摘 要

脉冲神经网络（SNN，Spiking Neural Networks）是第三代神经网络，其工作机理与生物的神经系统更接近，其模仿神经元的脉冲机制只在新的输入脉冲尖峰到达时才计算神经元，因此能在硬件设备上有效地实现节能模式，而且 SNN 层内连接与反向连接的复杂拓扑结构具有解决复杂问题和实时应用的潜力。但类脑计算中网络地仿真都伴随着大量串行的基于复杂浮点计算的神经元和突触更新，因此类脑计算地仿真普遍存在的主要问题是较低的仿真计算效率、较高的能量损耗。

针对这个问题，本文首先对神经元更新方程的不同数值求解方法进行量化实验研究，并通过分析找到最佳的求解方法，并使用位宽压缩方法对 SNN 网络的参数进行定制化混合精度地设计，然后在此基础之上设计一种基于 FPGA 的定制计算方法，最后把该方法应用于 NEST 框架下实现一个脉冲神经网络仿真加速器系统，具体的研究内容如下：

第一，完成对 SNN 的量化研究分析。对 SNN 的量化研究主要分三部分，第一部分是 SNN 网络模型的轻量化研究，首先是回顾卷积神经网络（CNN，Convolutional Neural Network）常见的优化网络拓扑结构和位宽压缩的方法，并基于此提出针对仿生性 SNN 的混合精度量化方法。第二部分是通过对类脑仿真器 NEST 的负载进行量化分析，分析出 SNN 仿真的计算密集点。最后一部分是通过比较不同神经元更新方程的不同数值求解方法的量化实验结果，选择使用计算相对简单且具有生物表现多样的 Izhikevich 神经元作为后续的主要研究对象。

第二，完成基于 FPGA 的定制计算加速方法设计。在选定 Izhikevich 作为量化的研究对象后，对经过前向欧拉法求解的膜电位更新方程的参数进行混合精度量化以减少内存资源的开销和提高计算效率。然后基于 FPGA 平台，完成对 Izhikevich 神经元的可定制流式架构设计，采用平衡数据路径的手段优化单个神经元的膜电位更新，并利用数据重排、流水并行的技术加速整体神经元的膜电位更新。最后通过软件仿真实验比较该定制计算方法在不同并行度、不同 FPGA 平台上的表现以确定最佳的设计方案。

第三，实现基于 NEST 开源框架的硬件加速仿真器系统。完成通用软硬件数据高速交互接口的设计，并通过优化内存分配、降低数据传输总量、减少通信启动次数等方法优化 ARM 和 FPGA 的数据传输。最后实验结果表明，相比于 i7-10700 CPU，经典的丘脑外侧膝状核网络模型和液体状态机模型在 ZCU102 上神经元更新部分的计算性能提升 2.26 和 3.02 倍，更新部分的能效比提升 8.06 和 10.8 倍。

关键词：类脑计算；混合精度；定制化；Izhikevich；FPGA

Abstract

Spiking Neural Networks (SNN) is the third generation of neural networks. Its working mechanism is closer to the biological nervous system. It imitates the pulse mechanism of neurons and only calculates neurons when new input pulse spikes arrive. Therefore, It can effectively implement energy-saving mode on hardware devices, and the complex topology of connections and reverse connections in the SNN layer has the potential to solve complex problems and real-time applications. However, network simulation in brain-inspired computing is accompanied by a large number of serial updates of neurons and synapses based on complex floating-point calculations. Therefore, the simulation of brain-inspired computing generally has the problems of slow processing speed and high power consumption.

Regarding this issue, this article first conducts a quantitative experimental study on different numerical solution methods for the neuron update equation, finds the best solution method through analysis, and uses bit-width compression to customize the parameter of the SNN network. Then, on this basis, a custom computing method based on FPGA is designed, and finally this method is applied to an accelerator system for simulating a pulse neural network under the NEST framework. The specific research content is as follows:

First, complete the quantitative research and analysis of SNN. The quantitative research on SNN is mainly divided into three parts. The first part is the lightweight research of the SNN network model. First, it reviews the common optimization network topology and bit width compression methods of convolutional neural network (CNN), and based on This paper proposes a mixed-precision quantization method for bioinspired SNNs. The second part is to analyze the calculation-intensive points of SNN simulation by quantitatively analyzing the load of the brain-inspired simulator NEST. The last part is to compare the quantitative experimental results of different numerical solution methods of different neuron update equations, and choose Izhikevich neurons, which are relatively simple in calculation and have diverse biological performance, as the main research object in the follow-up.

Second, complete the FPGA-based custom computing acceleration method design. After Izhikevich was selected as the quantified research object, the parameters of the membrane potential update equation solved by the forward Euler method were quantified with mixed precision to reduce memory resource overhead and improve computational efficiency. Then, based on the FPGA platform, complete the design of the customizable flow structure of Izhikevich neurons, optimize the membrane potential update of a single neuron by means of balanced data paths, and accelerate the membrane potential update of the overall neuron by using data rearrangement and parallel pipeline technology . Finally, through software simulation experiments, the performance of the customized calculation method on different parallelism and different FPGA platforms is compared to determine the best design scheme.

Third, implement a hardware acceleration emulator system based on the NEST open source framework. Complete the design of the high-speed interactive interface for general software and hardware data, and optimize the data transmission between ARM and FPGA by optimizing memory allocation, reducing the total amount of data transmission, and reducing the number of communication startups. The final experimental results show that compared with the i7-10700 CPU, the classic lateral geniculate nucleus network model and the liquid

state machine model on the ZCU102 have improved computing performance by 2.26 and 3.02 times in the neuron update part, and the energy efficiency ratio of the update part has increased by 8.06 and 10.8 times.

Keywords: Brain-like Computing; Mixed-precision; Customized; Izhikevich; FPGA

目　录

第一章 绪论

1.1 课题研究背景与意义

近年来，人工神经网络（ANN，Artificial Neural Networks）以过去认为不可能的方式改变了世界。从理论上讲，人工神经网络主要是使用理想化的计算单元来构建的，这些计算单元由于其生物学灵感而通常被称为“神经元”。根据计算单元的特点，神经网络可以分为三代。第一代由阈值单元（例如感知器）构成，神经元的输出是二进制（0，1），并且这是通过对加权突触输入进行简单阈值处理而获得的[1]。下一代是通过在计算单元输出处应用连续激活函数来构造的。尽管前两代已被证明在各种应用中有非常成功的表现[2-4]，但仍暴露出许多缺点，例如，连续的数值被用作神经元处理和传输的信息、密集的矩阵乘法与存储器访问等。这种机制与架构需要大量的计算资源和极高的系统配置，因此其效率低且不容易在硬件上尤其是便携式设备上实现。这些缺点导致了第三代网络的出现，称为脉冲神经网络。

类脑计算以脉冲神经网络为基础，与深度神经网络（DNN，Deep Neural Networks）等第二代神经网络相比，SNN 的工作机理与生物的神经系统非常相似，在大脑中，神经元通过发送动作电位序列（也称为棘波序列）相互交流，SNN 模仿这种机制，只在新的输入脉冲尖峰到达时才计算神经元，因此，网络能有效地处于适合于在硬件设备上实现的节能模式，并在实时应用中显示出巨大的潜力[5-7]，因此在大量的任务中广泛应用，例如图像和孤立词识别[8,9]、目标检测[10]、导航[11]和运动控制[12]，也被成为第三代神经网络。SNN 的许多特性是研究者感兴趣的，因此出现了一个研究分支，其试图将 SNN 分类为不同的类型，即神经元模型、数据编码器、学习算法和网络架构，在这些研究的基础上，多位研究者对 SNN 研究进行了不同方式的总结和评述[13-16]。

目前 SNN 主要以软件为主的方式实现神经元并完成网络的搭建与仿真模拟，它具有极高的精度，极强的灵活性等特点，基于 SNN 推出的类脑仿真器如 NEST[17]，NEURON[18]，Brain2[19]等已得到了大量的类脑计算模型在其框架下的应用，并且在此基础之上形成了各自的生态圈。类脑计算中网络的仿真都伴随着大量串行的神经元更新与突触更新，无法充分利用神经网络中高并行度的计算特点，因此基于软件实现的 SNN 都会存在的问题是仿真计算效率低、能耗高。当研究大规模网络时，通常希望尽可能快地模拟它们，然而即使是当今最快的超级计算机也无法完成这项网络仿真任务[20]，因此神经形态计算和特定应用的新颖硬件架构非常有吸引力。尽管专用类脑芯片的计算速度有了显著提升，但系统的效率不高、开发成本过高、灵活性远不如传统的通用处理器。

现场可编程逻辑门阵列（FPGA，Field-Programmable Gate Array）作为一种特殊的数字电路集成方式，具有灵活的可编程性、丰富的计算资源、极短的开发周期等优点。从理论上讲，FPGA 作为一种可编程器件，允许开发自定义逻辑，这可以放松对神经网络实现的限制[21]。它不仅提供了大量的计算资源，而且比专用集成电路（ASIC，Application-Specific Integrated Circuit）的开发时间更短。由于其高度可重构性和并行速

度，FPGA 被认为是在一定程度上模拟生物神经网络自然可塑性的合适平台。在过去，人们做了大量的实验来在 FPGA 上部署第一代和第二代网络，其中许多实验在计算效率和功耗方面显示出令人印象深刻的结果，由于脉冲神经网络在这些方面具有明显的优势，很多研究人员逐渐关注基于 FPGA 平台的 SNN 加速器的设计与实现。近年来，随着半导体技术的不断进步，可编程器件技术和工具的功能有了很大的提高，FPGA 提供了大量的芯片资源（例如，逻辑单元和存储器），允许以负担得起的成本实现复杂的硬件设计。高级综合(HLS，High-level synthesis)工具允许开发人员根据算法描述生成硬件实现，从而减少开发时间，并使非硬件专家能够访问该技术。尽管设计工作量仍然很大，但可编程器件技术在灵活性和效率之间提供了一个很好的折衷方案，因此被广泛认为是非常适合神经网络仿真的潜在技术近年来发展起来的许多数字神经形态结构都利用了这一点。

综上所述，类脑计算的特性表明它能以更节能的方式完成一些实时应用任务，例如目标检测、跟踪、导航和运动控制，尽管脉冲神经网络在通用 CPU 上能表现出高度的灵活性，但当进行大规模的仿真时会存在低效率、高功耗的问题，因此同时具有高性能和灵活性的设计是一项挑战。

1.2 国内外研究现状

在神经网络的性能方面，主要的研究点有：算法优化、网络拓扑结构优化、网络的量化、通信优化、硬件平台加速等。第二代人工神经网络（CNN，Convolutional Neural Network）的研究较为成熟，所以上述各研究点都做了较多的研究工作，绝大多数是前几种软件上的优化策略结合最后的硬件优化实现。算法优化主要是从降低算法复杂度，减少计算的角度优化网络，如 Winograd[22]提出了一种针对卷积的运算加速算法，通过减少滤波器的计算量来实现快速卷积。还有人是从神经网络最优的拓扑结构角度出发，要搜索到一种最优神经结构，就避免不了遍历，因此此法也是一种剪枝的实现，如 Dong[23]等人提出了一种新的神经结构搜索方法，该法通过一个可梯度下降学习的采样器避免了遍历所有子图，搜索到一个最优的神经架构，该方法使用图形处理器（GPU，Graphics Processing Unit）平台进行加速。关于 CNN 网络的量化，最为典型的就是二值化神经网络[24](BNN, Binarized Neural Network)，该网络在 DNN 的基础上约束训练网络权重和神经元值为+1 或者-1（即二值化），因减少了对内存、计算的需求而显著提高了算法的执行效率，但这会引起相应的精度损失。更进一步则可以对每一个量化层选择一个最佳比特位宽，正如 Wei 等人[25]采用了一种基于神经结构搜索的量化位宽搜索来获得一个混合精度的量化网络，能在不损失精度下最小化模型的大小，该方法同时使用 FPGA 平台进行硬件加速。也有人研究比较了各硬件平台加速情况的好坏，如 Nurvitadhi[26]等人比较了相同的 BNN 模型在 CPU、GPU、ASIC 和 FPGA 上的不同表现，最后结果表明 FPGA 的效率要优于 CPU 和 GPU 的，这是因为尽管后两者的计算峰值理论性能很高，但 BNN 等模型并不能充分利用这些平台的计算性能，而且 FPGA 不必锁定在固定的 ASIC 解决方案当中，具有更高的灵活性。

