, PCM)和自旋电子学(Spintronics).

(1) 可变电阻式存储器

RRAM 是以非导性材料的电阻在外加电场作用下, 在高阻态和低阻态之间实现可逆转换为基础的 NVM. 绝缘介质层(阻变层)被夹在两层金属之间, 当偏压变化时电阻会在高、低两种状态间切换, 阻变层中的导电通路会随之呈现导通或断开两种状态, 从而实现了“0”,“1”状态的区分和存储. 目前, 一共有 24 种二元金属氧化物具有阻变特性, 可充当氧化物介质, 10 种金属材料可用作两端电极[196], 图 16 展示了 RRAM 的结构.

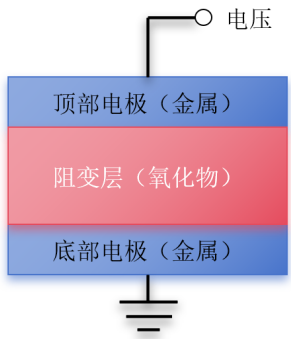


图 16 电阻式随机存储器

RRAM 器件可以通过与外部电容平行连接来实现 LIF 神经元, 此时内部膜电位编码在 RRAM 的电导中.

当 RRAM 处于导通状态时, 通过电路的电流会突然增加, 从而产生模拟脉冲, 两端的电压代表 LIF 特性.

文献[197]为了在 SNN 的硬件实现中密集集成突触, 选择在一个突触内并行使用多个 RRAM, 当进行突触读取时, 所有 RRAM 都会同时被读取, 而当突触电导改变时, STDP 机制只在一个随机挑选的 RRAM 上启动. 实验证明, 随着更多的 RRAM 被添加到突触中, SNN 的训练得到了明显改善.

文献[198]利用 RRAM 器件, 开发了一种基于存内计算架构的高能效、高精度 SNN 硬件, 支持 STDP 的在线学习. 仿真结果表明, 在使用 MNIST 数据集进行评估时, 所提出的架构具有很高的能效, 每次脉冲的能耗约为 20 fJ, 同时保持了 95%的推理准确率.

(2) 相变存储器

几乎所有材料, 包括金属、半导体和绝缘体, 都可以存在非晶相和晶体相. 然而, 在这些材料中, 只有极少数材料同时具备所有特性, 使其可用于数据存储技术, 以材料相的形式存储信息. 而这些相变材料正是 PCM 存储技术的核心, 它们可以在电流流过电极时产生的热效应作用下, 在非晶态和晶体态之间切换[199]. PCM 材料的原理如图 17 所示.

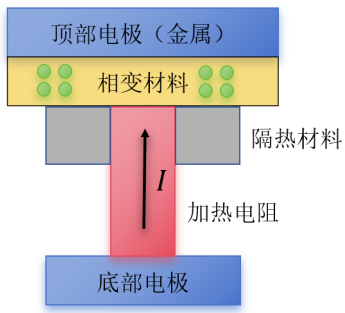


图 17 相变存储器

在神经形态计算领域, PCM 器件的可变电流可用于实现 IF 脉冲神经元, 其中膜电位通过连续的结晶脉冲进行时间整合, 器件在超过阈值后会转变为结晶状态, 随后又会复位为非晶状态, 与 IF 脉冲神经元的行为表现一致.

文献[200]将 SNN 与生物启发训练算法相结合, 并在基于 PCM 忆阻器的神经形态硬件上实现了这种方法. 这种方法实现的 SNN 不仅具备片上学习能力, 而且对硬件缺陷具有鲁棒性, 可在 4 位精度下实现与浮点 32 位相当的性能, 同时与纯软件实现的 SNN 性能旗鼓相当.

文献[201]提出的 eSpine 是一种基于 PCM 忆阻器的神经形态计算模型, 通过在映射机器学习工作负载时考虑每根横杆内的耐久性变化, 确保具有较高激活度的突触始终在耐久性较高的忆阻器上实现, 并保护临界忆阻器(耐久性较低的忆阻器)不被过度使用, 最终显著提高了 SNN 部署时的使用寿命.

(3) 自旋电子学

自旋电子学是凝聚态物理学中研究电子自旋特性的一个领域, 其目的是提高电子设备的效率, 在信息存储方面拥有巨大潜力[202]. 近年来, 自旋转移矩磁随机存储器(Spin Transfer Torque-Magnetic Random Access Memory, STT-MRAM)技术取得了长足进步, 已成为嵌入式和独立式 NVM 的突破性技术, 可提供快速编程和高耐用性[203]. 在后续的研究中, STT-MRAM 还展示出了实现突触行为的能力, 可以作为随机记忆器件[204].

磁性隧道结(Magnetic Tunnel Junctions, MTJ)是磁性随机存储器等自旋电子学器件的核心结构, 由两个铁磁性纳米磁体和夹在它们中间的间隔层(氧化镁)构成[205]. MTJ 可根据两个调频层的相对磁化方向以两种不同的电阻状态存在, 其自旋动力学方程如式 15 所示:

$$\frac{\partial \hat{m}}{\partial t} = -|\gamma|(\hat{m} \times H_{EFF}) + \alpha\left(\hat{m} \times \frac{\partial \hat{m}}{\partial t}\right) + \frac{1}{qN_s}(\hat{m} \times I_s \times \hat{m}) \tag{15}$$

其中$\hat{m}$是自由层磁化的单位矢量, $H_{EFF}$是有效磁场, 包括形状各向异性磁场、外部磁场和热磁场, $\gamma$是电子

的回旋磁化率, $\alpha$是吉尔伯特阻尼比. 方程的前两项通常用来表示泄漏行为, 最后一项表示积分行为. 这一特点使得 MTJ 材料与 LIF 脉冲神经元的相性极佳.

文献[206]提出了一种由 MTJ 支持的概率深度 SNN, 可将已完全训练好的 ANN 转化成前馈结构的 SNN. 这种 SNN 模型使用速率编码将 ANN 输入转换为泊松脉冲序列, 经过突触权重调整, 产生的突触后电流流经 MTJ 装置下方的重金属. 如图 18 所示, 图中的重金属作为概率脉冲神经元, MTJ 位于重金属顶部, 在存在热噪声的情况下, 流经重金属的写入电流会随机切换 MTJ. 仿真结果表明, 原 ANN 对 MNIST 手写数字识别的准确率为 98.56%, 而经过 MTJ 元件转换得来的 SNN 的测试准确率仍高达 97.6%. 与基线数字 CMOS 实现相比, 基于 MTJ 的实现在能效上提高了 20 倍.

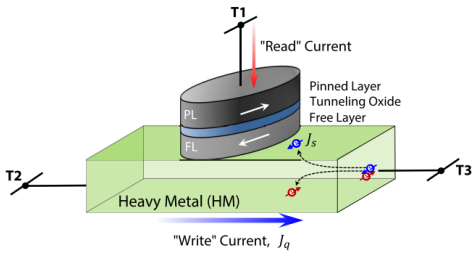


图 18 磁性隧道结[204]

文献[207]提出了一种基于 STT-RAM 的神经突触内核, 用于实现 SNN 的硬件加速器. 其计算核心包括一个由非易失性 STT-RAM 组成的交叉横杆阵列、IO 外围电路和用于脉冲神经元的数字逻辑, 避免了对昂贵的外围数模/模数转换电路的需求, 在单位能耗和单位面积吞吐量上比基于静态随机存储器的同等设计高出约 6 倍.

### 3.4 硬件映射方法

研究表明, 虽然软件仿真具有高灵活度、高精度的特点[208], 但不能充分利用神经网络的高并行性, 而且处理速度慢, 功耗高. 硬件实现可以改善软件仿真的不足, 充分体现神经网络的高并发特性[209]. 因此, 有必要研究硬件映射方法, 进而高效地将 SNN 模型部署在各类硬件平台上.

与传统的冯·诺依曼体系结构不同, 在神经形态硬件中, 计算单元(神经元)和存储单元(突触)像横杆一样分布在硬件内部, 每个横杆限制了每个突触后神经元允许有多少个突触前连接和多少缓冲空间可用于互连发送和接收脉冲. 这些硬件资源和分布方式既影响模型的准确性, 也影响吞吐量、延迟和能耗等硬件性能. 将 SNN 映射到横杆的原理如图 19 所示, 自突触前神经元的脉冲向横杆注入电流, 电流是沿行施加的脉冲电压(即输入激活$x_i$)与交叉点突触元件电导(即突触权重$w_{ij}$)的乘积. 沿列的电流求和是并行进行的, 实现了神经元兴奋$x_i$向前传播所需的$\sum w_{ij}x_i$求和值.

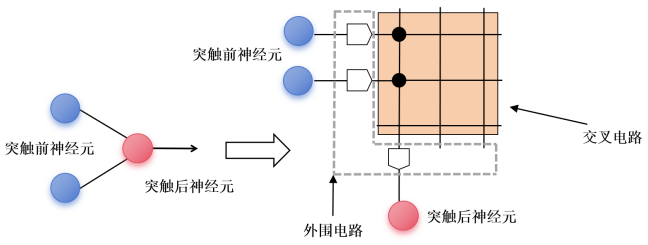


图 19 SNN 映射横杆阵列

为了降低能耗, 横杆的尺寸受到限制, 每个神经元只能容纳有限数量的突触. 在大型芯片中,会使用 NoC 这样的方式将多个横杆集成在一起. 然而, 大规模 SNN 需要的大量内核间通信会导致 NoC 拥塞, 并显著降低

性能. 因此, 有必要设计合适的硬件映射方法, 使 SNN 模型的部署更高效.