针对 SNN 性能方面，主要是围绕着一种新型的类脑体系架构展开的，当然这种架构也是多方面性能研究的一种综合体现，比如通信、负载均衡、计算加速等。SNN 不同于第二代神经网络，脉冲神经网络通常是具有时间信息的、大规模的，单台计算机无法满足仿真的计算需求，所以往往会采用一种分布式的方式实现实时仿真模拟，但仿真速度仍然难以满足神经科学者的研究所需，所以一种新的类脑体系架构仍然是国内外研究者的研究热点。早期的研究如 Steve 等人在 2014 年就提出了一个尖峰神经网络架构(SpiNNaker)[27]，旨在提供一台大规模并行的百万核计算机，适用于生物实时大规模尖峰网络的建模。最近的研究如 2020 年浙江大学联合之江实验室共同研制出一种用于类脑仿真的计算机，其内包含 792 颗“达尔文 2”的芯片，每个芯片由 576 个计算内核组成，这些芯片能支持 1.2 亿脉冲神经元和近千亿的神经突触[28]，与小鼠大脑的神经元数目相当。有利用超级计算机[29]通过确保随机访问发生在高速缓存中进行加速仿真，尽管都能解决通用 CPU 平台的低性能问题，但会造成更高的能耗。郁[30]等人首先分析类脑模型工作负载的特性并对其进行计算建模，然后完成与计算节点数的映射关系，能为 SNN 工作负载在 FPGA 集群上以较快的速度确定合理的计算节点数。

随着对并行和分布式计算模型研究的深入，发现当计算节点数量不断增加时，模拟用于计有效计算的时间比例降低，此时通信会占主体，因而如何优化通信也成为一个研究热点，如 Fernandez-Musoles 等人[31]把通信建模成一个超图网络，然后提出了一种基于超图的负载分配和动态稀疏通信策略，计算效率提升了 40%，仿真时间减少了约 70%。脉冲神经网络的基本计算单元是神经元和突触，因此也有大量学者针对神经元、突触进行优化计算。在 2021 年，Peng[32]等人在 FPGA 平台上针对 Izhikevich 提出了一种快速收敛的旋转坐标数字计算方法（CORDIC，Coordinate Rotation DIgital Computer），该方法消除了原先算法的冗余迭代和相关计算，优化了神经元膜电位的更新方程。Sun 等人于 2022 年针对 STDP 突触的权重更新提出了一种简化的线性学习规则[33]，在正向转递中引入了向前传递的停止机制能减少峰值的不必要计算，并使用硬件友好的输入量化方案来减少编码阶段和向前传递中的计算复杂性。Guo 等[34]人基于欧拉和三阶龙格库塔（RK，Runge-Kutta）两种数值方法探讨了生物时间常数对 SNN 脉冲发射过程中对于信号传递的影响，并通过一种基于参数的优化方案选来自动的选择适当的时间常数，提高了 SNN 网络在数字识别等应用中损失函数的收敛速度。

1.3 本文主要工作

本文得到国家自然科学基金的支持，设计并实现了一个基于 FPGA 的脉冲神经网络仿真加速器系统。首先是通过仿真性能的对比，选择 NEST 作为主要的研究对象，并对 SNN 进行量化的研究分析，主要分三点：第一点是对 SNN 数据位宽压缩的分析与研究；第二点是对 NEST 的内存、计算和通信负载进行量化分析与建模；第三点是对不同神经元方程的数值求解方法进行量化与分析，选择生物脉冲表现多样性且计算相对简单的 Izhikevich 作为主要的加速研究内容。然后设计基于 Izhikevich 神经元与 FPGA 平台的 SNN 定制计算方法，并对设计方案进行仿真数值实验分析。最后是基于经典的 NEST

软件仿真器，把定制计算方法用于 NEST 的仿真加速，并实现一个规模可拓展的 SNN 仿真加速器系统。本文的主要研究工作如下：

(1) 完成对 SNN 的量化研究分析

主要是对 SNN 的三部分进行量化研究与分析，第一部分是对 SNN 的网络量化方法的分析与研究，并提出了一种针对仿生型 SNN 的定制位宽压缩方法。第二部分是对 SNN 软件仿真器 NEST 的内存、计算和通信负载进行量化分析与建模。第三部分是针对神经元方程的不同数值求解方法进行研究量化分析，并在三种具有代表性的 Leaky-Integrate-and-Fire、Hodgkin-Huxley、Izhikevich 神经元进行详细设计与实验分析，最后选择生物脉冲表现多样性且计算相对简单的 Izhikevich 作为后续主要的加速研究对象。这些量化研究分析为后续基于 FPGA 的定制计算方法的设计以及最后基于 NEST 的硬件加速仿真器系统的实现提供了方向和理论基础依据。

(2) 完成基于 FPGA 的 SNN 定制计算方法的设计

对 Izhikevich 以及 SNN 仿真网络进行进一步计算特性的分析，明确 Izhikevich 神经元的更新方程采用的数值求解方法以及 SNN 网络中的计算密集点。根据前文提出的位宽压缩方法，对 Izhikevich 神经元的参数进行混合精度的定制化设计，量化以后不仅能减少硬件资源的开销还能提高计算的效率。基于 FPGA 平台，完成对 Izhikevich 神经元可定制的流式架构设计，并通过数据重排、平衡数据路径以及流水并行的手段来加速神经元的膜电位更新。最后在仿真实验阶段，统计该定制计算方法在不同并行度下、不同 FPGA 平台的时间开销、资源开销表现，确定在不同平台上的最佳并行度，为后续基于 NEST 的硬件加速仿真器的实现提供加速方案的设计支持。

(3) 实现基于 NEST 框架的硬件加速仿真器系统

把之前提出的定制计算方法应用于 NEST 内，提出一个软硬件仿真加速器的整体系统架构，并在 SD 卡上划分一定的内存分区特定的用于 ARM 与 FPGA 的 PL 部分的信息交互，设计一套较为通用的软硬件通用数据传输方法用于完成 SNN 的仿真模拟，并在此基础之上优化数据交互方式，减少数据通信时间。最后把基于该定制计算方法设计的神经元更新 IP 核集成到电路中，完成整体的电路连接设计与比特流的生成，并在 PYNQ-Z2 平台上实现两种经典的类脑仿生网络的仿真模拟，并验证与原始 ARM 核版本、intel CPU 版本神经元更新部分的时间与能效比的提升效果。为了检验方案的拓展性，在同是 PYNQ 框架下的 ZCU102 平台上也实现了仿真加速器系统。

1.4 论文结构

本文将分六个章节介绍研究背景、内容以及相关的设计与实验研究，并在下面对每一章的主要内容进行一个简要的分析：

第一章：绪论。与传统深度学习相比，本章首先介绍类脑计算的优势所在，并有望解决传统深度学习高功耗的问题，以及目前类脑计算研究中软件或硬件实现方法中各自存在的优点与缺点。随后本文介绍目前国内外对于人工神经网络的性能研究，以及在 CNN 与 SNN 领域的研究工作。最后介绍本文的主要工作和论文的整体结构。

第二章：脉冲神经网络与类脑计算基础。本章节先介绍类脑计算以及脉冲神经网络的基本概念知识，并对脉冲神经网络内具有代表性的、经典的 LIF、Izhikevich、HH 神经元模型以及静态、STDP 突触等模型的基础知识进行详细分析。然后对类脑计算的发展进行梳理与总结工作，并对其中一类 SNN 仿真器的几种类型进行介绍、比较与实验数据分析，选出表现较好的 NEST 软件仿真器作为后续的主要研究重点，并在最后介绍了两种较为经典的类脑计算模型，将用于在最后实验阶段的结果比较。

第三章：脉冲神经网络量化方法研究。本章将对 SNN 的三部分内容进行量化分析研究。首先是对 SNN 网络的量化研究，通过 CNN 等深度学习领域常用的量化方法以及 SNN 的量化研究的回顾与总结，提出了一种针对仿生性 SNN 的混合精度位宽压缩量化方法。其次是对 NEST 仿真器工作负载的量化研究，然后通过量化与建模以分析出其性能瓶颈和计算密集点。最后是针对神经元更新方程不同数值求解方法的量化研究与分析，以分析不同神经元在不同求解方法下的计算特性，通过研究确定了计算相对较少且具有生物多样性的 Izhikevich 神经元作为后续的主要研究对象，并采用前项欧拉法进行数值求解。这一章为后文的设计研究与实验平台的搭建提供了理论支撑。

第四章：基于 FPGA 的 SNN 定制计算方法研究。前一章确定了研究的神经元以及相应的数值求解方法，那就利用所提出的位宽压缩方法针对 Izhikevich 的参数进行混合精度的量化设计。然后基于 FPGA，完成对 Izhikevich 可定制的流式架构设计，并通过平衡数据路径的方法针对单神经元的更新进行优化，利用数据重排以及流水并行的手段优化整体 SNN 神经元的更新。最后通过仿真实验比较不同并行度、FPGA 平台下，该方案的资源利用率、时间开销的表现，以确定最佳的并行度。本章定制计算方法的设计为第五章仿真器实验平台的搭建提供了方案支持。

第五章：基于 NEST 的硬件加速仿真器系统。把定制计算方法应用于 NEST 框架下，提出一个软硬件结合的仿真加速器系统的整体架构。首先需要划分出一块内存，专门用于 ARM 和 FPGA 的 PL 部分进行信息交互，在此基础之上设计一套较为通用的数据传输接口，并优化 NEST 仿真的数据交互，减少通信的次数以减少仿真时间。然后把基于该定制计算方法设计生成的神经元更新 IP 核集成到整体电路中，并完成相应接口的连线与比特流的生成。最后在不同平台上进行两种经典类脑计算的仿真模拟实验，实验结果显示，相比较 i7-10700，LGN 和 LSM 模型在 ZCU102 上神经元更新部分的性能提升 2.26 和 3.02 倍，能效比提升 8.06 和 10.8 倍。

第六章：总结与展望。本章对全文的主要工作进行了整体的概括与总结，在此基础之上提出了目前设计还存在的问题与不足，并为这些问题与不足提供了一些可行的思路方案以及对未来更好的一种设计思路的展望。

第二章 脉冲神经网络与类脑计算基础

2.1 类脑计算

长期以来，大自然一直是人类技术进步的灵感和创造力的源泉，并为许多工程挑战创造了巧妙的解决方案，从鸟类的飞行，到蚁群使用的搜索策略，到为植物提供燃料的光合作用过程，以及无数其他方面。但是，大自然最有趣的产物也许是被称之为智能的解决方案:人类大脑。人类大脑是比较人工智能（AI，Artificial Intelligence）系统能力的黄金标准，尤其是那些基于机器学习（ML，Machine Learning）的系统。因此，一些最先进的 ML 算法，即神经网络，也会受到大脑复杂网络(约 860 亿神经元)的启发。神经网络在过去十年中已经取得了显著的成就，原因主要是（1）用于训练大型神经网络模型的可用数据量增加了;（2）计算能力增加了;（3）神经网络结构和训练算法的理论和设计的关键发展。然而这些改进是以模型的大小和复杂性为代价的，来自文献[35]的数据表明准确率的小提升是以指数级大模型为代价的，此外，在传统的 CPU/GPU 硬件上实现这些最先进的模型会带来巨大的开销，这与具有严格的大小、重量和功率限制的应用域不兼容，例如，移动设备上的人工智能、卫星、传感器，以及一般的边缘人工智能。在这些领域实现人工智能是通过两种根本不同的方法。在第一种方法中，现代深度神经网络（DNN）通过量化/剪枝和共享技术、低秩分解方法、传递/压缩加权方法和知识蒸馏等技术进行压缩。第二种方法是协同设计硬件和算法，更接近地模拟生物智能惊人的效率背后的解剖和生理特征。这种方法被称为神经形态计算，也叫类脑计算，是描述一组大脑启发的硬件和算法，用于具有不同程度生物学合理性的神经网络。

类脑计算通过硬件和算法设计相结合来模拟生物神经系统的行为，通常以大脑为中心的一种大规模并行和分布式计算的新范式。在神经形态计算的最纯粹形式中，其思想是硬件应该完全定义算法，而不需要软件，尽管能效高但灵活性极低。然而，硬件/软件协同设计通常通过不同层次的设计抽象来增加更多的灵活性，降低了一定的效率。在该领域，一个正在进行的争论和最大的挑战之一是确定需要对生理过程的多少细节进行建模，以忠实地捕捉潜在的计算原理。在神经元的情况下，有各种各样的复杂程度不同的模型，在该系列的一端，如多室 Hodgkin-Huxley[36]模型，Fitzhugh-Nagumo[37]模型，Morris-Lecar[38]模型能捕捉到粒子传导的详细行为并能再现尖峰神经元的几种复杂行为，然而这些类型的模型是计算密集型的，通常为神经科学的研究保留。在另一端是简单的阈值模型，如 McCulloch-Pitts[39]模型只捕捉简单的神经行为(如神经元是否高于或低于特定的速率)，其他简单的模型，如 Sigmoid 和 RELUS 只传递关于神经元的尖峰率的信息，而不传递详细的时间信息。由于它们的简单性，计算效率非常高，大多数现代 DNN 都使用它们。最后，第三组尖峰神经元模型位于频谱的中间，具有足够的复杂性来捕获尖峰统计的重要时间信息，但具有足够的抽象性来提高计算效率，这些模型也适合于相对简单的硬件实现，关于这些模型的更多细节将在 2.2 小节中给出。

早期的类脑计算的研究工作主要集中在使用模拟电路建模真实的生物系统[40-43]，如

开发了硅视网膜和感觉运动系统[42]、利用 CMOS 芯片设计了一种与生物耳蜗原理相同的电子耳蜗[43]。近几十年来，类脑计算的实现已经转移到数字领域，以获得更好的噪声恢复能力和更高的可伸缩性。研究的重点也从单个神经元的实现扩展到网络和神经元间的通信架构，除了数字系统，新兴材料和器件，如记忆电阻器，相变材料，光子电路也正在研究类脑计算的混合解决方案。

2.2 脉冲神经网络

SNN 比传统的人工神经网络(ANN)具有更多生物学上可信的特征，与生物神经系统相似，SNN 本质上是一个动态和有状态的网络。SNN 最明显的特性是信息被表示、传输和处理为离散的脉冲事件，也称为动作电位[44]。尖峰是生物神经系统中的电脉冲，在 SNN 数学模型中，尖峰通常用狄拉克函数表示。虽然脉冲可以实现低功耗的信息传输和处理，但不可微的狄拉克函数也给 SNN 训练带来了很大的挑战，阻碍了梯度下降算法的应用。此外，与神经网络中神经元间的连接通过权值系数控制的线性比例无损传递信息不同，SNN 的连接/突触可以包含多个状态变量和参数，这一特性使 SNN 在处理时空序列方面更加强大，但也增加了其实现的复杂性。值得注意的是，SNN 和 ANN 之间的边界并不总是明确的，虽然大多数 SNN 模型使用尖峰，但也有基于速率的 SNN 模型，其中神经元的输出不再是离散的尖峰，而是实值瞬时尖峰速率。这样的模型可以解释为 ANN[45]，还有将 SNN 和 ANN 融合在一起的[46,47]和硬件[48]，在这项工作中，SNN 这个名字被用来指代产生尖峰作为其输出的模型，其中神经元和突触是最基本的计算单元。在高抽象层次上，所有的峰值模型都具有以下共同特征：（1）它们处理来自多个输入的信息，产生单个或多个峰值；（2）兴奋性输入增加了脉冲产生的概率，抑制性输入降低了脉冲产生的概率；（3）使用至少一个状态变量来描述其动态特性，当模型的内部变量数值达到某一阈值时，模型将产生一个或多个峰值。神经元通过一种叫做突触的特殊连接来相互连接和交流，与神经元模型相似，突触模型在复杂性和生物学上的合理性也各不相同。

2.2.1 神经元模型

在 SNN 中使用的尖峰神经元模型需要有足够的复杂性来捕获生物神经元的关键动态特性，同时不需要大量的计算或硬件开销。图 2-1 显示了各种文献提出的不同模型与其计算成本的比较，数据来源于文献[49,50]，其中纵轴表示模型显示的生物特征的数量，正误差条表示如果仔细调整参数，模型能显示的特征的最大数量，横轴表示模型的计算复杂度，图中的正误差条表示如果仔细调整参数模型能显示特征的最大数量。比较不同的模型发现增加更多的生物特征会导致计算成本的指数增长，本文将选取带泄露积分触发模型（LIF，Leaky-Integrate-and-Fire）、霍奇金-赫胥黎模型（HH，Hodgkin-Huxley）、Izhikevich 模型这三种具有代表性、经典的神经元模型进行详细讲解并剖析其计算特性。

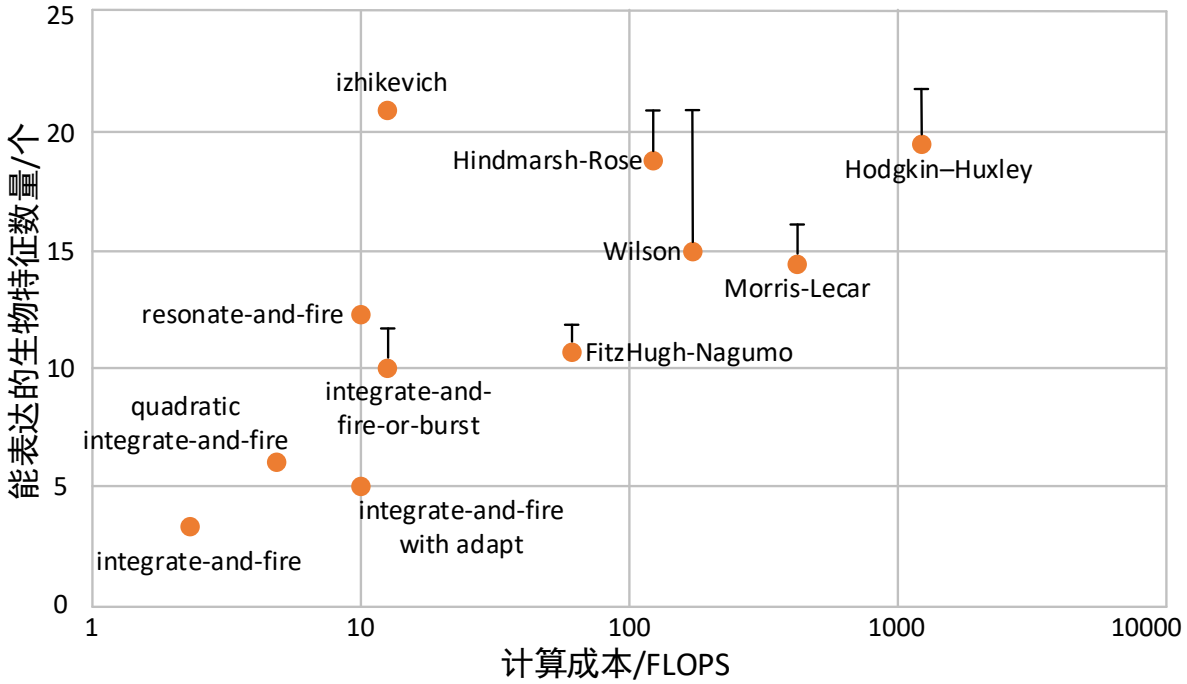

图 2-1 比较不同的尖峰神经元模型

（1） LIF 模型

泄漏的积分-点火模型是最简单的神经元峰值[51]模型之一。它只涉及一个微分方程，它把一个神经元膜模型成一个电容 C 的电容器和一个电阻 R 的电阻器并联，其膜电位动力学更新方程为：

$$\begin{cases} \tau_m \dfrac{du}{dt} = -u + RI \\ f(u \geq u_{th}) then \{ u = u_{rst} \end{cases} \tag{2-1}$$

其中 τ_m=RC 是电路的时间常数， I 是刺激电流。当膜电位(u)的数值不断积累并达到阈值模电位(u_{th})时，此神经元就会产生一个脉冲，然后膜电位就会被重置成静止值(u_{rst})且该神经元在接下来一段时间内不允许膜电位发生变化，称为不应期。因此这个模型很简单、没有表现出任何具有生物特性的尖峰行为，公式(2-1)可以进一步简化为以下公式：

$$\begin{cases} \dfrac{du}{dt} = -\dfrac{u - u_{rst}}{\tau_m} + \sum_{i=1}^{n} w_i I_i, & u < u_{th} \\ u = u_{rst}\ , & u \geq u_{th} \end{cases} \tag{2-2}$$

$\sum_{i=1}^{n} w_i I_i$ 是权重与输入脉冲的乘积，累加到膜电位上，n 表示别的神经元连入当前神经元的数量，当有一个输入峰值到达时， I_i 被设置为 1，否则为 0。如果没有超过阈值，膜电位呈指数衰减，直到达到剩余电位的最大值，如图 2-2 为输入电流为 0 时 LIF 神经元膜电位的衰减曲线图。

其中， $R = 8.22\ \text{M}\Omega$, C=5.0675 nF, $u_{th} = 30$ mV, $u_{rst} = 0$ mV, $t_r = 5$ ms， t_r 表示在每次发射脉冲后的绝对不应期时间。

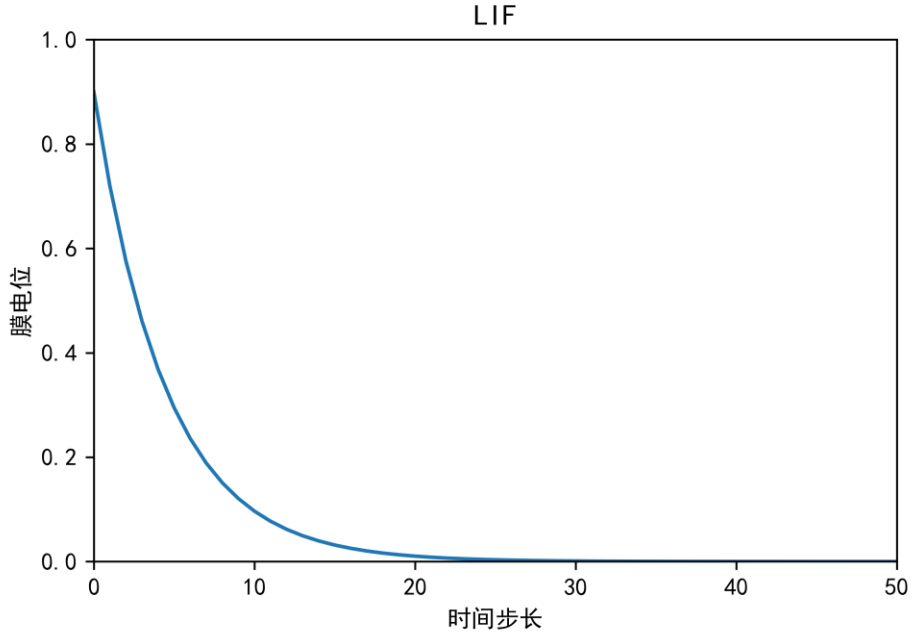

图 2-2 LIF 神经元的衰减曲线图

（2） HH 模型

霍奇金-赫胥黎模型诞生于[52]，是最具生物物理学意义的神经元峰值模型之一，该模型是基于对鱿鱼巨轴突的研究而设计的，旨在描述神经元对外部电流刺激的响应，并考虑了不同离子通道和泄漏电流的影响。Hodgkin-Huxley 模型包含一个非线性常微分方程组，其动力学方程为(2-3)公式：

$$C_m \frac{du}{dt} = -g_{Na}(u - V_{Na})m^3 h - g_K(u - V_K)n^4 - g_L(u - V_L) + I \tag{2-3}$$

其中 u 是膜电位；C_m 是膜面积上的有效电容；t 是时间；g_{Na}、g_K、g_L 分别为钠、钾、渗漏通道的电导；V_{Na}、V_K 和 V 分别是钠、钾和泄漏通道的反转电位；I 为每面积的刺激电流；m，n，h 是由公式(2-4)决定：

$$\begin{cases} \dfrac{dm}{dt} = \alpha_m(1-m) - \beta_m m \\ \dfrac{dn}{dt} = \alpha_n(1-n) - \beta_n n \\ \dfrac{dh}{dt} = \alpha_h(1-h) - \beta_m h \end{cases} \tag{2-4}$$

其中系数 α 和 β 由公式(2-5)决定：

$$\begin{cases} \alpha_m = \dfrac{2.5 - 0.1u}{\mathrm{Exp}[2.5 - 0.1u] - 1} & \beta_m = 4\,\mathrm{Exp}[-\dfrac{u}{18}] \\ \alpha_{\mathrm{n}} = \dfrac{0.1 - 0.01u}{\mathrm{Exp}[0.1 - 0.01u] - 1} & \beta_m = 0.125\,\mathrm{Exp}[-\dfrac{u}{80}] \\ \alpha_h = 0.07\,\mathrm{Exp}[-\dfrac{u}{20}] & \beta_h = \dfrac{1}{\mathrm{Exp}[3 - 0.1u] + 1} \end{cases} \tag{2-5}$$