随着 SNN 规模的扩大, 将所有神经元映射到单个内核中并不可行. 一种可能的解决方案是将 SNN 分割并映射到多核神经形态硬件上. 然而, 将神经元映射到具有最低脉冲延迟和能耗的不同内核已被证明是一个 NP-complete 难题[210]. 目前, 将 SNN 分割并映射到多核神经形态硬件的主要过程如图 20 所示, 在分区前, 需要使用仿真工具(如上一章提到的 NEST 和 CARLsim 等)对 SNN 进行模拟, 然后统计分析网络的各项仿真性能, 再将 SNN 模型划分为多个簇, 保证每个簇的神经元数量不超过单个神经形态内核的容量, 接着选择适当的内核执行 SNN 的分区应用, 最后将 SNN 模型部署到硬件上.

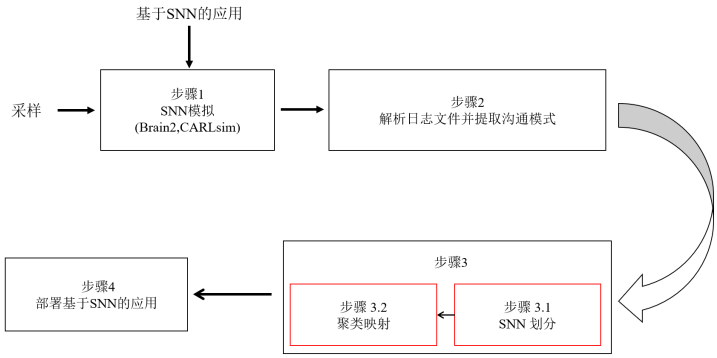


图 20 SNN 映射多核神经形态硬件

SpiNeMap[211]是一种将 SNN 映射到基于横杆阵列的神经形态硬件的方法, 分两个步骤运行 SpiNeCluster 和 SpiNePlacer. SpiNeCluster 是一种基于启发式的聚类技术, 可将 SNN 划分为突触群组, 群组内的局部突触映射到硬件的横杆上, 群组间的全局突触映射到共享互连上. 然后, SpiNePlacer 采用元启发式方法在硬件上找到本地和全局突触的最佳位置, 最大限度地降低能耗和脉冲延迟.

MAMAP[212]是一种缓解 NoC 通信拥塞的基于记忆算法的映射方法, 所提出的记忆算法结合了粒子群优化算法[213]的全局搜索能力和 TABU[214]的局部搜索能力, 可以快速找到最佳映射方案, 通过最大化带宽利用率来最小化神经形态硬件的延迟和功耗. 与 SpiNeMap 相比, MAMAP 可将平均延迟降低 63%, 平均能耗降低 69%.

SNEAP[215]的映射方法和 SpiNeMap[201]类似, 在此前的基础上采用了一种有效的图分割方法来提高分割质量, 同时大幅缩短分割时间, 并使用平均跳数来评估平均神经元通信延迟和功耗, 而不是使用模拟器来评估平均神经元通信延迟和功耗, 与 SpiNeMap 相比, SNEAP 的端到端执行时间平均减少 418 倍, 能耗和脉冲延迟平均分别减少 23%和 51%.

NeuMap[216]是一种简单快速的将 SNN 映射到多核神经形态硬件上的方法. 该方法首先通过计算获得 SNN 的通信模式, 再利用局部连接, 将相邻层划分为子网络, 并将每个子网络划分为多个集群, 同时满足硬件资源限制. 最后采用元启发式算法, 在缩小的搜索空间内寻找最佳的簇到核映射方案. 实验结果表明, 与 SpiNeMap[211]和 SNEAP[215]相比, NeuMap 的平均能耗分别降低 84%和 17%, 脉冲延迟分别降低了 55%和 12%.

针对当下流行的神经形态芯片,已经有一些专用的映射方法出现, 如 SpiNNaker 的 PACMAN[217]、TrueNorth 的 Corelets[218]、TianJic 的 TJSim[219]和 Loihi 的 NxTF[220]. 除此之外, 文献[221]提出了一种模块化构建 SNN 分区的映射方法, 可将 SNN 模型部署到多核处理器上, 其支持 TrueNorth 和 Loihi 等众多流行的神经形态芯片, 与运行在传统 GPU 上的 ANN 相比, 精度相当, 但在功耗指标上明显更优.

DFSynthesizer[222]是一个端到端映射框架, 用于将基于 SNN 的机器学习算法映射到神经形态硬件. 该框架分为四个步骤. 首先, 分析机器学习程序, 并使用代表性数据生成 SNN 工作负载. 其次, 对 SNN 工作负载进

行分区, 生成适合目标神经形态硬件横杆的集群. 第三, 利用同步数据流图的丰富语义来表示聚类 SNN 程序, 从而可以根据关键硬件约束条件进行性能分析, 如横杆数量、每条横杆的尺寸、芯片上的缓冲空间和通信带宽. 最后, 使用调度算法在硬件的横杆上执行集群, 从而保证了硬件性能. 结果表明, 与当前的映射方法相比, DFSynthesizer 提供了更严格的性能保证.

E3NE[223]是一个可自动生成适用于 FPGA 的高效 SNN 模型的端到端映射框架. E3NE 使 SNN 模型的推理能够部署在几乎任意大小的 FPGA 上, 该框架根据应用要求和硬件限制优化硬件模块的实例化和配置, 从而实现高性能, 同时采用动态量化方案生成脉冲序列, 并从预先训练的 SNN 模型转换权重. E3NE 是首个基于寄存器传输层(Register Transfer Level, RTL)硬件块的 SNN 端到端映射框架, 在准确性、延迟、功耗和硬件资源方面均优于之前的基于 FPGA 的 SNN 加速器.

尽管过去 20 年来, 人们一直在研究将大脑运作原理映射到神经形态硬件上[224], 但这些现有的映射方法大多仅探讨了其性能特征. 如上所述, 目前的主流硬件映射方法基本采取的是先分割再映射的步骤, 将关注点放在了资源分配、脉冲延迟, 通信拥塞和能耗等性能指标上, 忽略了映射方法的可靠性与鲁棒性. 然而, 由于外部干扰或其他神经元的随机行为, 有些神经元可能会受到噪声[225,226]的影响, 导致计算结果不可靠[227], 尤其是将神经网络映射到硬件中时, 模型不可避免地会受到硬件级故障的影响[228]. 因此, 未来还需要在保证性能的同时, 提出更具可靠性和鲁棒性的硬件映射方法.

## 4 硬件部署的故障与容错

从实际生产的角度来看, 许多计算设备在量产时不可避免地会出现制造缺陷. 为了充分利用未来的半导体技术, 有必要找到经济有效的方法, 使得从一开始就能利用这些不完美的元件, 甚至在元件功能随时间退化的情况下也不影响整体功能. 于是, 神经网络模型自身的容错特性对于成功将大型神经网络集成到边缘设备上至关重要. 优秀的故障与容错机制能提供可靠的方法来保持机器的状态, 从而保证可靠和不间断的设备性能[229].

根据神经生物学的研究, 人脑能够容忍少量的突触或神经元故障, 甚至将噪声作为计算的来源[227]. 在第二代 ANN 中, 也具备一定对不精确、不确定性和故障的容忍度[230]. 作为更具生物启发性的第三代神经网络, SNN 也被认为是更具前途的容错神经网络, 具有固有的容错性、鲁棒性和恢复性[231]. 然而文献[228]指出, 在实践中,如果没有适当的设计, 神经网络不能被认为是本质上的容错. 此外, 传统计算中使用的标准容错技术, 如三重模块冗余(Triple Module Redundancy, TMR)和用于存储器的纠错码(Error Correction Code, ECC), 会在神经形态硬件平台上失效, 因为这些冗余方法会产生过高的开销, 更不利于复杂模型的硬件部署.将 SNN 部署到边缘设备上时, 目前的大多数工作都将重点放在以降低能耗为代表的性能优化上. 尽管性能最大化是当之无愧的首要关注点, 但这并不一定意味着它是应该追求的唯一目标[232]. 随着硬件底层半导体技术变得越来越不可靠, 计算设备的某些组件失效的可能性也在增加, 这使得许多 SNN 模型在硬件上的部署无法彻底发挥神经形态硬件中百亿级集成的全部潜在优势. 因此, 当硬件故障问题不可避免时, 可靠的容错与恢复策略有助于减少灾难性错误的发生, 为计算设备提供更强的健壮性与鲁棒性.

### 4.1 故障与容错介绍

在容错神经系统中有三个基本概念, 即故障、错误和失败. 三个概念对应着从硬件层经过网络模型到行为层, 它们之间存在因果关系, 如图 21 所示:

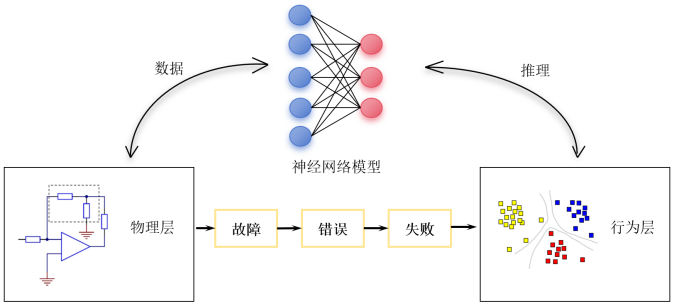


图 21 故障与容错系统

故障是系统中引起错误的异常物理状态; 错误则是神经网络模型中故障的表现形式, 表现为网络中组分的逻辑状态与其预期值不同; 失败是指系统由于其元件中的错误或环境中的扰动而无法执行其预期的功能或行为[233]. 将错误传播到行为层会导致系统功能失败, 但是, 硬件层中的故障不一定会导致错误或失败, 因为这个故障可能会失效, 未对顶层产生影响.