其中 $\mathrm{Exp}[\gamma]$ 表示 e^{γ}，常数通常由公式(2-6)给出：

$$\begin{cases} V_{Na}=115\ \text{mV} \quad g_{Na}=120\dfrac{\text{ms}}{\text{cm}^2} \\ V_K=-12\ \text{mV} \quad g_K=36\dfrac{\text{ms}}{\text{cm}^2} \\ V_L=10.6\ \text{mV} \quad g_L=0.3\dfrac{\text{ms}}{\text{cm}^2} \\ C_m=1\dfrac{\mu F}{\text{cm}^2} \end{cases} \tag{2-6}$$

通过参数的修改，HH 模型可以表现出所有的神经元尖峰行为，因此这个模型在生物物理学上意义重大。但从这些方程中，很容易看出霍奇金-赫胥黎模型是一个非常昂贵的计算系统，因此在该模型上做的应用并不多，当把上述参数带入方程，设计时间步长为 0.01ms、外部电流为 10、仿真 100ms 的 HH 模型膜电位波形变化图。

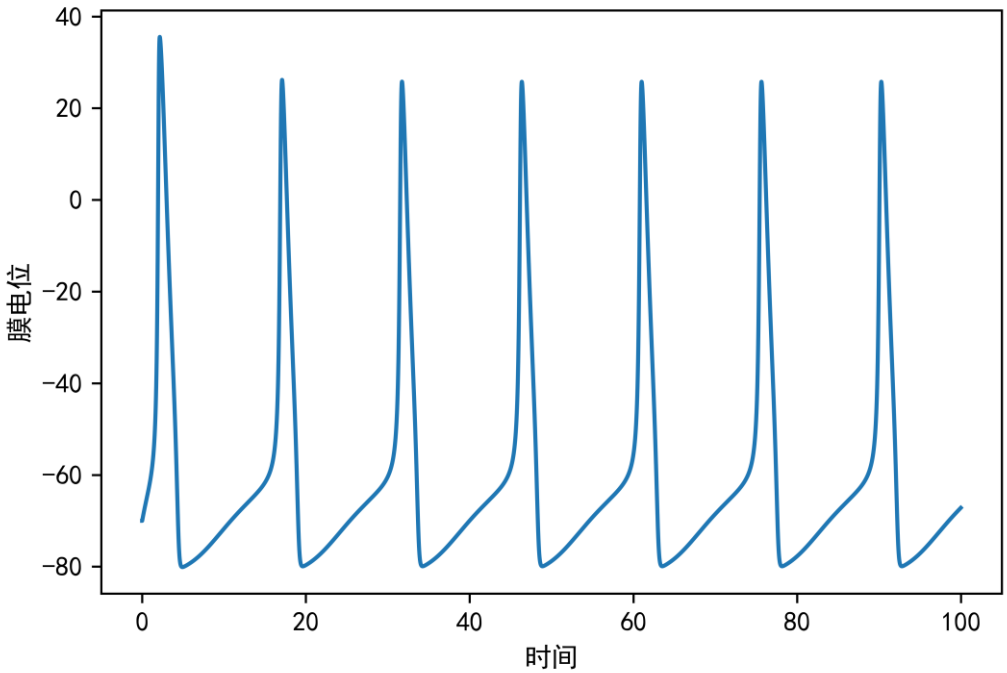

图 2-3 HH 神经元的波形变化图

（3） Izhikevich 模型

2003 年 Izhikevich 引入了一个基于映射的模型[53]，该模型结合了霍奇金-赫胥黎模型生物学合理性以及带泄露积分触发模型计算高效的优点。因此，Izhikevich 神经元模型已被普遍接受为准确且计算成本较低、计算高效的模型，由两个耦合的常微分方程组成，尽管计算相对简单，但可以至少实现 20 种生物尖峰神经元的计算特性。下面介绍了两个耦合的常微分方程：

$$\begin{cases} \dfrac{dv}{dt}=0.04v^2+5v+140-u+I\_e \\ \dfrac{du}{dt}=a(bv-u) \end{cases} \tag{2-7}$$

复位条件如下：

$$if\,(v \ge v_{th}) \quad then \begin{cases} v = c \\ u = u + d \end{cases} \tag{2-8}$$

其中，v是膜电位，u是恢复变量，I_e是外部施加电流，v_{th}是阈值电位，a是恢复变量u的时间尺度，b是u对v的敏感度，c是神经元在发射脉冲尖峰后v的复位值，d代表神经元在发射脉冲尖峰后u的复位值。Izhikevich 模型不仅计算方程较为简单，还可以通过调节a、b、c、d等参数的值产生不同的尖峰行为，以表现出神经元多样的生物特征行为，图 2-4 为文献[53]提到的其中一种 Izhikevich 的简化波形，此时$a=0.02$，$b=0.2$，$c=-65$，$d=2$，$v_{th}=30$。此外，也有文献证明了随着时间步长[54,55]的变化，Izhikevich 模型的行为也发生了变化，当然如果想提高精度时，这个模型需要的计算也会进一步提升。

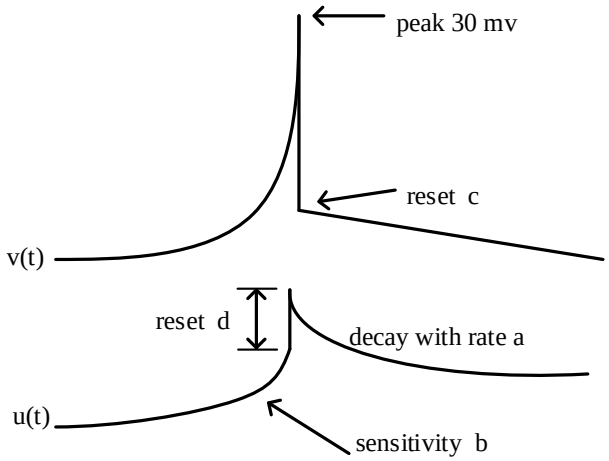

图 2-4 Izhikevich 神经元简化波形

2.2.2 突触模型

脉冲神经网络中神经元之间的交流是在突触上进行的，当突触前神经元产生动作电位时，会对脉冲事件信息进行相应的处理，并传递到突触后神经元以刺激后神经元膜电位的变化，整个过程结束算完成了一次脉冲传导，如图 2-5 所示。一般来说，突触可以根据权值变化状态分成两类，静态突触和动态突触，静态突触在网络仿真过程中权重保持不变，而动态突触在仿真过程中能根据一定的规则动态调节权重值。

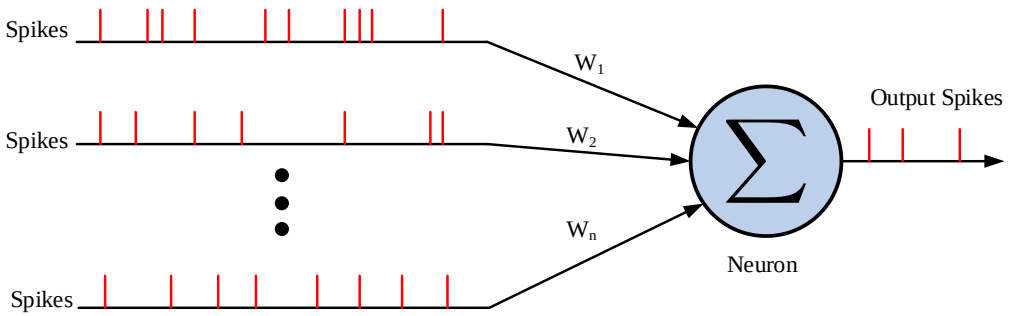

图 2-5 脉冲传导

相比较静态突触，动态突触就计算相对复杂，因其权重会随着仿真增加或者降低，其中比较典型的是 STDP(spike-time-dependent plasticity)突触模型[56]，这是一种基于输入（突触前）脉冲和输出（突触后）脉冲之间的差异局部更新突触权值的无监督学习方法，

与传统的只增加权值的 Hebbian 规则相比，它既增加权值又减少权值，如图 2-6 所示，Δw 的方程如公式 2-9 所示。

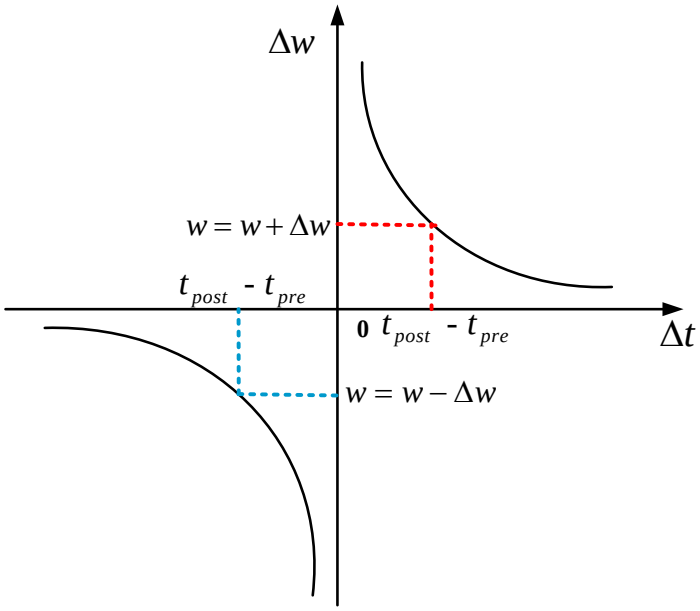

图 2-6 STDP 曲线

$$\Delta w=\begin{cases} A_{+}\exp\left(-\dfrac{\Delta t}{\tau}\right), & \Delta t\geq 0 \\ A_{-}\exp\left(-\dfrac{\Delta t}{\tau}\right), & \Delta t<0 \end{cases} \tag{2-9}$$

这里 Δt 是输出峰值时间 t_{post} 和输入峰值时间 t_{pre} 之间的差值，A_{+} 和 A_{-} 分别是 Δt 值大于 0 和小于 0 时的系数，τ 是突触泄漏的时间常数。当输入的脉冲事件信号引起神经元向后发出输出脉冲事件信号时，$t_{post} > t_{pre}$，根据 STDP 曲线的规则，将会增加突触的权重数值，反之，$t_{post} < t_{pre}$，则降低突触的权重数值。在本工作中，在现有的设计[57]的基础上，采用了 STDP 算法的一个变体，因为它对于在线学习的硬件实现非常简单，算法由式(2-10)描述：

$$\begin{cases} \dfrac{d\ \text{apre}}{dt}=-\dfrac{\text{apre}}{\tau_{pre}} \\ \dfrac{d\ \text{apost}}{dt}=-\dfrac{\text{apost}}{\tau_{post}} \\ \text{onpre}:\begin{cases} w=w-\eta_{post}\text{apost} \\ \text{apre}=A_{pre} \end{cases} \\ \text{onpost}:\begin{cases} w=w+\eta_{pre}\text{apre} \\ \text{apost}=A_{post} \end{cases} \end{cases} \tag{2-10}$$

其中，apre 和 apost 为两个变量($\text{apre}\geq 0, \text{apost}\leq 0$)，$\eta_{post}$ 和 η_{pre} 为学习速率，apre 和 apost 为两个常量。当一个输入脉冲(即突触前脉冲)到达时，onpre 更新被执行，当神经元被触发(即突触后脉冲)，onpost 更新被执行。

2.3 类脑仿真框架

根据不同的开发来源和发展路径，类脑计算基础软件可以分为三类(图 2-7):

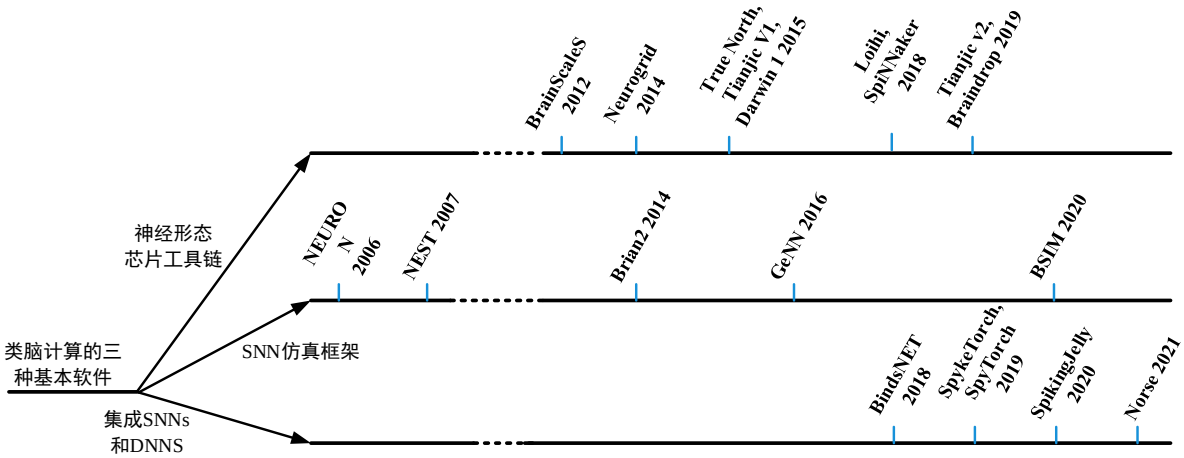

图 2-7 类脑计算的基本软件发展

第一类是开发神经形态芯片并提供他们自己的软件工具链，包括 IBM 的 TrueNorth 芯片[58]、SpiNNaker、清华大学的 Tianjic V1、V2[59]和浙江大学的 Darwin 芯片，以及斯坦福大学的 NeuroGrid[60]等。这种类型的研究通常使用端到端软件和硬件的协同设计解决方案，制造特定于目标芯片的工具链，甚至应用模型。此外，有一些新兴的研究试图通过构建一种“通用”领域特定语言或开发框架来避免被绑定到特定类型的芯片上，将软件和硬件连接起来，这种研究通常还处于类脑计算设计的早期基础阶段，目前仍然存在许多问题尚未解决，对实际芯片缺乏普遍支持，难以用于多样的类脑计算科学研究。

第二类是 SNN 模拟仿真框架，源于计算神经科学领域，其目标是理解生物系统，包括 NEST、NEURON、Brain2、GeNN 等。这类工作的特点是支持更详细的神经元活动的动态过程描述，支持多种类型的突触模型和多种类型的突触可塑性，从而能够更详细地模拟生物神经网络，同时要求用户具有一定的计算神经科学基础。与深度学习领域广泛使用的开发框架相比，其使用更为复杂:框架通常使用 C 语言开发，跨平台能力尚有不足，对各种后端硬件的深度优化还缺乏广泛的支持，包括在数值模拟中广泛使用的通用图形处理单元（GPGPUs）。

第三类是近年来出现，将 SNN 表示/计算特性集成到广泛使用的深度学习开发框架中[61-63]。这类工作的主要受众是脑力启迪型智能应用开发者，其目的是结合深度学习框架开发的便利性(包括 BP 算法及其变体)与 SNN 事件稀疏性的特点，还可以将深度学习领域的各种资源应用于 SNN 进行加速。一些研究[64,65]发现生物神经元的详细结构(如树形突起)不仅仅是为了连接，它们还可以进行操作，即神经元本身也可能是一个多层网络，这就需要有能力构建突触的精细动力学模型，这对现有的基于张量和张量计算的框架建模方法是一个潜在的挑战。此外，这类工作基本上处于初级阶段，现有的深度学习框架的后端[包括神经网络编译器[66]]通常是为深度学习加速芯片设计的，并不一定就适用于类脑计算芯片。

2.3.1 脉冲神经网络仿真器

与神经形态芯片相比，脉冲神经网络仿真器是以软件的方式，试图在传统计算机上模拟大脑启发计算的过程，是计算神经科学领域的重要模拟工具，接下来将对几种经典的、主流的仿真器的特点进行介绍。

（1）NEURON

NEURON[67]是一款发展超过 35 年主要用于模拟具有复杂分支解剖结构和生物物理膜特性的神经元网络的仿真器，这包括膜附近的细胞外电位、多通道类型、不均匀的通道分布和离子积累。它可以处理扩散反应模型，并将扩散功能集成到突触和细胞网络的模型中，使用神经元模拟的形态学详细模型能够表示神经元电学和生物物理特性的空间多样性。与其他许多仿真器不同的是，NEURON 使用前向欧拉方法，而不是后向欧拉和克兰克－尼科尔森法，以获得更高的准确性和稳定性。

NEURON 编程主要包含两种语言，高级计算语言（HOC，High Order Calculator）和模型描述语言（NMODL，Model Description Language）。HOC 是一种与 C 语言相近的解释型语言，可用于实现抽象数据类型和数据封装，从 2006 年开始，添加了 python 接口，使得 NEURON 得模型更具兼容性和可移植性。NMODL 语言的应用场所主要是对于细胞膜电生理特性的描述，使模型可以用动力学方案或联立微分方程和代数方程组来表示，使用 HOC 解释程序提供得接口将 NMODL 定义的生物机制插入到 HOC 定义的神经模型中，并且通过自动转化为 C 语言来提高效率。

它可以用丰富的真实性和细节来模拟神经元的行为，从代表离子通道到代表树状结构中电信号的传播，为在神经元和神经元网络中实现电信号和化学信号的生物学模型提供强大而灵活的环境，可以用于实验与数据密切相关的问题，尤其是那些具有复杂解剖学和生理学特性的神经元问题。

（2）NEST

NEST 最早是由 Gewaltig 和 Diesmann 等人设计，是一款用于尖峰神经网络模型的模拟器。NEST 支持不同层次的生物细节的神经元和突触模型，重点关注生物系统的动力学、大小和结构，并且提供 50 多种神经元模型和 10 多种突触模型，包括典型的神经元模型如 Izhikevich 和 Hodgkin-Huxley，以及不同变体的 STDP 模型。NEST 在定义神经元网络方面非常灵活，提供了方便有效的命令，可以在仿真过程中随时检查和修改神经元和连接状态，NEST 是由 C++实现的，它提供了一个名为 PyNEST 的 Python 接口。

NEST 可以充分利用本地网络中的多线程、多核和多计算节点。在计算机集群上，NEST 分发网络到可用计算机上，每台计算机都创建自己的部分网络，并且仅存储与自身神经元的连接，这不仅缩短了仿真的运行时间，而且还增加了可用于 NEST 仿真的计算机总内存。在单台计算机上，NEST 会把部分网络分配到可用处理器上，每个处理器执行网络的一小部分。减少的仿真时间与处理器数量成比例，NEST 利用了这一事实让每个处理器都贡献自己的快速缓存。

NEST 具有模块化架构，可以添加模块并针对特定问题定制 NEST，主要有以下三种方式进行扩展：1.在模拟内核级别，可以为神经元和设备添加新模型；2.在模拟语言级别，可以添加函数和库；3.最后，在 C++层面，可以扩展模拟语言。NEST 还可以有效地从单核笔记本扩展

到多节点超级计算机，而且 NEST 有一个大规模的开发人员社区，在那里有许多例子被分享，以帮助用户构建他们自己的模拟项目。

（3）Brian2

Brian2 是一个 SNN 模拟工具，是在考虑计算神经科学的基础上开发出来的，因此其旨在应用神经形态计算，大多数的评估也都是针对神经科学工作量进行的。Brian2 是用 Python 编写的，用户很容易学习和开发，很灵活且易于扩展，方便用户使用微分方程来定义神经元和突触的模型和仿真方法。Brian2 使用了自己的通用描述方法，这种特殊的编写方式允许它在不修改核心代码的情况下，很容易生成针对不同平台的代码，增强了适应性和可伸缩性。Brian 是一个时钟驱动的模拟器，这意味着模拟被分解成时间步 dt，在每个时间步 t 神经元和突触模型被更新为时间 $t+dt$。

（4）Nengo

Nengo 是一款基于神经工程框架(NEF)构建大规模模型的软件工具，NEF 是一种认知过程如何在生物基质中实现的理论，也是一款由时钟驱动的模拟器。Nengo 的最初版本是用 Java 编写的，而最新版本是用 Python 实现的，Nengo 有自己的基于 CPU 的模拟器，以及 FPGA 版本 NengoFGPA，Nengo 也有一个针对 Intel Loihi 架构的模拟器后端版本，称为 Nengo Loihi。

（5）GeNN

GeNN[68]是一个用于 SNN 仿真的软件工具包。GeNN 通过代码生成将神经元模拟的不同描述转换成统一的 C++代码，并与 NVIDIA GPU 连接生成 C++可执行文件，GeNN 支持 SpineML、Python (PyGeNN)和 Brian2 (Brian2GeNN)接口。代码生成还使 GeNN 能够支持不同的平台，如 Linux、Mac OS X 和 Windows。GeNN 主要优化大规模神经网络的性能，大大提高了仿真速度。虽然代码生成限制了仿真的灵活性(不能在运行时调整网络结构)，但这些方法提高了整体执行性能和效率。

2.3.2 脉冲神经网络仿真器性能对比

根据上一小节的介绍，可以知道这些 SNN 软件仿真器主要的设计用途都不尽相同，也都有各自的特点，因此他们在 SNN 仿真中表现出来的计算性能、特性也存在着差异。通过查阅相关资料，发现文献[69]主要介绍并对第二类里经典的几款软件仿真框架做了以下三种实验：

（1）前馈网络

如图 2-8 显示了 5 个模拟器在分层前馈架构上的结果，文中评估了不同的网络、不同的隐含层数量和每层不同的神经元数量。这个评估中的每个网络除了测试中指定的隐藏神经元外，还有一个输入层 64 个神经元和一个输出层 10 个神经元，其中 Bindsnet 和 NEST 测试结果最好，在 15 分钟的时间内完成所有测试。

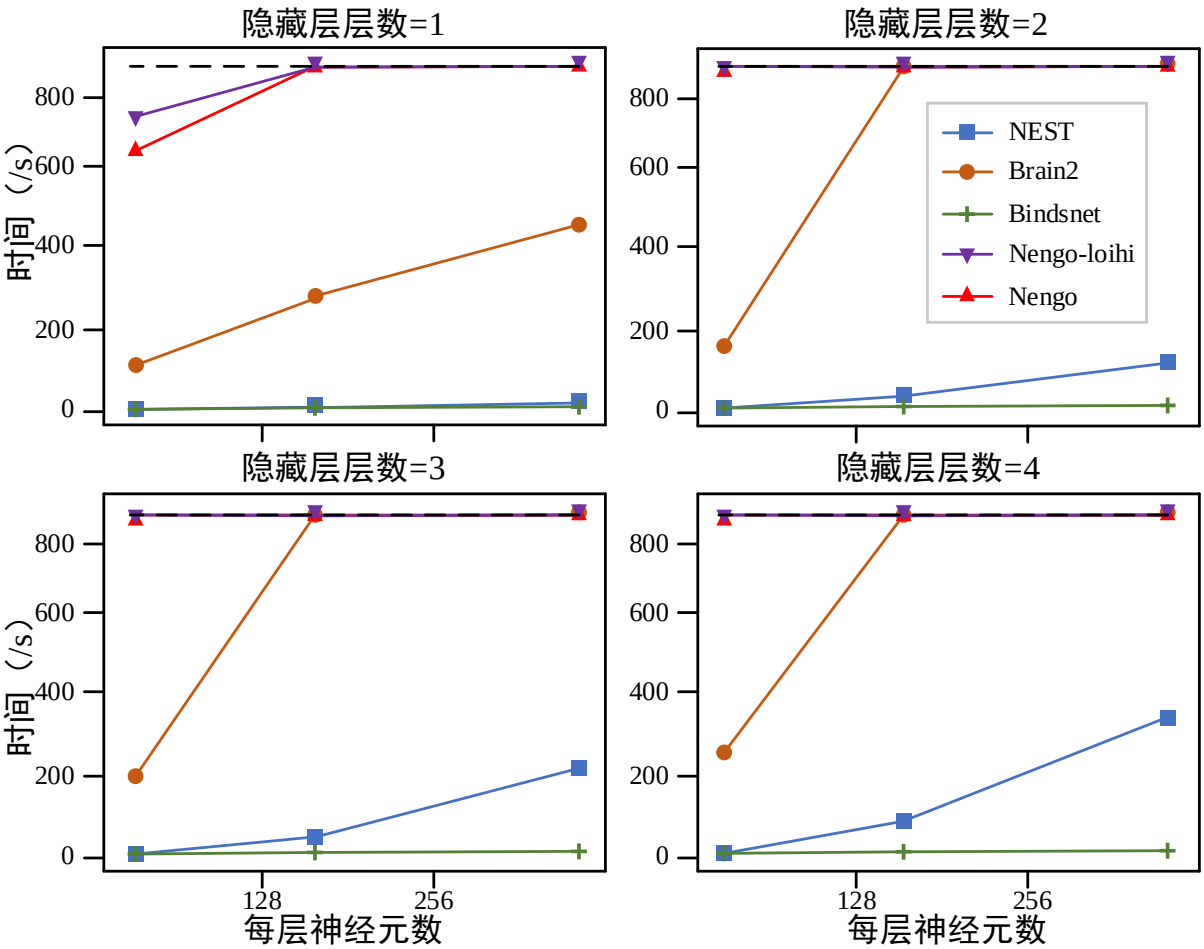

图 2-8 基于单核系统的前馈网络评估

（2）进化算法

模拟进化算法的活动中，需要在很短的时间内加载和模拟许多非常小的网络，并在每次模拟之后监视它们的输出。在这种情况下，该文献通过增加种群规模或每一代评估的网络数量来衡量问题的难度，并对每个种群规模总共评估了三代，如图 2-9 所示。在该例子中，可以看出 NEST 模拟器执行的比其他模拟器好，网络规模在几十到几千之间。