针对容错神经系统中最底层的硬件故障, 可以分为两类: 软故障和硬故障[234,235]. 软故障是由运行过程中出现的不同周期间或器件间的变化引起的, 包括读/写操作过程中的状态变化, 可能只持续很短的时间, 通常是外部干扰的结果, 也称为瞬时故障. 硬故障是由制造步骤引起的, 也可能是由成型过程或持续应力引起的, 这种故障是连续的, 不随时间改变的, 主要是由不可逆转的物理损伤造成的, 也称为永久故障. 在当前半导体技术构建的数字计算系统中, 绝大多数故障都是软故障[236], 软故障比硬故障更难检测.

对于神经网络的容错性, 如果从神经网络$\mathcal{N}$得到的故障网络$\mathcal{N}_{fault}$所执行的计算$\mathcal{H}_{\mathcal{N}_{fault}}$能够接近$\mathcal{H}_{\mathcal{N}}$, 则执行计算$\mathcal{H}_{\mathcal{N}}$的神经网络$\mathcal{N}$被称为容错网络. 从形式上讲, 对于$\epsilon > 0$的情况, 如果$\mathcal{N}$能够容忍任何大小最多为$\mathcal{N}_{fails}$的子集的故障组件(例如神经元/突触), 则称$\mathcal{N}$为 $\epsilon-$容错神经网络[237], 如式 16 所示:

$$\left\|\mathcal{H}_{\mathcal{N}}(\mathcal{X}) - \mathcal{H}_{\mathcal{N}_{fault}}(\mathcal{X})\right\| \leq \epsilon, \forall \mathcal{X} \in \mathcal{T} \tag{16}$$

其中, $\mathcal{X}$是应用于网络$\mathcal{N}$和$\mathcal{N}_{fault}$的任何刺激, 属于训练集$\mathcal{T}$或输入数据的一部分. 给定一个任务, 容错的目标是确定一个网络$\mathcal{N}$, 该网络既能执行所需的计算, 又具有相对于$\mathcal{T}$的$\epsilon-$容错特性. $\epsilon$可以理解为网络功能的故障阈值, 低于这个阈值, 网络就不能再按照预期执行其功能. 因此, 神经网络的容错取决于性能可接受程度的定义及其预期应用[238].

### 4.2 脉冲神经网络的容错研究

早在 90 年代初, 就有一些关于 ANN 容错性的研究成果问世, 但在随后的 20 年间, 人们几乎遗忘了这一问题[239]. 近年来, 随着边缘人工智能的研究发展, 越来越多的工作将神经网络模型部署到嵌入式系统中, 这些边缘硬件设备或由于自身材料制造问题带来硬故障, 或因为持续性工作发生软故障, 故障的产生给装置的正常运行带来隐患. 因此, 如何处理边缘设备运行 AI 应用时出现的硬件故障已成为当下亟待解决的问题之一.

与 ANN 相比, SNN 被认为继承了生物大脑卓越的容错能力, 能够抵御由制造缺陷、降压内存操作、辐射和老化等引起的硬件故障. 然而, 在最近的故障注入实验中, 这种假设被证明并不成立[240].

为了确定 SNN 的故障模型, 文献[241]在仅考虑由制造缺陷和老化现象引起的硬故障的前提下, 首次尝试对实现 SNN 的元件中可能出现的故障进行分类定义. 故障模型包括最坏情况下的故障行为, 如死神经元故障和死突触故障, 其中死突触故障以零权重建模. 结果表明, 要想明显降低识别率, 需要较高的故障密度. 根据过往理论研究[239,242], 故障神经元比故障突触对神经网络行为的影响更大, 同时, SNN 中使用的在线学习算法能有效减轻突触变化或输入噪声对网络鲁棒性的影响.

文献[243]研究了前馈结构的 SNN 在使用不同算法训练时对死突触故障的恢复能力. 结果表明, 复原特性在很大程度上取决于训练算法, 而且在所有情况下, 准确率都会随着故障率的增加而迅速下降.

文献[244]利用大规模故障注入实验的观察结果, 确定了关键故障类型和位置. 故障模拟显示, 发生在网络中任何位置的饱和神经元故障都可能是致命的, 而所有其他类型的故障如果发生在最后几层的神经元中, 都会影响分类的准确性. 为了解决关键神经元的故障, 该工作为 SNN 提出了一种神经元容错策略. 该策略分为两步, 在第一个准备步骤中, 使用 dropout[245]训练 SNN, 使部分层的某些神经元故障类型处于被动状态. 在第二步中, 执行主动容错, 检测所有层中剩余的神经元故障并从中恢复.

为了使 SNN 性能效率最大化, 很多研究设计并使用了专用的硬件加速器[246,247]. 然而, 这种加速器容易受到软故障的影响, 这是由高能粒子撞击引起的, 并表现为硬件层的位翻转, 如图 22 所示, 这些误差会改变 SNN 加速器计算引擎中的权重值和神经元操作, 从而导致输出错误和精度下降. SNN 硬件加速器对硬件故障的恢复能力取决于网络拓扑结构、电路实现方式和大小, 以及训练算法、正在执行的任务和预期故障率.

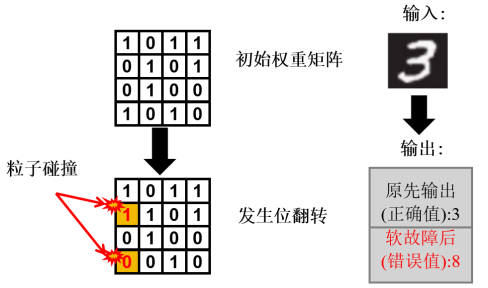


图 22 SNN 的软故障[249]

针对 SNN 硬件加速器中出现的软故障, 文献[248]提出了一种新的方法——SoftSNN, 可以在不采取冗余执行策略的情况下减轻 SNN 加速器的权重寄存器(突触)和神经元中的软故障, 从而在低延迟和低功耗的情况下保持准确性. 实验结果表明, 对于 900 个神经元的高故障率网络, 与冗余方法相比, SoftSNN 将准确率下降保持在 3%以下, 同时将延迟和能量分别降低了 3 倍和 2.3 倍.

文献[249]基于排序和选择映射机制, 提出了一种基于 NoC 的神经形态硬件的容错 SNN 映射算法和架构, 目标是在几乎不影响成本的情况下, 从高级故障的影响中完全恢复系统, 并允许对神经元进行排序和快速选择, 以实现容错映射. 评估结果表明, 与之前的映射框架相比, 所提出的机制可以在 20%的冗余率和 40%的故障率下保持 100%的映射效率.

为了提高神经形态计算的容错能力, 未来还需要进行更深层次的探索, 包括但不限于以下方面:

I) 全面的故障模型. 随着制造工艺技术的不断发展, 更现实的故障模型需要基于对现代制造技术的深刻理解来开发, 由于故障行为的多样性以及许多条件下的不可预知性, 如何建立更全面的故障模型将是该领域最值得关注的问题之一.

II) 架构和应用层面的容错. 由于神经网络模型的规模和复杂性正在迅速增长, 因此需要设计高效的大规模容错机制, 充分发挥神经系统架构的固有特性, 以支持未来不断更新发展的边缘智能应用的容错行为.

III) 生物启发式的容错机制. 寻找生物学上合理的容错机制有助于启发神经形态计算的容错技术发展, 包括探索主动容错原理和生物细胞的自我修复机制等. 目前, 已经有一些研究利用生物学中星形胶质细胞的修复原理, 实现了更具容错能力的 SNN 模型以及边缘部署[250,251].

## 5 总结与展望

人工智能技术的蓬勃发展为我们的世界带来了天翻地覆的变化. 尽管 DNN 在过去十几年中取得了卓越成果, 但随着 IoT 生态的发展, 越来越多的边缘设备需要智能化部署, 如何在资源严格受限的硬件平台中实现人工智能模型是一项极具挑战的任务.

受生物学原理启发, 神经形态计算通过模拟生物神经系统的结构和功能, 为 IoT 和边缘智能应用提供了

一种高效节能的计算范式. 作为神经形态计算的代表, SNN 因其具有低功耗高能效等优势, 是实现边缘智能的理想模型.

在本文中, 我们首先介绍了 SNN 的模型原理, 并针对模型在硬件实现时面临的问题充分论述了可行的解决方向. 随后总结了目前将 SNN 部署到边缘设备时常用的软件工具链. 接着重点叙述了 SNN 在主流神经形态硬件平台上的部署现状与发展方向. 最后针对硬件故障这一无法避免的问题, 对 SNN 的容错技术进行了整理与分析.

近年来, 神经形态计算领域取得了显著进展, 但 SNN 的边缘部署仍面临一些亟待解决的难题. 本文将分析和总结当前研究中存在的局限和挑战, 并提出可能的解决方案.

(1) 脉冲神经网络模型缺陷

尽管 SNN 在能效与生物合理性上表现优异, 但 SNN 模型自身仍存在一些局限性.

I) 脉冲神经元计算性能与生物可信度的矛盾. 尽管目前广泛采用的 LIF 模型在计算上具有低成本的优势, 但其相较于其他脉冲神经元模型在生物学上的置信度较低, 因此, 如何权衡脉冲神经元的学习能力和生物合理性是一个值得深入研究的课题.

为应对这一挑战, 可以探索新的学习规则, 通过生物学原理的启发或者更精细的数学建模, 设计机制更优秀的脉冲神经元. 也可以针对具体的应用场景, 设计更合适的脉冲神经元. 例如, Tempotron 神经元由于其时序敏感的特性, 在处理序列数据时具有优势, 更适合应用于语音或者信号识别场景. 进化脉冲神经元能自适应地调整脉冲事件的速率, 更适合在功耗受限的边缘设备上应用.