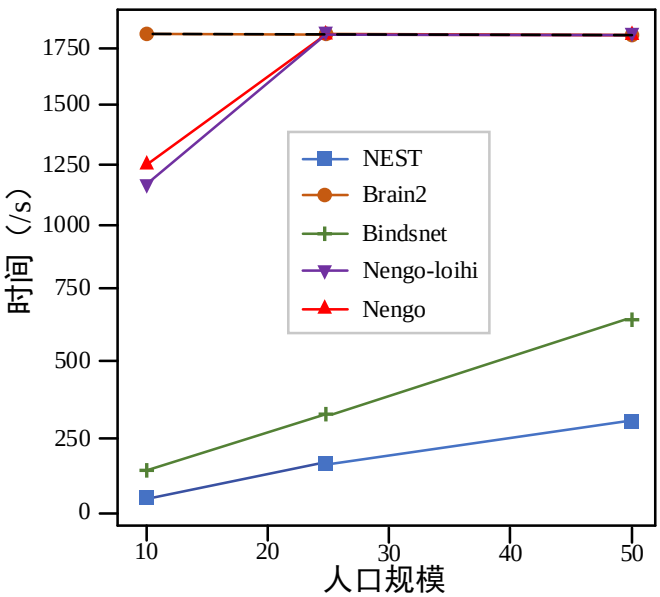

图 2-9 模拟器运行进化算法五代不同的人口规模

（3）最短路径

在尖峰神经网络模拟器上评估了一个单源最短路径计算任务，测试了不同大小的近似幂律图，它们都有相似的突触密度，并将最大计算时间限制在 15 分钟。图 2-10 给出了各仿真器上最短路径的计算结果，从图中可以看出，Nengo 和 NengoLoihi 没有在规定时间内完成任何网络规模的最短路径测试。随着仿真时间的延长，NEST 的性能明显优于其他仿真器，特别是对于较大的图。

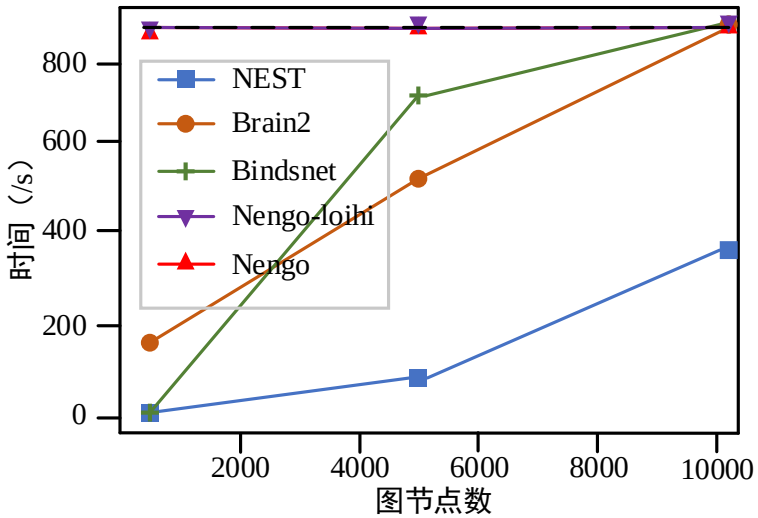

图 2-10 单核 CPU 上不同模拟器的 SNN 最短路径时间统计

综合比较上述各图中，各模拟器在几种实验中的不同表现，发现 NEST 不仅在小型稀疏网络中表现最好，还是在重构能力方面最灵活的模拟器，在运行 SNN 进化算法时，NEST 比其他仿真器明显提升至少两倍的提速。同时，对于大型稀疏的网络，NEST 的多节点和多线程也能比其他模拟器至少有两倍的加速。NEST 是一款用于尖峰神经网络模型的模拟器，支持不同层次的生物细节的神经元和突触模型，重点关注生物系统的动力学、大小和结构。因此，本文选择 NEST 作为后续实验的基础平台框架。

2.4 经典的类脑仿真计算模型

2.4.1 丘脑外侧膝状核网络模型

丘脑外侧膝状核模型(Lateral Geniculate Nucleus, LGN)在传统上被视为视觉皮层的门户[70]，是控制注意力反应增益的看门人，LGN 会把视觉感觉信息映射到初级视觉皮层(V1)。来自视网膜(RET，Retina)的感觉信息由神经节细胞轴突传递到 LGN，神经节细胞在丘脑皮质中继(TCR，Thalamo-Cortical Relay)和中间神经元(IN，Interneurons)细胞上产生兴奋性突触，TCR 细胞的轴突是将感觉信息传递到视觉皮层(主要是第 4 层)的主要载体，视觉皮层(主要是第 5 层和第 6 层)被认为向 LGN 的 TCR 和 IN 细胞发送重要的反馈信号。丘脑网状核(TRN，Thalamic Reticular Nucleus)是包围丘脑背侧的一层很薄的神经元组织，从 TCR 到视觉皮层的所有信息传递路径都有主要的分支，这些分支使兴奋性突触到达 TRN 细胞，同样，所有从视觉皮层到 TCR 和 IN 细胞的反馈也通过轴突分支传递到 TRN。因此，TRN 具有战略性的地位，这使得它能够“监控”LGN 和视觉

皮层之间的所有交流。除此之外，在初级视觉皮层受损但仍然可以对某些图像做出反应的盲视行为中，Ajina[71]等人指出 LGN 在绕过 V1 的通路中起到关键性的作用。该模型包含约 70-80%个 TCR 细胞和 20-25%个 IN 细胞，表明 TCR 与 IN 之比大概是 4:1[6]，而在 TRN 上没有一个精确可用的相应数据信息，所以在此模型中任意设置为单元格数量的两倍，即将 TCR，IN 和 TRN 群体中的细胞比例设置为 8:2:4。

本文把 LGN 模型的突触布局成如图 2-12 所示，该模型的突触分布基于哺乳动物和啮齿类动物背侧膝状核的实验数据。LGN 的 TCR 和 IN 细胞都接受来自视网膜刺突神经元的兴奋性输入，IN 细胞在自身和 TCR 细胞上制造抑制性突触，而关于从 TCR 到 IN 的突触通路的信息是模糊的，在这里被忽略了，TCR 细胞在 TRN 细胞上产生兴奋性突触，TRN 细胞在 TCR 细胞和自身上产生抑制突触。其中网络中的突触连接是稀疏的，实线表示兴奋性连接，虚线表示抑制性连接，线上的数字代表模型中突触前细胞种群和突触后细胞种群之间的泊松连接概率。

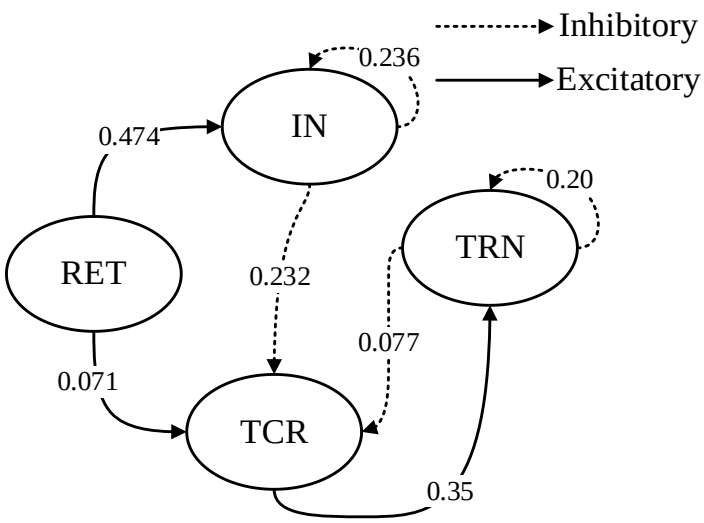

图 2-11 丘脑外侧膝状核网络模型拓扑图

2.4.2 液体状态机模型

液体状态机(LSM，Liquid State Machine)是一种使用 SNN 的储层计算机。LSM 包含了大量具有时空属性的神经元节点，在被外部输入刺激或者接收到其他神经元的事件输入时，内部的每个神经元节点都可以变成输入神经元（就像是一块石头落入水中在表面产生的波纹一样），各节点间的连接是随机生成的，当然在确立连接后就不再改变。这说明两两神经元之间连接权值不需要训练，一般可以采用灵活多变的算法来训练输出层以达到所需的精度及目标任务。文献[72]中提到的 LSM 模型，网络拓扑示意图如图 2-12 所示，图中中间层用若干个 Izhikevich 神经元随机连接构成，输入层用泊松发射机提供输入信号，因为中间层是静态的权重，所以神经元之间的连接用静态突触来实现。

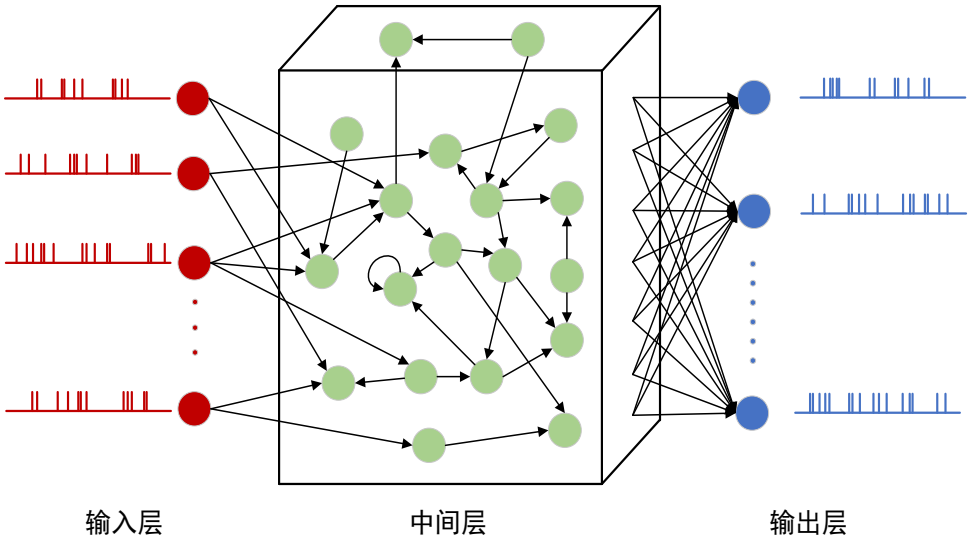

图 2-12 LSM 网络拓扑图

2.5 本章小结

本章首先通过对类脑计算基本概念的介绍，引出了脉冲神经网络的概念，并对脉冲神经网络中常见的 LIF、Izhikevich、HH 神经元模型以及静态、STDP 突触等模型的基础知识进行介绍分析。然后对类脑的仿真框架进行一个整理和梳理的工作，把与类脑计算相关的基础软件分为三个大类：一、神经形态芯片以及提供的软件工具链；二、脉冲神经网络模拟仿真框架；三、将 SNN 表示/计算特性集成到广泛使用的深度学习开发框架。再对第二类几种较为主流的、经典的 SNN 模拟仿真器进行一个介绍，并通过查阅文献资料比较各种仿真器的性能差异，选出可以进行分布式计算且在仿真实验表现较好的 NEST 作为后续实验的基础软件框架。在最后，本文介绍了两种较为经典的脉冲神经网络模型，将用于在最后实验阶段的结果比较。

第三章 脉冲神经网络量化方法研究

相比较传统 CNN 的量化，脉冲神经网络的量化除了有对整体网络模型的量化，还有关于神经元、突触更新方程的量化。对于网络模型的量化，这是一种在压缩和加速深度神经网络推理领域获得大量运用的优化技术，包括固定位量化和混合精度量化。对于神经元等更新方程的量化，其实就是对连续的更新方程进行离散数值的求解，因为不同的求解方法将会引起方程不同程度的量化，也会导致神经元、突触等的更新计算的复杂度和时间。因此选择一个好的数值求解方法，能以更小的计算代价获得更高的仿真精度，以此来提升脉冲神经网络仿真的性能。

本章在回顾 CNN 较为成熟的量化方法与技巧后，提出了一种基于 SNN 的位宽压缩方法，同时对 NEST 的工作负载进行量化建模以便理论推导出计算密集点，最后是通过量化不同的数值求解方法的计算开销，以找出某数值求解方法能最小的计算时间代价获得更高精度的 SNN 网络仿真，同时提升脉冲神经网络性能的提升。

3.1 神经网络的轻量化

3.1.1 CNN 的轻量化

CNN 的轻量化其实就是一个网络模型的压缩过程，常用的方法有剪枝和位宽量化等，这些方法能节省内存开销和减少计算量操作，在下文种主要是介绍针对位宽的压缩量化研究

在对 CNN 的固定位宽量化研究里，最为经典的就是使用二元化神经网络（BNN，Binarized Neural Networks），该网络在计算参数的梯度时使用二元的权值和激活值，因此在网络正向推理的过程中，二值化神经网络能大幅降低所需内存的大小和访问的次数，并将大多数的运算操作替换为逐位操作，大幅度提高了能效。具体的做法是，当训练 BNN 时，将权重或者激活值限制为+1 或者-1，这里存在两种不同的二值化函数，一种是确定的如公式 3-1，另一种是随机的二值化如公式 3-2。

$$x^b = \mathrm{Sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise,} \end{cases} \tag{3-1}$$

其中 x^b 是一个二值化变量(0 或者 1)，而 x 是一个实值变量，它能很简单的实现。

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x), \\ -1 & \text{with probability } 1-p, \end{cases} \tag{3-2}$$

其中，σ 为“hard sigmoid”函数：

$$\sigma(x) = \mathrm{clip}(\frac{x+1}{2},0,1)=\max(0,\min(1,\frac{x+1}{2})). \tag{3-3}$$

尽管随机化二值函数比确定二值函数更具说服力，但很难实现，因为它需要硬件在量化的时候生成随机比特，因此绝大部分的 BNN 主要使用的是确定性的二值化函数。当然，把权重和激活值都二值化，对于最后网络的准确率肯定是要下降一部分的，甚至在某些数据集上表现得并不理想。

混合精度量化是一种对神经网络更深入、细粒度的量化，需要对每一层的权值、激活值截取一定的范围，然后将该范围进行线性量化。具体的说，对于第 k 层的每个权值 w，首先需要将其映射到 $[-c,c]$ 的范围内，然后将其线性化为 a_k，如公式(3-4)所示：

$$\text{quantize}(w,a_k,c)=\text{round}(\text{clamp}(w,c)/s)\times s \tag{3-4}$$

其中 $\text{round}(\cdot,x)$ 函数能将值截断为 $[-x,x]$，缩放因子 S 被定义为 $s=c/(2^{a_k-1}-1)$，并通过寻找使原始权重分布 W_k 与量化后的权重分布 (W_k,a_k,x) 之间的 KL 散度值最小化的最优值 x 来选择 C 的值，如公式(3-5)所示：

$$c=\arg\min_x D_{KL}(W_k \,\|\, \text{quantize}(W_k,a_k,x)) \tag{3-5}$$

其中 $D_{KL}(\cdot\|\cdot)$ 是表征两个分布范围之间距离的 KL 散度。而对于激活值，除了将截断范围由 $[-c,c]$ 变化为 $[0,c]$，因为激活值（即 ReLU 层的输出）是非负的，值得一提的是，这种混合精度量化方法通过手动来获得最后的量化值，最后推理的准确率并不一定是最高的（即可能不是最优解）。

更进一步的方案如图 3-1 所示不仅考虑到不同的层具有不同的冗余度且在硬件上的表现也不同[73]，还通过硬件加速器在设计循环中的反馈来自动的确定量化策略的最优解。该方案将硬件加速器集成到反馈激励回路中，能让推理模块获得硬件的直接反馈而不是依赖于间接的信号信息，因此，不同的资源约束（如延迟、能耗、模型大小）会在该方案下产生截然不同的最佳量化策略。

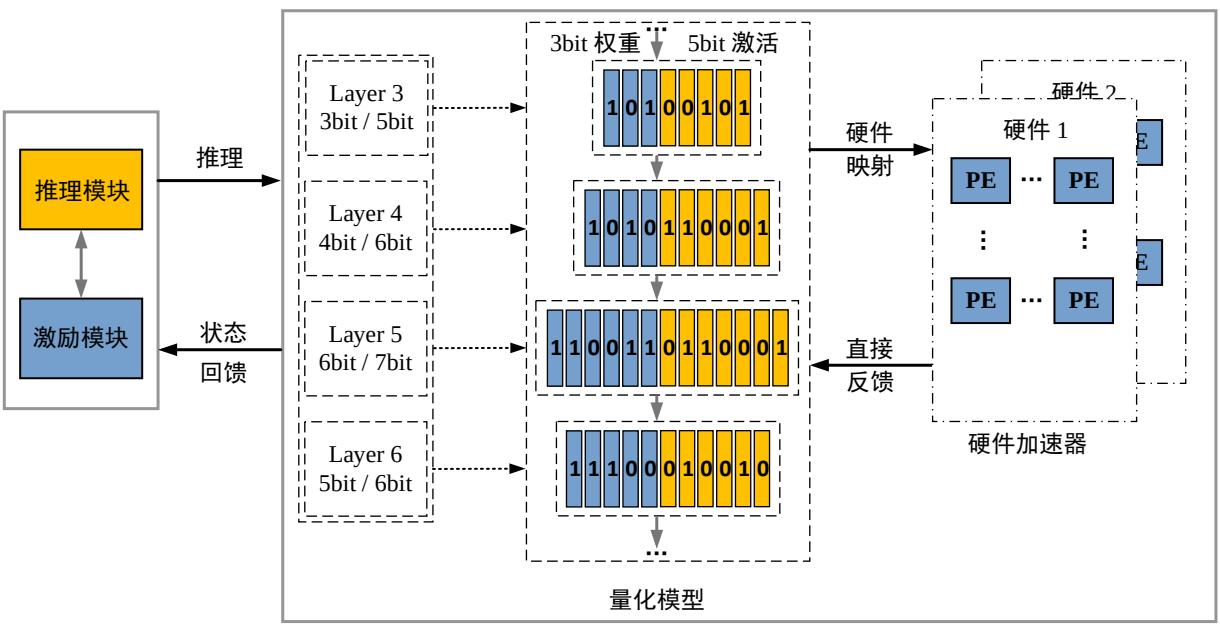

图 3-1 自动-混合精度量化模型示意图

3.1.2 SNN 的轻量化

首先所有的 SNN 模型可以被分为两种，机器学习型脉冲神经网络和仿生型脉冲神经网络，前者可以用于分类、识别、检测等任务，后者主要用于类脑研究与仿生模拟计算实验。其次，相对于 CNN 的量化，SNN 模型的量化具有较大的挑战性，因为 SNN 是一种基于生物神经系统的神经网络模型，其神经元之间的信息传递是通过脉冲信号进

行的，需要采用特殊的量化方法来减少计算复杂度和计算效率。总结常见的 SNN 量化方法为以下四种：

（1） 阈值量化：在 SNN 中，神经元的激活阈值通常是一个连续变化的值，表示神经元发放输出脉冲的条件。为了减少计算复杂度，可以将激活阈值量化为离散值。具体地，可以将阈值分成若干个离散的阈值点，并使用一个离散的变量表示神经元是否激活，这种方法可以减少神经元激活状态的数量，从而降低网络的计算复杂度。

（2） 权重量化：权重是连接神经元之间的参数，其数值通常是连续的实数。在 SNN 中，可以将权重量化为离散值，以减少网络参数的数量，并提高计算效率。常见的权重量化方法包括二进制权重和三进制权重等。二进制权重只能取值+1 和-1，而三进制权重可以取值+1、0 和-1，可以更好地表示网络中的零值。

（3） 时间量化：SNN 中的脉冲信号是时间相关的，因此时间量化方法可以减少网络的计算复杂度。一种常见的方法是将时间分成离散的时间步，并将每个时间步的输入和输出量化为离散值，这说明时间量化越精细，则模拟结果越准确，但会造成更多的计算成本。

（4） 网络结构量化：对 SNN 的网络结构进行量化，主要是为了降低网络的计算复杂度。例如，通过将网络拆分为较小的子网络可以减少网络中神经元和连接的数量，或通过对网络连接进行量化将大规模的连接权重分成多个小矩阵。

不过不管是机器学习型还是仿生型 SNN，在计算机上实现的都是经过阈值量化、时间量化后的模型，即通常实现的都是阈值固定、最小时间步长固定的神经元模型，关于后者，近期有研究表明[34]，最小时间步长常数能对 SNN 中的信息传输产生影响，因此选择一个合适的最小时间步长能提高 SNN 对数字识别的收敛速度，这是对在 SNN 训练阶段一种加速方案。至于权重量化和网络结构量化是两种能普遍的用于机器学习型 SNN 网络模型推理阶段进行加速的方法，如图 3-2 所示，而对于仿生型 SNN 的网络结构是按照生物神经网络进行连接的，即是不能被改变的，在网络模拟仿真时也就用不上网络结构量化的方法，即目前主要是对 SNN 内突触权重、神经元膜电位的量化研究。

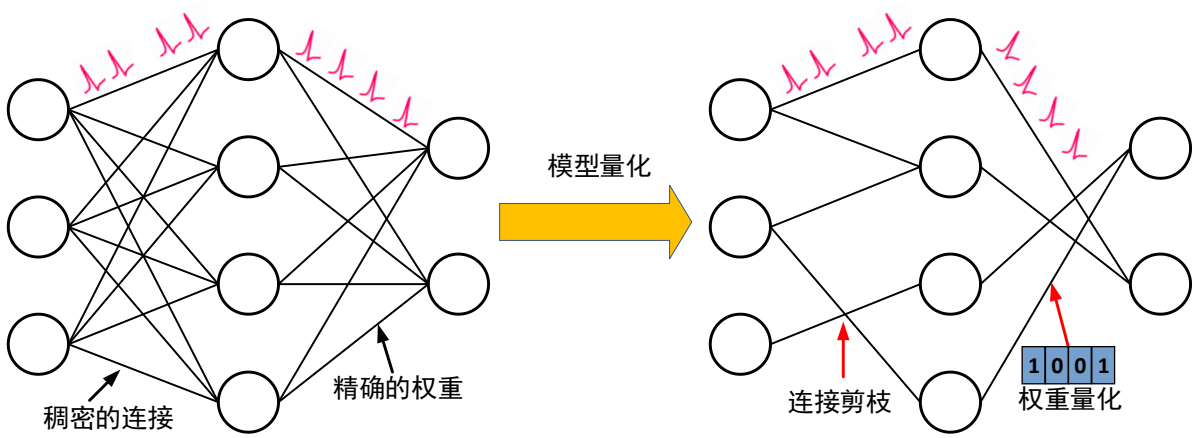

图 3-2 剪枝与权重量化示意图

3.1.3 SNN 位宽压缩

不管是对机器学习型 SNN 还是仿生型 SNN，位宽压缩都是一种重要的技术，它可以大幅度降低存储和计算的成本，当然如果针对具体的 SNN 类型不同，所需要压缩位宽的具体对象也就会不同：

（1）对于机器学习型 SNN，其中的激活值和权重通常是双精度的浮点值，这会带来大量的存储和计算成本，而位宽压缩可以实现把两者压缩到几比特，这不仅有利于加速模型的训练和推理，还有利于把 SNN 存储和部署到各种实际的设备中，包括嵌入式设备和移动设备等，这些设备通常具有有限的存储和计算资源。