II) 速率编码与时间编码的缺陷. 速率编码和时间编码各具优势, 也存在相应的缺陷. 例如, 速率编码仅关注时间窗内的脉冲数, 忽略了码间干扰, 不能充分利用脉冲序列中包含的时空信息, 而许多时间编码的方案虽然能更好地利用时序信息, 但使用了复杂的突触函数, 导致功耗增加, 不利于边缘设备部署.

为解决这一问题, 可以考虑将速率编码和时间编码相结合, 根据具体任务, 在网络层级之间或不同的神经元之间采用适当的编码方式. 目前, 已经有研究工作在此方向进行探索, 期待未来有更深入的进展. 除此之外,设计基于硬件平台的脉冲编码器也是一个值得研究的方向.

III) 脉冲神经网络训练存在难点. 在 SNN 的研究历程中, 如何有效地训练 SNN 模型始终是一个备受关注的话题. SNN 由于其脉冲稀疏性, 在功耗方面具有显著优势, 但也因为脉冲事件的不可微性, 无法直接使用成熟的反向传播算法进行训练, 目前也尚无公认的最佳训练方法. 尤其针对无法使用高算力平台进行训练的边缘部署场景, 如果边缘设备需要保持在线学习的能力,那么除了传统训练算法的性能指标外, 算法存储资源利用率、权重更新复杂度等指标也应当加入考虑. 因此, 选择合适的训练算法是 SNN 边缘部署的核心挑战之一.

为应对这一挑战, 优化现有的训练方法是一个长期的研究目标. 目前, 已经有许多 STDP 规则的变体,基于脉冲的反向误差传播方法也在不断发展, 主流 ANN 转换训练法训练的 SNN 在准确率方面已接近传统 ANN. 此外, 已经有工作尝试将更优秀的生物学机制融合到 SNN 的训练方法中, 生物合理性和模型学习能力的有机结合将是未来可行的方向之一.

IV) 传统静态模型结构固有的缺陷. 目前, 有许多相关工作可以看作是传统人工神经网络(如 CNN 和 RNN)与 SNN 的结合. 尽管这类 SNN 融合了传统结构在图像数据与序列数据处理时拥有的优势并继承了脉冲传播在低功耗的优点, 但其无法彻底解决传统 ANN 模型过于复杂时训练难度高, 可解释性弱, 以及对于硬件故障的容错能力差等问题.

为解决这一问题, 探索动态拓扑结构的 SNN 是一个可行的方向. 以 ESNN 为代表的动态模型能自适应地调整网络结构, 灵活地发挥所选神经形态硬件平台的特有优势, 更适用于嵌入式边缘场景.

(2) 神经形态硬件平台设计难点

除 SNN 模型本身的局限性外, 神经形态硬件平台在设计上存在相当困难的挑战.

I) 硬件编程门槛过高. 以 FPGA 平台为例, 在 FPGA 平台上的编程往往需要 AI 领域的研究人员对硬件描

述语言以及电路原理有所了解, 且 FPGA 平台缺乏像 TensorFlow, PyTorch 这样成熟的软件开发框架的支持, 开发门槛非常高. 对于神经形态芯片与其他最新非易失性存储工艺器件, 开发人员不仅需要使用专用的编程工具, 有时甚至需要在物理材料方面具备一定的知识.

设计完备的神经形态计算工具链是 SNN 边缘部署的重要方向之一. 首先, 高效且完善的工具链能很大程度地简化开发过程, 为领域的研究提供更便捷的条件. 目前已有的 SNN 软件编程框架以及仿真工具需要不断更新迭代, 以提供更高的兼容性和稳定性. 此外, HLS 高层次综合工具可以将高级编程语言转化为硬件级别的描述并自动生成电路, 能有效降低 SNN 模型在 FPGA 平台部署的门槛. 最后, 设计具备高可扩展性的硬件映射方法也是一个很好的研究思路.

II) 神经形态硬件存在性能瓶颈. 大多数神经形态硬件都是低功耗、小面积的器件, 这类硬件平台往往在内存等资源上非常有限, 不适合部署大规模复杂模型. 此外, 传统 CMOS 电路受到二维连接和有限的互连金属及路由协议的限制, 在模拟真实的三维生物大脑结构时仍然面临巨大挑战, 这是目前主流神经形态芯片架构的最大瓶颈.

未来, 探索高能效的神经形态硬件设计有多种方向. 比如采用更先进的工艺, 突破传统 CMOS 工艺的束缚, 或者开发更高效的忆阻器阵列. 也可以设计更优的硬件架构, 例如异质融合传统 ANN 与 SNN 两种计算范式以提高整体性能.

III) 硬件平台非完全可靠. 现有研究表明, 神经网络的容错能力实际上相当有限, 即使是被认为继承了生物大脑卓越容错能力的 SNN, 在近期的故障注入实验中也未能表现出预期的性能. 由于硬件平台的底层支持并非完全可靠, 许多 SNN 模型在硬件上的部署无法彻底发挥神经形态硬件的潜在优势.

为了提高 SNN 边缘部署任务整体的可靠性, 既可以从硬件平台切入, 结合生物学原理设计更可靠更稳定的神经形态硬件, 也可以从 SNN 模型入手, 设计更优秀的故障与容错机制, 提高边缘智能应用的鲁棒性.

综上所述, 本文对神经形态计算从 SNN 模型到边缘部署的情况做了详细的总结与分析, 旨在吸引不同学科的研究者, 通过跨学科的思想交流与合作研究, 推动神经形态计算领域的发展, 让人工智能技术遍布生活的每一个角落.

**References**:

[1] Bodria F, Giannotti F, Guidotti R, et al. Benchmarking and survey of explanation methods for black box models[J]. Data Mining and Knowledge Discovery, 2023, 37(5): 1719-1778.

[2] Kong X, Tang X, Wang Z. A survey of explainable artificial intelligence decision[J]. Syst. Eng. Theory Pract, 2021, 41: 524-536.

[3] Khan S, Rahmani H, Shah S A A, et al. A guide to convolutional neural networks for computer vision[J]. 2018.

[4] Chu Q, Ouyang W, Li H, et al. Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism[C]//Proceedings of the IEEE international conference on computer vision. 2017: 4836-4845.

[5] Zhou C, Sun C, Liu Z, et al. A C-LSTM neural network for text classification[J]. arXiv preprint arXiv:1511.08630, 2015.

[6] Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning[J]. arXiv preprint arXiv:1605.05101, 2016.

[7] Bojarski M. End to end learning for self-driving cars[J]. arXiv preprint arXiv:1604.07316, 2016.

[8] Chen C, Seff A, Kornhauser A, et al. Deepdriving: Learning affordance for direct perception in autonomous driving[C]//Proceedings of the IEEE international conference on computer vision. 2015: 2722-2730.

[9] Huang G, Sun Y, Liu Z, et al. Deep networks with stochastic depth[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer International Publishing, 2016: 646-661.

[10] Alom M Z, Rahman M M, Nasrin M S, et al. COVID_MTNet: COVID-19 detection with multi-task deep learning approaches[J]. arXiv preprint arXiv:2004.03747, 2020.

[11] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous activity[J]. The bulletin of mathematical biophysics, 1943, 5: 115-133.

[12] Minsky M, Papert S A. Perceptrons, reissue of the 1988 expanded edition with a new foreword by Léon Bottou: an introduction to computational geometry[M]. MIT press, 2017.

[13] Cybenko G. Approximation by superpositions of a sigmoidal function[J]. Mathematics of control, signals and systems, 1989, 2(4): 303-314.

[14] Funahashi K I. On the approximate realization of continuous mappings by neural networks[J]. Neural networks, 1989, 2(3): 183-192.

[15] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators[J]. Neural networks, 1989, 2(5): 359-366.

[16] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.

[17] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[18] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[19] Zhang M, Wang J, Wu J, et al. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks[J]. IEEE transactions on neural networks and learning systems, 2021, 33(5): 1947-1958.

[20] Maass W. Networks of spiking neurons: the third generation of neural network models[J]. Neural networks, 1997, 10(9): 1659-1671.

[21] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[J]. Advances in neural information processing systems, 2015, 28.

[22] Tavanaei A, Ghodrati M, Kheradpisheh S R, et al. Deep learning in spiking neural networks[J]. Neural networks, 2019, 111: 47-63.

[23] Stone J V. Principles of neural information theory[J]. Computational Neuroscience and Metabolic Efficiency, 2018.

[24] Wang B, Xue C, Liu H, et al. SNNIM: A 10T-SRAM based Spiking-Neural-Network-In-Memory architecture with capacitance computation[C]//2022 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2022: 3383-3387.

[25] Pan X, Luo W, Shuai Y, et al. Hardware implementation of edge neural network computing for sensor with memristors based on single-crystalline LiNbO 3 thin film[J]. IEEE Sensors Journal, 2023, 23(8): 8526-8534.

[26] Lee C, Sarwar S S, Panda P, et al. Enabling spike-based backpropagation for training deep neural network architectures[J]. Frontiers in neuroscience, 2020, 14: 497482.

[27] Wang Y, Liu H, Zhang M, et al. A universal ANN-to-SNN framework for achieving high accuracy and low latency deep Spiking Neural Networks[J]. Neural Networks, 2024, 174: 106244.

[28] Lien H H, Chang T S. Sparse compressed spiking neural network accelerator for object detection[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(5): 2060-2069.

[29] Liu K, Cui X, Ji X, et al. Real-Time Target Tracking System with Spiking Neural Networks Implemented on Neuromorphic Chips[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 70(4): 1590-1594.

[30] Feng L, Zhang Y, Zhu Z. An efficient multilayer spiking convolutional neural network processor for object recognition with low bitwidth and channel-level parallelism[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69(12): 5129-5133.

[31] Huang J, Serb A, Stathopoulos S, et al. Text classification in memristor-based spiking neural networks[J]. Neuromorphic Computing and Engineering, 2023, 3(1): 014003.

[32] Zou C, Cui X, Kuang Y, et al. A Hybrid Spiking Recurrent Neural Network on Hardware for Efficient Emotion Recognition[C]//2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2022: 332-335.

[33] Kumar N, Tang G, Yoo R, et al. Decoding eeg with spiking neural networks on neuromorphic hardware[J]. Transactions on Machine Learning Research, 2022.

[34] Mao R, Li S, Zhang Z, et al. An ultra-energy-efficient and high accuracy ECG classification processor with SNN inference assisted by on-chip ANN learning[J]. IEEE Transactions on Biomedical Circuits and Systems, 2022, 16(5): 832-841.

[35] Chu H, Yan Y, Gan L, et al. A neuromorphic processing system with spike-driven SNN processor for wearable ECG classification[J]. IEEE Transactions on Biomedical Circuits and Systems, 2022, 16(4): 511-523.

[36] Roggen D, Hofmann S, Thoma Y, et al. Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot[C]//NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings. IEEE, 2003: 189-198.

[37] Johnson A P, Liu J, Millard A G, et al. Homeostatic fault tolerance in spiking neural networks: A dynamic hardware perspective[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2017, 65(2): 687-699.

[38] Pfeiffer M, Pfeil T. Deep learning with spiking neurons: opportunities and challenges[J]. Frontiers in neuroscience, 2018, 12: 409662.

[39] Hodgkin A L, Huxley A F. A quantitative description of membrane current and its application to conduction and excitation in nerve[J]. The Journal of physiology, 1952, 117(4): 500.

[40] Gerstner W, Kistler W M. Spiking neuron models: Single neurons, populations, plasticity[M]. Cambridge university press, 2002.

[41] Paugam-Moisy H, Bohte S M. Computing with spiking neuron networks[J]. Handbook of natural computing, 2012, 1: 1-47.

[42] Nitzsche S, Pachideh B, Luhn N, et al. Digital hardware implementation of optimized spiking neurons[C]//2021 International Conference on Neuromorphic Computing (ICNC). IEEE, 2021: 126-134.

[43] Dayan P, Abbott L F. Theoretical neuroscience: computational and mathematical modeling of neural systems[M]. MIT press, 2005.

[44] Izhikevich E M. Which model to use for cortical spiking neurons?[J]. IEEE transactions on neural networks, 2004, 15(5): 1063-1070.

[45] Aamir S A, Stradmann Y, Müller P, et al. An accelerated LIF neuronal network array for a large-scale mixed-signal neuromorphic architecture[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2018, 65(12): 4299-4312.

[46] Izhikevich E M. Simple model of spiking neurons[J]. IEEE Transactions on neural networks, 2003, 14(6): 1569-1572.

[47] Heidarpur M, Ahmadi A, Ahmadi M, et al. CORDIC-SNN: On-FPGA STDP learning with izhikevich neurons[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2019, 66(7): 2651-2661.

[48] Jolivet R, Gerstner W. The spike response model: a framework to predict neuronal spike trains[C]//International Conference on Artificial Neural Networks. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003: 846-853.

[49] Qiao G C, Ning N, Zuo Y, et al. Direct training of hardware-friendly weight binarized spiking neural network with surrogate gradient learning towards spatio-temporal event-based dynamic data recognition[J]. Neurocomputing, 2021, 457: 203-213.

[50] Diehl P U, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity[J]. Frontiers in computational neuroscience, 2015, 9: 99.

[51] Mostafa H. Supervised learning based on temporal coding in spiking neural networks[J]. IEEE transactions on neural networks and learning systems, 2017, 29(7): 3227-3235.

[52] Rueckauer B, Lungu I A, Hu Y, et al. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification[J]. Frontiers in neuroscience, 2017, 11: 682.

[53] Panchapakesan S, Fang Z, Li J. SyncNN: Evaluating and accelerating spiking neural networks on FPGAs[J]. ACM Transactions on Reconfigurable Technology and Systems, 2022, 15(4): 1-27.

[54] Im J, Kim J, Yoo H N, et al. On-Chip Trainable Spiking Neural Networks Using Time-To-First-Spike Encoding[J]. IEEE Access, 2022, 10: 31263-31272.

[55] Bohte S M, Kok J N, La Poutre H. Error-backpropagation in temporally encoded networks of spiking neurons[J]. Neurocomputing, 2002, 48(1-4): 17-37.

[56] Yu Q, Tang H, Tan K C, et al. A brain-inspired spiking neural network model with temporal encoding and learning[J]. Neurocomputing, 2014, 138: 3-13.

[57] Comsa I M, Potempa K, Versari L, et al. Temporal coding in spiking neural networks with alpha synaptic function[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 8529-8533.

[58] Park S, Kim S, Choe H, et al. Fast and efficient information transmission with burst spikes in deep spiking neural networks[C]//Proceedings of the 56th Annual Design Automation Conference 2019. 2019: 1-6.

[59] Van Rullen R, Thorpe S J. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex[J]. Neural computation, 2001, 13(6): 1255-1283.

[60] Fairhall A L, Lewen G D, Bialek W, et al. Efficiency and ambiguity in an adaptive neural code[J]. Nature, 2001, 412(6849): 787-792.

[61] Alam S H, Foshie A, Rose G. A Runtime-Reconfigurable Hardware Encoder for Spiking Neural Networks[C]//Proceedings of the Great Lakes Symposium on VLSI 2023. 2023: 203-206.

[62] Hebb D O. The organization of behavior: A neuropsychological theory[M]. Psychology press, 2005.

[63] Caporale N, Dan Y. Spike timing–dependent plasticity: a Hebbian learning rule[J]. Annu. Rev. Neurosci., 2008, 31(1): 25-46.

[64] Song S, Miller K D, Abbott L F. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity[J]. Nature neuroscience, 2000, 3(9): 919-926.

[65] Dan Y, Poo M M. Spike timing-dependent plasticity: from synapse to perception[J]. Physiological reviews, 2006, 86(3): 1033-1048.

[66] Qiao G C, Hu S G, Wang J J, et al. A neuromorphic-hardware oriented bio-plausible online-learning spiking neural network model[J]. IEEE Access, 2019, 7: 71730-71740.

[67] Joo B, Han J W, Kong B S. Energy-and area-efficient CMOS synapse and neuron for spiking neural networks with STDP learning[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(9): 3632-3642.

[68] Liu S, Wang J J, Zhou J T, et al. An area-and energy-efficient spiking neural network with spike-time-dependent plasticity realized with SRAM processing-in-memory macro and on-chip unsupervised learning[J]. IEEE Transactions on Biomedical Circuits and Systems, 2023, 17(1): 92-104.

[69] Qu L, Zhao Z, Wang L, et al. Efficient and hardware-friendly methods to implement competitive learning for spiking neural networks[J]. Neural Computing and Applications, 2020, 32(17): 13479-13490.

[70] Sun C, Sun H, Xu J, et al. An energy efficient STDP-based SNN architecture with on-chip learning[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(12): 5147-5158.

[71] Tavanaei A, Maida A. BP-STDP: Approximating backpropagation using spike timing dependent plasticity[J]. Neurocomputing, 2019, 330: 39-47.

[72] Gomar S, Ahmadi M. Digital realization of PSTDP and TSTDP learning[C]//2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018: 1-5.

[73] Bellec G, Salaj D, Subramoney A, et al. Long short-term memory and learning-to-learn in networks of spiking neurons[J]. Advances in neural information processing systems, 2018, 31.

[74] Neftci E O, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks[J]. IEEE Signal Processing Magazine, 2019, 36(6): 51-63.

[75] Shrestha S B, Orchard G. Slayer: Spike layer error reassignment in time[J]. Advances in neural information processing systems, 2018, 31.

[76] Rathi N, Chakraborty I, Kosta A, et al. Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware[J]. ACM Computing Surveys, 2023, 55(12): 1-49.

[77] Zheng N, Mazumder P. Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity[J]. IEEE transactions on neural networks and learning systems, 2017, 29(9): 4287-4302.

[78] Qiao G C, Ning N, Zuo Y, et al. Batch normalization-free weight-binarized SNN based on hardware-saving IF neuron[J]. Neurocomputing, 2023, 544: 126234.

[79] Zhou P, Choi D U, Kang S M, et al. Backpropagating Errors Through Memristive Spiking Neural Networks[C]//2023 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2023: 1-5.

[80] Sankaran A, Detterer P, Kannan K, et al. An event-driven recurrent spiking neural network architecture for efficient inference on FPGA[C]//Proceedings of the International Conference on Neuromorphic Systems 2022. 2022: 1-8.

[81] Liang L, Chen Z, Deng L, et al. Accelerating spatiotemporal supervised training of large-scale spiking neural networks on gpu[C]//2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022: 658-663.

[82] Bohnstingl T, Woźniak S, Pantazi A, et al. Online spatio-temporal learning in deep neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 34(11): 8894-8908.

[83] Bellec G, Scherr F, Subramoney A, et al. A solution to the learning dilemma for recurrent networks of spiking neurons[J]. Nature communications, 2020, 11(1): 3625.

[84] Ferré P, Mamalet F, Thorpe S J. Unsupervised feature learning with winner-takes-all based stdp[J]. Frontiers in computational neuroscience, 2018, 12: 24.

[85] Cao Y, Chen Y, Khosla D. Spiking deep convolutional neural networks for energy-efficient object recognition[J]. International Journal of Computer Vision, 2015, 113: 54-66.

[86] Xu Y, Tang H, Xing J, et al. Spike trains encoding and threshold rescaling method for deep spiking neural networks[C]//2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017: 1-6.

[87] Xu Q, Peng J, Shen J, et al. Deep CovDenseSNN: A hierarchical event-driven dynamic framework with spiking neurons in noisy environment[J]. Neural Networks, 2020, 121: 512-519.

[88] Wang Y, Xu Y, Yan R, et al. Deep spiking neural networks with binary weights for object recognition[J]. IEEE Transactions on Cognitive and Developmental Systems, 2020, 13(3): 514-523.

[89] Lew D, Lee K, Park J. A time-to-first-spike coding and conversion aware training for energy-efficient deep spiking neural network processor design[C]//Proceedings of the 59th ACM/IEEE Design Automation Conference. 2022: 265-270.

[90] Diehl P U, Neil D, Binas J, et al. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing[C]//2015 International joint conference on neural networks (IJCNN). ieee, 2015: 1-8.

[91] Sengupta A, Ye Y, Wang R, et al. Going deeper in spiking neural networks: VGG and residual architectures[J]. Frontiers in neuroscience, 2019, 13: 95.

[92] Zhang J, Zhang L. Spiking neural network implementation on fpga for multiclass classification[C]//2023 IEEE International Systems Conference (SysCon). IEEE, 2023: 1-8.

[93] Hu Y, Tang H, Pan G. Spiking deep residual networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 34(8): 5200-5205.

[94] Zhang D, Jia S, Wang Q. Recent advances and new frontiers in spiking neural networks[J]. arXiv preprint arXiv:2204.07050, 2022.

[95] Guo W, Yantır H E, Fouda M E, et al. Toward the optimal design and FPGA implementation of spiking neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(8): 3988-4002.

[96] Chen Y, Yu Z, Fang W, et al. Pruning of deep spiking neural networks through gradient rewiring[J]. arXiv preprint arXiv:2105.04916, 2021.

[97] Yin S, Venkataramanaiah S K, Chen G K, et al. Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations[C]//2017 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2017: 1-5.

[98] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE transactions on neural networks, 1994, 5(2): 157-166.

[99] Alemi A, Machens C, Deneve S, et al. Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules[C]//Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).

[100] Gilra A, Gerstner W. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network[J]. Elife, 2017, 6: e28295.

[101] Meulemans A, Tristany Farinha M, García Ordóñez J, et al. Credit assignment in neural networks through deep feedback control[J]. Advances in Neural Information Processing Systems, 2021, 34: 4674-4687.

[102] Urbanczik R, Senn W. Learning by the dendritic prediction of somatic spiking[J]. Neuron, 2014, 81(3): 521-528.

[103] Saleh A Y, Hameed H, Najib M, et al. A novel hybrid algorithm of differential evolution with evolving spiking neural network for pre-synaptic neurons optimization[J]. Int. J. Advance Soft Compu. Appl, 2014, 6(1): 1-16.

[104] Schaffer J D. Evolving spiking neural networks: A novel growth algorithm corrects the teacher[C]//2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). IEEE, 2015: 1-8.

[105] Vazquez R. Izhikevich neuron model and its application in pattern recognition[J]. Australian Journal of Intelligent Information Processing Systems, 2010, 11(1): 35-40.

[106] López-Vázquez G, Ornelas-Rodríguez M, Espinal A, et al. Evolving random topologies of spiking neural networks for pattern recognition[J]. Computer Science and Information Technology, 2019, 9(7): 41-56.

[107] Yusuf Z M, Hamed H N A, Yusuf L M, et al. Evolving spiking neural network (ESNN) and harmony search algorithm (HSA) for parameter optimization[C]//2017 6th international conference on electrical engineering and informatics (ICEEI). IEEE, 2017: 1-6.

[108] Zhang A, Han Y, Niu Y, et al. Self-evolutionary neuron model for fast-response spiking neural networks[J]. IEEE Transactions on Cognitive and Developmental Systems, 2021, 14(4): 1766-1777.

[109] Abadi M, Barham P, Chen J, et al. {TensorFlow}: a system for {Large-Scale} machine learning[C]//12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016: 265-283.

[110] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[111] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. 2014: 675-678.

[112] Hazan H, Saunders D J, Khan H, et al. Bindsnet: A machine learning-oriented spiking neural networks library in python[J]. Frontiers in neuroinformatics, 2018, 12: 89.

[113] Fang W, Chen Y, Ding J, et al. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence[J]. Science Advances, 2023, 9(40): eadi1480.

[114] Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

[115] Mozafari M, Ganjtabesh M, Nowzari-Dalini A, et al. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron[J]. Frontiers in neuroscience, 2019, 13: 625.

[116] Frémaux N, Gerstner W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules[J]. Frontiers in neural circuits, 2016, 9: 85.

[117] Javanshir A, Nguyen T T, Mahmud M A P, et al. Advancements in algorithms and neuromorphic hardware for spiking neural networks[J]. Neural Computation, 2022, 34(6): 1289-1328.

[118] Gewaltig M O, Diesmann M. Nest (neural simulation tool)[J]. Scholarpedia, 2007, 2(4): 1430.

[119] Bekolay T, Bergstra J, Hunsberger E, et al. Nengo: a Python tool for building large-scale functional brain models[J]. Frontiers in neuroinformatics, 2014, 7: 48.

[120] Fidjeland A K, Roesch E B, Shanahan M P, et al. NeMo: a platform for neural modelling of spiking neurons using GPUs[C]//2009 20th IEEE international conference on application-specific systems, architectures and processors. IEEE, 2009: 137-144.

[121] Yavuz E, Turner J, Nowotny T. GeNN: a code generation framework for accelerated brain simulations[J]. Scientific reports, 2016, 6(1): 18854.

[122] Stimberg M, Brette R, Goodman D F M. Brian 2, an intuitive and efficient neural simulator[J]. elife, 2019, 8: e47314.

[123] Niedermeier L, Chen K, Xing J, et al. CARLsim 6: an open source library for large-scale, biologically detailed spiking neural network simulation[C]//2022 International Joint Conference on Neural Networks (IJCNN). IEEE, 2022: 1-10.

[124] Xu X, Ding Y, Hu S X, et al. Scaling for edge inference of deep neural networks[J]. Nature Electronics, 2018, 1(4): 216-222.

[125] Mead C. Neuromorphic electronic systems[J]. Proceedings of the IEEE, 1990, 78(10): 1629-1636.

[126] Dennard R H, Gaensslen F H, Yu H N, et al. Design of ion-implanted MOSFET's with very small physical dimensions[J]. IEEE Journal of solid-state circuits, 1974, 9(5): 256-268.

[127] Li H, Ota K, Dong M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing[J]. IEEE network, 2018, 32(1): 96-101.

[128] Chen D, Singh D. Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms[C]//2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2013: 297-304.

[129] Boutros A, Yazdanshenas S, Betz V. You cannot improve what you do not measure: FPGA vs. ASIC efficiency gaps for convolutional neural network inference[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2018, 11(3): 1-23.

[130] Plagwitz P, Hannig F, Teich J, et al. SNN vs. CNN Implementations on FPGAs: An Empirical Evaluation[C]//International Symposium on Applied Reconfigurable Computing. Cham: Springer Nature Switzerland, 2024: 3-18.

[131] Isik M, Paul A, Varshika M L, et al. A design methodology for fault-tolerant computing using astrocyte neural networks[C]//Proceedings of the 19th ACM international conference on computing frontiers. 2022: 169-172.

[132] Khodamoradi A, Denolf K, Kastner R. S2n2: A fpga accelerator for streaming spiking neural networks[C]//The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2021: 194-205.

[133] Umuroglu Y, Fraser N J, Gambardella G, et al. Finn: A framework for fast, scalable binarized neural network inference[C]//Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays. 2017: 65-74.

[134] Deng B, Fan Y, Wang J, et al. Auditory perception architecture with spiking neural network and implementation on FPGA[J]. Neural Networks, 2023, 165: 31-42.

[135] Cerezuela-Escudero E, Jimenez-Fernandez A, Paz-Vicente R, et al. Sound recognition system using spiking and MLP neural networks[C]//Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25. Springer International Publishing, 2016: 363-371.

[136] Sharifshazileh M, Burelo K, Sarnthein J, et al. An electronic neuromorphic system for real-time detection of high frequency oscillations (HFO) in intracranial EEG[J]. Nature communications, 2021, 12(1): 3095.

[137] Guerra-Hernandez E I, Espinal A, Batres-Mendoza P, et al. A FPGA-based neuromorphic locomotion system for multi-legged robots[J]. IEEE Access, 2017, 5: 8301-8312.

[138] Yousefzadeh A, Orchard G, Stromatias E, et al. Hybrid neural network, an efficient low-power digital hardware implementation of event-based artificial neural network[C]//2018 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2018: 1-5.

[139] Ju X, Fang B, Yan R, et al. An FPGA implementation of deep spiking neural networks for low-power and fast classification[J]. Neural computation, 2020, 32(1): 182-204.

[140] Kuang Z, Wang J, Yang S, et al. Digital implementation of the spiking neural network and its digit recognition[C]//2019 Chinese Control And Decision Conference (CCDC). IEEE, 2019: 3621-3625.

[141] Neil D, Liu S C. Minitaur, an event-driven FPGA-based spiking network accelerator[J]. IEEE transactions on very large scale integration (VLSI) systems, 2014, 22(12): 2621-2628.

[142] Wang Q, Li Y, Shao B, et al. Energy efficient parallel neuromorphic architectures with approximate arithmetic on FPGA[J]. Neurocomputing, 2017, 221: 146-158.

[143] Zhang C M, Qiao G C, Hu S G, et al. A versatile neuromorphic system based on simple neuron model[J]. AIP Advances, 2019, 9(1).

[144] Zhang J, Wu H, Wei J, et al. An asynchronous reconfigurable SNN accelerator with event-driven time step update[C]//2019 IEEE Asian Solid-State Circuits Conference (A-SSCC). IEEE, 2019: 213-216.

[145] Abderrahmane N, Miramond B. Information coding and hardware architecture of spiking neural networks[C]//2019 22nd Euromicro Conference on Digital System Design (DSD). IEEE, 2019: 291-298.

[146] Guo S, Wang L, Wang S, et al. A systolic SNN inference accelerator and its co-optimized software framework[C]//Proceedings of the 2019 Great Lakes Symposium on VLSI. 2019: 63-68.

[147] Losh M, Llamocca D. A low-power spike-like neural network design[J]. Electronics, 2019, 8(12): 1479.

[148] Han J, Li Z, Zheng W, et al. Hardware implementation of spiking neural networks on FPGA[J]. Tsinghua Science and Technology, 2020, 25(4): 479-486.

[149] Fang H, Mei Z, Shrestha A, et al. Encoding, model, and architecture: Systematic optimization for spiking neural network in FPGAs[C]//Proceedings of the 39th International Conference on Computer-Aided Design. 2020: 1-9

[150] Wang S Q, Wang L, Deng Y, et al. SIES: A novel implementation of spiking convolutional neural network inference engine on field-programmable gate array[J]. Journal of Computer Science and Technology, 2020, 35: 475-489.

[151] Aung M T L, Qu C, Yang L, et al. DeepFire: Acceleration of convolutional spiking neural network on modern field programmable gate arrays[C]//2021 31st International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2021: 28-32.

[152] Li S, Zhang Z, Mao R, et al. A fast and energy-efficient SNN processor with adaptive clock/event-driven computation scheme and online learning[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(4): 1543-1552.

[153] Zheng H, Guo Y, Yang X, et al. Balancing the cost and performance trade-offs in SNN processors[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 68(9): 3172-3176.

[154] Gerlinghoff D, Wang Z, Gu X, et al. E3NE: An end-to-end framework for accelerating spiking neural networks with emerging neural encoding on FPGAs[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(11): 3207-3219.

[155] Zhang J, Wang R, Pei X, et al. A fast spiking neural network accelerator based on BP-STDP algorithm and weighted neuron model[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 69(4): 2271-2275.

[156] Liu Y, Chen Y, Ye W, et al. FPGA-NHAP: A general FPGA-based neuromorphic hardware acceleration platform with high speed and low power[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(6): 2553-2566.

[157] Ye W, Chen Y, Liu Y. The implementation and optimization of neuromorphic hardware for supporting spiking neural networks with MLP and CNN topologies[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 42(2): 448-461.

[158] Chen Q, Gao C, Fang X, et al. Skydiver: A spiking neural network accelerator exploiting spatio-temporal workload balance[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(12): 5732-5736.

[159] Sommer J, Özkan M A, Keszocze O, et al. Efficient hardware acceleration of sparsely active convolutional spiking neural networks[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 3767-3778.

[160] Liu H, Chen Y, Zeng Z, et al. A low power and low latency FPGA-based spiking neural network accelerator[C]//2023 International Joint Conference on Neural Networks (IJCNN). IEEE, 2023: 1-8.

[161] Wang Z, Zhong Y, Cui X, et al. A spiking neural network accelerator based on ping-pong architecture with sparse spike and weight[C]//2023 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2023: 1-5.

[162] Li J, Shen G, Zhao D, et al. Firefly: A high-throughput hardware accelerator for spiking neural networks with efficient dsp and memory optimization[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023, 31(8): 1178-1191.

[163] Hofmann J. An Improved Framework for and Case Studies in FPGA-Based Application Acceleration-Computer Vision, In-Network Processing and Spiking Neural Networks[J]. 2020.

[164] Ullah S, Rehman S, Shafique M, et al. High-performance accurate and approximate multipliers for FPGA-based hardware accelerators[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 41(2): 211-224.

[165] Kim Y, Li Y, Park H, et al. Neural architecture search for spiking neural networks[C]//European conference on computer vision. Cham: Springer Nature Switzerland, 2022: 36-56.

[166] Nane R, Sima V M, Pilato C, et al. A survey and evaluation of FPGA high-level synthesis tools[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 35(10): 1591-1604.

[167] Misra J, Saha I. Artificial neural networks in hardware: A survey of two decades of progress[J]. Neurocomputing, 2010, 74(1-3): 239-255.

[168] Neil D, Liu S C. Effective sensor fusion with event-based sensors and deep network architectures[C]//2016 IEEE international symposium on circuits and systems (ISCAS). IEEE, 2016: 2282-2285.

[169] Benjamin B V, Gao P, McQuinn E, et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations[J]. Proceedings of the IEEE, 2014, 102(5): 699-716.

[170] Schemmel J, Brüderle D, Grübl A, et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling[C]//2010 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2010: 1947-1950.

[171] Qiao N, Mostafa H, Corradi F, et al. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses[J]. Frontiers in neuroscience, 2015, 9: 141.

[172] Moradi S, Qiao N, Stefanini F, et al. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)[J]. IEEE transactions on biomedical circuits and systems, 2017, 12(1): 106-122.

[173] Merolla P A, Arthur J V, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface[J]. Science, 2014, 345(6197): 668-673.

[174] Furber S B, Galluppi F, Temple S, et al. The spinnaker project[J]. Proceedings of the IEEE, 2014, 102(5): 652-665.

[175] Davison A P, Brüderle D, Eppler J M, et al. PyNN: a common interface for neuronal network simulators[J]. Frontiers in neuroinformatics, 2009, 2: 388.

[176] Davies M, Srinivasa N, Lin T H, et al. Loihi: A neuromorphic manycore processor with on-chip learning[J]. Ieee Micro, 2018, 38(1): 82-99.

[177] Yang Y S, Kim Y. Recent trend of neuromorphic computing hardware: Intel's neuromorphic system perspective[C]//2020 International SoC Design Conference (ISOCC). IEEE, 2020: 218-219.

[178] Pei J, Deng L, Song S, et al. Towards artificial general intelligence with hybrid Tianjic chip architecture[J]. Nature, 2019, 572(7767): 106-111.

[179] Ma D, Shen J, Gu Z, et al. Darwin: A neuromorphic hardware co-processor based on spiking neural networks[J]. Journal of systems architecture, 2017, 77: 43-51.

[180] Benini L, De Micheli G. Networks on chip: A new paradigm for systems on chip design[C]//Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition. IEEE, 2002: 418-419.

[181] Li Y, Wang Z, Midya R, et al. Review of memristor devices in neuromorphic computing: materials sciences and device challenges[J]. Journal of Physics D: Applied Physics, 2018, 51(50): 503002.

[182] Gaines B R. Stochastic computing[C]//Proceedings of the April 18-20, 1967, spring joint computer conference. 1967: 149-156.

[183] Alawad M, Yoon H J, Tourassi G. Energy efficient stochastic-based deep spiking neural networks for sparse datasets[C]//2017 IEEE International Conference on Big Data (Big Data). IEEE, 2017: 311-318.

[184] Chakraborty I, Jaiswal A, Saha A K, et al. Pathways to efficient neuromorphic computing with non-volatile memory technologies[J]. Applied Physics Reviews, 2020, 7(2).

[185] Chua L. Memristor-the missing circuit element[J]. IEEE Transactions on circuit theory, 1971, 18(5): 507-519.

[186] Strukov D B, Snider G S, Stewart D R, et al. The missing memristor found[J]. nature, 2008, 453(7191): 80-83.

[187] Chua L O, Kang S M. Memristive devices and systems[J]. Proceedings of the IEEE, 1976, 64(2): 209-223.

[188] Zamarreño-Ramos C, Camuñas-Mesa L A, Pérez-Carrasco J A, et al. On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex[J]. Frontiers in neuroscience, 2011, 5: 26.

[189] Emelyanov A V, Nikiruy K E, Demin V A, et al. Yttria-stabilized zirconia cross-point memristive devices for neuromorphic applications[J]. Microelectronic Engineering, 2019, 215: 110988.

[190] Demin V A, Nekhaev D V, Surazhevsky I A, et al. Necessary conditions for STDP-based pattern recognition learning in a memristive spiking neural network[J]. Neural Networks, 2021, 134: 64-75.

[191] Burr G W, Shelby R M, Sebastian A, et al. Neuromorphic computing using non-volatile memory[J]. Advances in Physics: X, 2017, 2(1): 89-124.

[192] Midya R, Wang Z, Asapu S, et al. Artificial neural network (ANN) to spiking neural network (SNN) converters based on diffusive memristors[J]. Advanced Electronic Materials, 2019, 5(9): 1900060.

[193] Wang Z, Joshi S, Savel'Ev S, et al. Fully memristive neural networks for pattern classification with unsupervised learning[J]. Nature Electronics, 2018, 1(2): 137-145.

[194] Jo S H, Chang T, Ebong I, et al. Nanoscale memristor device as synapse in neuromorphic systems[J]. Nano letters, 2010, 10(4): 1297-1301.

[195] Chen L, Xiong X, Liu J. A survey of intelligent chip design research based on spiking neural networks[J]. IEEE Access, 2022, 10: 89663-89686.

[196] Yu S. Resistive random access memory (RRAM)[M]. Morgan & Claypool Publishers, 2016.

[197] Shukla A, Prasad S, Lashkare S, et al. A case for multiple and parallel RRAMs as synaptic model for training SNNs[C]//2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018: 1-8.

[198] El Arrassi A, Gebregiorgis A, El Haddadi A, et al. Energy-efficient SNN implementation using RRAM-based computation in-memory (CIM)[C]//2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC). IEEE, 2022: 1-6.

[199] Wong H S P, Raoux S, Kim S B, et al. Phase change memory[J]. Proceedings of the IEEE, 2010, 98(12): 2201-2227.

[200] Bohnstingl T, Šurina A, Fabre M, et al. Biologically-inspired training of spiking recurrent neural networks with neuromorphic hardware[C]//2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2022: 218-221.

[201] Titirsha T, Song S, Das A, et al. Endurance-aware mapping of spiking neural networks to neuromorphic hardware[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(2): 288-301.

[202] Bader S D, Parkin S S P. Spintronics[J]. Annu. Rev. Condens. Matter Phys., 2010, 1(1): 71-88.

[203] Diao Z, Li Z, Wang S, et al. Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory[J]. Journal of Physics: Condensed Matter, 2007, 19(16): 165209.

[204] Vincent A F, Larroque J, Locatelli N, et al. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems[J]. IEEE transactions on biomedical circuits and systems, 2015, 9(2): 166-174.

[205] Fong X, Kim Y, Yogendra K, et al. Spin-transfer torque devices for logic and memory: Prospects and perspectives[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 35(1): 1-22.

[206] Sengupta A, Parsa M, Han B, et al. Probabilistic deep spiking neural systems enabled by magnetic tunnel junction[J]. IEEE Transactions on Electron Devices, 2016, 63(7): 2963-2970.

[207] Kulkarni S R, Kadetotad D V, Yin S, et al. Neuromorphic hardware accelerator for SNN inference based on STT-RAM crossbar arrays[C]//2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2019: 438-441.

[208] Brette R, Rudolph M, Carnevale T, et al. Simulation of networks of spiking neurons: a review of tools and strategies[J]. Journal of computational neuroscience, 2007, 23: 349-398.

[209] Thomas D, Luk W. FPGA accelerated simulation of biologically plausible spiking neural networks[C]//2009 17th IEEE symposium on field programmable custom computing machines. IEEE, 2009: 45-52.

[210] Garey M R, Johnson D S, Stockmeyer L. Some simplified NP-complete problems[C]//Proceedings of the sixth annual ACM symposium on Theory of computing. 1974: 47-63.

[211] Balaji A, Das A, Wu Y, et al. Mapping spiking neural networks to neuromorphic hardware[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 28(1): 76-86.

[212] Zhang L, Li S, Qu L, et al. MAMAP: Congestion relieved memetic algorithm based mapping method for mapping large-scale SNNs onto NoC-based neuromorphic hardware[C]//2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2020: 640-647.

[213] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proceedings of ICNN'95-international conference on neural networks. ieee, 1995, 4: 1942-1948.

[214] Glover F, Laguna M. Tabu search[M]. Springer US, 1998.

[215] Li S, Guo S, Zhang L, et al. SNEAP: A fast and efficient toolchain for mapping large-scale spiking neural network onto NoC-based neuromorphic platform[C]//Proceedings of the 2020 on Great Lakes Symposium on VLSI. 2020: 9-14.

[216] Xiao C, Chen J, Wang L. Optimal mapping of spiking neural network to neuromorphic hardware for edge-AI[J]. Sensors, 2022, 22(19): 7248.

[217] Galluppi F, Davies S, Rast A, et al. A hierachical configuration system for a massively parallel neural hardware platform[C]//Proceedings of the 9th conference on Computing Frontiers. 2012: 183-192.

[218] Esser S K, Merolla P A, Arthur J V, et al. From the cover: Convolutional networks for fast, energy-efficient neuromorphic computing[J]. Proceedings of the National Academy of Sciences of the United States of America, 2016, 113(41): 11441.

[219] Deng L, Wang G, Li G, et al. Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation[J]. IEEE Journal of Solid-State Circuits, 2020, 55(8): 2228-2246.

[220] Rueckauer B, Bybee C, Goettsche R, et al. NxTF: An API and compiler for deep spiking neural networks on Intel Loihi[J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2022, 18(3): 1-22.

[221] Zou C, Cui X, Chen G, et al. Modular building blocks for mapping spiking neural networks onto a programmable neuromorphic processor[J]. Microelectronics Journal, 2022, 129: 105612.

[222] Song S, Chong H, Balaji A, et al. DFSynthesizer: Dataflow-based synthesis of spiking neural networks to neuromorphic hardware[J]. ACM Transactions on Embedded Computing Systems (TECS), 2022, 21(3): 1-35.

[223] Gerlinghoff D, Wang Z, Gu X, et al. E3NE: An end-to-end framework for accelerating spiking neural networks with emerging neural encoding on FPGAs[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(11): 3207-3219.

[224] Rahiminejad E, Azad F, Parvizi-Fard A, et al. A neuromorphic CMOS circuit with self-repairing capability[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(5): 2246-2258.

[225] Timcheck J, Kadmon J, Boahen K, et al. Optimal noise level for coding with tightly balanced networks of spiking neurons in the presence of transmission delays[J]. PLoS computational biology, 2022, 18(10): e1010593.

[226] Srinivasan S, Stevens C F. Robustness and fault tolerance make brains harder to study[J]. BMC biology, 2011, 9: 1-3.

[227] Maass W. Noise as a resource for computation and learning in networks of spiking neurons[J]. Proceedings of the IEEE, 2014, 102(5): 860-880.

[228] Torres-Huitzil C, Girau B. Fault and error tolerance in neural networks: A review[J]. IEEE Access, 2017, 5: 17322-17341.

[229] Shukla M, Kumar A, Mahajan P. Reliable Fault Tolerance and Recovery for VLSI Systems[C]//2024 International Conference on Optimization Computing and Wireless Communication (ICOCWC). IEEE, 2024: 1-6.

[230] Mahdiani H R, Fakhraie S M, Lucas C. Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors[J]. IEEE transactions on neural networks and learning systems, 2012, 23(8): 1215-1228.

[231] Schuman C D, Potok T E, Patton R M, et al. A survey of neuromorphic computing and neural networks in hardware[J]. arXiv preprint arXiv:1705.06963, 2017.

[232] Alippi C. Selecting accurate, robust, and minimal feedforward neural networks[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2002, 49(12): 1799-1810.

[233] Nelson V P. Fault-tolerant computing: Fundamental concepts[J]. Computer, 1990, 23(7): 19-25.

[234] Vatajelu E I, Prinetto P, Taouil M, et al. Challenges and solutions in emerging memory testing[J]. IEEE Transactions on Emerging Topics in Computing, 2017, 7(3): 493-506.

[235] Xia L, Gu P, Li B, et al. Technological exploration of RRAM crossbar array for matrix-vector multiplication[J]. Journal of Computer Science and Technology, 2016, 31(1): 3-19.

[236] Pop P, Izosimov V, Eles P, et al. Design optimization of time-and cost-constrained fault-tolerant embedded systems with checkpointing and replication[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2009, 17(3): 389-402.

[237] Neti C, Schneider M H, Young E D. Maximally fault tolerant neural networks[J]. IEEE Transactions on Neural Networks, 1992, 3(1): 14-23.

[238] Protzel P W, Palumbo D L, Arras M K. Performance and fault-tolerance of neural networks for optimization[J]. IEEE transactions on Neural Networks, 1993, 4(4): 600-614.

[239] Indiveri G. A low-power adaptive integrate-and-fire neuron circuit[C]//Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03. IEEE, 2003, 4: IV-IV.

[240] Spyrou T, El-Sayed S A, Afacan E, et al. Reliability analysis of a spiking neural network hardware accelerator[C]//2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022: 370-375.

[241] Vatajelu E I, Di Natale G, Anghel L. Special session: Reliability of hardware-implemented spiking neural networks (SNN)[C]//2019 IEEE 37th VLSI Test Symposium (VTS). IEEE, 2019: 1-8.

[242] Goodman D F M, Brette R. Brian: a simulator for spiking neural networks in python[J]. Frontiers in neuroinformatics, 2008, 2: 350.

[243] Schuman C D, Mitchell J P, Johnston J T, et al. Resilience and robustness of spiking neural networks for neuromorphic systems[C]//2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020: 1-10.

[244] Spyrou T, El-Sayed S A, Afacan E, et al. Neuron fault tolerance in spiking neural networks[C]//2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 743-748.

[245] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.

[246] Chen Q, He G, Wang X, et al. A 67.5 μJ/prediction accelerator for spiking neural networks in image segmentation[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 69(2): 574-578.

[247] Zhang J, Liang M, Wei J, et al. A 28nm configurable asynchronous SNN accelerator with energy-efficient learning[C]//2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). IEEE, 2021: 34-39.

[248] Putra R V W, Hanif M A, Shafique M. SoftSNN: Low-cost fault tolerance for spiking neural network accelerators under soft errors[C]//Proceedings of the 59th ACM/IEEE Design Automation Conference. 2022: 151-156.

[249] Yerima W Y, Ikechukwu O M, Dang K N, et al. Fault-tolerant spiking neural network mapping algorithm and architecture to 3D-NoC-based neuromorphic systems[J]. IEEE Access, 2023, 11: 52429-52443.

[250] Liu J, Harkin J, Maguire L P, et al. SPANNER: A self-repairing spiking neural network hardware architecture[J]. IEEE transactions on neural networks and learning systems, 2017, 29(4): 1287-1300.

[251] Johnson A P, Liu J, Millard A G, et al. Time-multiplexed system-on-chip using fault-tolerant astrocyte-neuron networks[C]//2018 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2018: 1076-1083.

**俞诗航**(2000－),男,硕士生,主要研究领域为边缘智能,脉冲神经网络.

**吴洲**(2001－),男,硕士生,主要研究领域为脉冲神经网络.

**赵健**(1979－),男,博士,副教授,IEEE 高级会员,主要研究领域为通信网络,神经计算.

**易梦军**(1997－),男,博士生,主要研究领域为边缘智能,联邦学习.

**申富饶**(1973－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为神经计算,机器人智能.