（2）对于仿生型 SNN 来说，除了权重需要压缩位宽之外，膜电位更需要进行位宽压缩，这是因为膜电位的计算是类脑计算模拟当中非常重要的一部分，它模拟了神经元的电压变化过程，对于模型的精度和计算速度都具有重要的影响。而膜电位的计算是使用时间积分器实现，这就需要进行高精度的浮点数计算，该计算会带来大量的存储开销与计算成本，会是实际平台无法提供满足模型仿真模拟计算的需求。将膜电位位宽压缩，不仅能大幅度降低对存储和计算的需求、提高模型仿真的计算效率，还能降低能耗。

需要值得注意的是，位宽压缩必然会带来一定的精度损失，位宽的降低会导致被量化数值误差的增加，进而影响模型的精度和性能。因此，在对 SNN 进行压缩位宽时，需要根据具体的应用场景和模型要求，平衡模型的大小、计算速度和精度等因素，选择合适的位宽与压缩方法，以达到最优的效果。

3.2 位宽压缩方法研究

对于 SNN 网络来说，神经元膜电位的电压变化是神经元计算和信息处理的基础，同时也是神经元和神经元之间进行通信和交互的重要方式。在类脑计算中，模拟膜电位的变化过程可以模拟神经元的计算和信息处理过程，从而实现类似于生物神经网络的计算模式，这种计算模式具有高度的并行性、鲁棒性和灵活性，可以实现非常复杂的计算任务。另外，膜电位的电压变化还可以用于学习和记忆，这为类脑计算的研究和应用提供了重要的理论和实践基础，因此神经元膜电位的量化显得尤为重要。不同的神经元除了都需要对膜电位进行位宽压缩外，还需要对权重以及各自的不同参数进行位宽压缩。

3.2.1 数值误差分析方法

目前对于 SNN 的数值误差分析还没有一个统一的标准，而常见的分析误差的方法有以下 3 种：绝对误差和平均绝对误差、相对误差和平均相对误差、均方根误差。

（1）绝对误差是值测量值与真实值之间的绝对值：

$$\Delta = |x - a| \tag{3-6}$$

其中 Δ 表示绝对误差，x 为测量值，a 为真值，在这里 x 表示为压缩后表示的参数数值，a 为压缩前表示的参数数值，绝对误差更关注两者间的绝对差异。而平均绝对误差是在所有的绝对误差求和之后，除以数量 n，是反应两组数值之间的绝对差异。

（2）相对误差为绝对误差与真值的比值，一般用百分比表示：

$$\varepsilon_{err} = \Delta / a = (x - a)/a \tag{3-7}$$

其中 ε_{err} 表示相对误差，对于单个样本数据的误差计算来说，相比于绝对误差，相对误差更能反应压缩后数值的真实性。而对于一组数据来说，平均相对误差更关注压缩后数值的相对大小，即会使压缩后的一组值更接近真值。

（3）均方根误差是将每个测量值的误差平方后求和，然后取平均值并开方:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - a_i)^2}{n}} \tag{3-8}$$

从(3-8)公式中可以发现，均方根误差相比于平均绝对误差会对过大或者过小的数值更敏感，而把它当作损失函数的话，会让一组数值与压缩后数值的误差最小，因为它也是更关注两组数据之间误差的绝对大小。

3.2.2 位宽压缩方法

因为常规压缩位宽的量化方法会对小数点后的数值做一个到整数区间的映射，最后都量化为整数值，因此这中间会丢失很多原先在小数部分能表示但在整数部分无法表示或者只能用临近值表示，将会导致数值并不精确。而某些参数（如膜电位的电压）对于数值精确度却有着很高的要求，如果继续使用常规的量化方法，将会导致很多误差过大，会使光滑的数值变化曲线变成阶梯式的数值变化图，因此本文采用一种保留小数部分的位宽压缩方法如图 3-3 所示。图中(a)表示原始表示参数数值的 double 数据类型，共占 64 位，其中 12 位为指数位（包括 1 位符号位）、52 位尾数部分，而压缩到右边共 W 位，其中整数部分占 I 位（包括 1 位符号位），小数部分占 W-I 位。

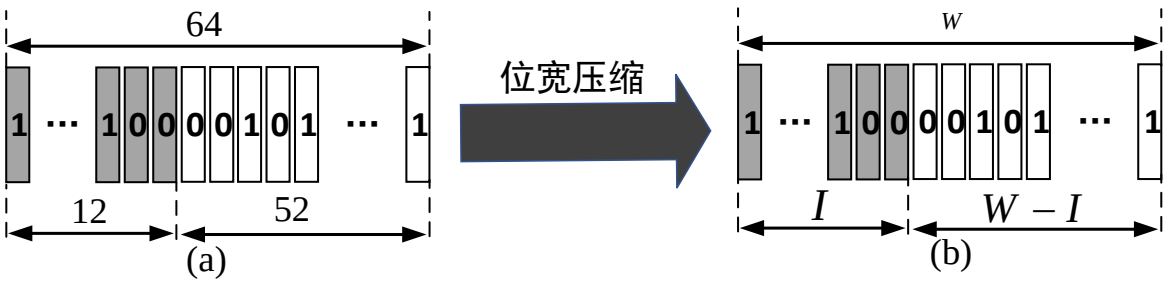

图 3-3 位宽压缩示意图

因此压缩后参数的数值误差可以由公式(3-9)表示，令 $L = W - I$：

$$Loss(I, L) = \varepsilon_{\min(I,L)} = \min\left(\frac{\left|x_L^I - a\right|}{|a|}\right) \tag{3-9}$$

其中 $I \in [1,12]$，$L \in [0,52]$，尽管这里存在两个变量，但分析数值本身可以得出一个结论，那就是整数部分减少一个位宽，对数值本身的相对误差影响远大于小数部分减少一个位宽，因此首先需要对整数部分的位宽进行量化。

（1）整数部分量化

对于神经元一些在整个仿真过程中始终保持不变的参数，即只在初始化阶段给予赋值的，而且一般这些参数都具有一定的生物含义，因此值是确定的，可以直接对整数部分位宽进行设计，需要满足 $2^{I-1} > a$，其中减一是减去一位的符号位。其次是随着仿真

过程会改变数值的参数，如权重、膜电位等，这些值并不能直接进行简单得量化，需要做一些预处理，即获取这些参数值的取值范围。

一种是能直接确认取值范围的，比如部分具有生物意义的参数其值的变化范围往往是有限的，比如生物膜电位的取值范围为-150mv 到 60mv 之间，那么膜电位的整数部分位宽可以直接设计为$I=9$，因为 9 位整数位（带 1 位符号）能表示-256mv 到 256mv 之间的整数，包含了膜电位的取值范围。另外，对于在机器学习型 SNN 里面，在训练完毕之后，模型的权重是固定的，因此能通过实验统计这类模型内权重的取值范围，然后对其位宽进行压缩。

另一种是不能直接确定取值范围的，比如仿生性 SNN 内后神经元这边接收到的权重，这个值不仅跟前神经元的突触权重有关系还跟网络的连接情况、前神经元是否激发有关系。假设前神经元数量为n，前后神经元的连接概率为α，那么对于突触的权重w_s来说，如果是静态突触，那这个值本身是不会发生改变的，那后神经元接收到权重的取值范围为：$[-n\times\alpha\times|w_s|, n\times\alpha\times|w_s|]$，而对于 STDP 等具备学习、记忆功能的突触，则并不是所有的网络都能进行权重的位宽压缩，因为在某些极端情况下，突触权值可能会变得极大或者极小，这导致取值范围变得过大。如果这个 SNN 网络是平衡的，那么对于这部分还是可以进行量化的，这里的平衡主要值得是整个 SNN 网络脉冲激发频率是较为稳定的，假设脉冲激发频率稳定为η，因为对于 STDP 等突触在脉冲激发频率稳定的情况下是会在w_s值得左右来回稳定得变化，假设变化值为d，那么后神经元接收到权重得取值范围为$[-n\times\alpha\times(|w_s|+|d|), n\times\alpha\times(|w_s|+|d|)]$。对于另一种传统的确定取值范围的方法，实验统计法，这个方法也能确定这两种不同突触的取值范围，这种方法需要运行一定时间、脉冲激发频率达到稳定的 SNN 网络，统计这段时间内权重的取值范围。

（2）小数部分量化

对于小数部分的位宽选取就涉及到 SNN 模型精度的选取问题。分析公式(3-9)，当整数部分位宽I确定下来之后，随着小数部分位宽L值得变大，数值的误差损失函数$Loss(I,L)$会不断变小（最后能收敛），因为2^{-L}数值更小，意味着能精确表示的值越多，模型的精度就越高。至于具体的位宽选择要看 SNN 网络对于精度的敏感度，假设$Loss(I,L)$在L_d位的时候已经达到10^{-5}，对于膜电位数值精度要求不高的模型，可以选择L_d位或者比L_d小一两位的位宽以减少资源的开销，而对于精度要求更好的模型，可以选择比L_d大几位的位宽。

3.3 NEST

3.3.1 NEST 仿真机制

NEST 仿真器主要分为创建和仿真阶段。在创建阶段，主要分为神经元、突触等 SNN 网络中成员的生成，以及成员间的连接的建立；在模拟仿真阶段，分别为准备、主循环、结束三个阶段，仿真过程中的时间开销主要都在主循环阶段，此阶段会不断的重复执行以下三个模块：分发事件、更新神经元节点、收集节点，一直到模拟仿真时间结束，如

图 3-4 所示，图中 t_{sim} 表示仿真器实际模拟时间，T_{set} 表示设定的模拟时间。

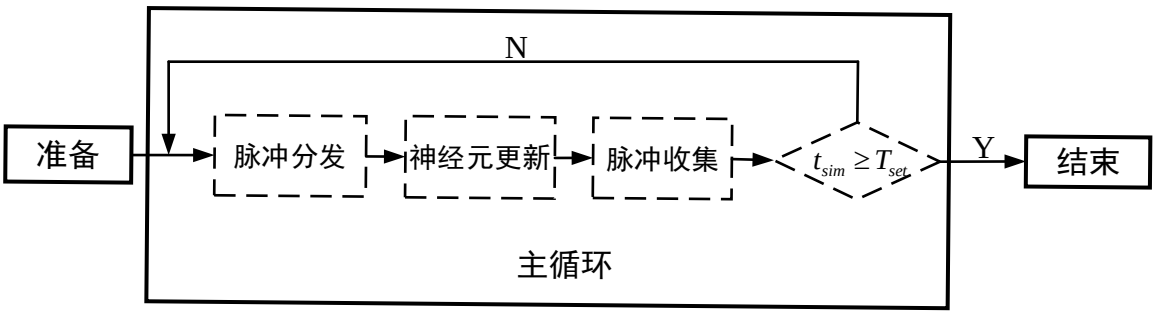

图 3-4 NEST 仿真流程图

在主循环阶段，SNN 模拟可以被看作是单个神经元通过交换尖峰相互作用的模拟的集合。通常，通过在每一步更新神经元的内部状态，以一种时间步的方式模拟每个神经元，在一个时间步长的神经元状态变化可能触发零个或更多的尖峰，然后发送到邻近的神经元，由于峰值必须在目标神经元的未来状态更新中被考虑，因此一个神经元的状态只能在它接收到所有时间戳更小的峰值时才会被更新。从这个 SNN 模拟的过程中，可以看到两个主要的操作是神经元更新和脉冲传递，它们共同构成了 SNN 模拟器的主要组件：神经元更新组件接收当前状态和传入的脉冲作为输入，并为每个神经元执行一个模拟步骤。这个计算步骤的输出由神经元的新状态和任何新发出的尖峰组成，尖峰传递组件负责在网络中的神经元之间传递尖峰信号，如果在同一进程中模拟连接的神经元，则交付可以在本地进行，也可以通过进程间通信或物理网络进行远程的信息交互。

SNN 模拟的一个重要方面是尖峰传输的延迟，反映了突触延迟，即信号到达并影响目标神经元所需要的时间步长。NEST 模拟器在同步方案中利用了突触延迟这一特性，以避免在每个时间步上传输所有新的峰值，相反，将仿真时间以神经网络中时延最小的大小划分为超步，超步中产生的所有尖峰只在超步（大小与脉冲时延相等）结束时发送(如图 3-5 所示)。该方法保证了所有神经元在一定时间内接收到传入的尖峰，同时降低了同步频率，这不仅较少了通信次数以减少通信时间，还降低了神经元之间同步的时间开销。另外，超级步骤还会使同步点之间的计算量更大，会使计算更密集，这有利于在 FPGA 上进行加速计算。

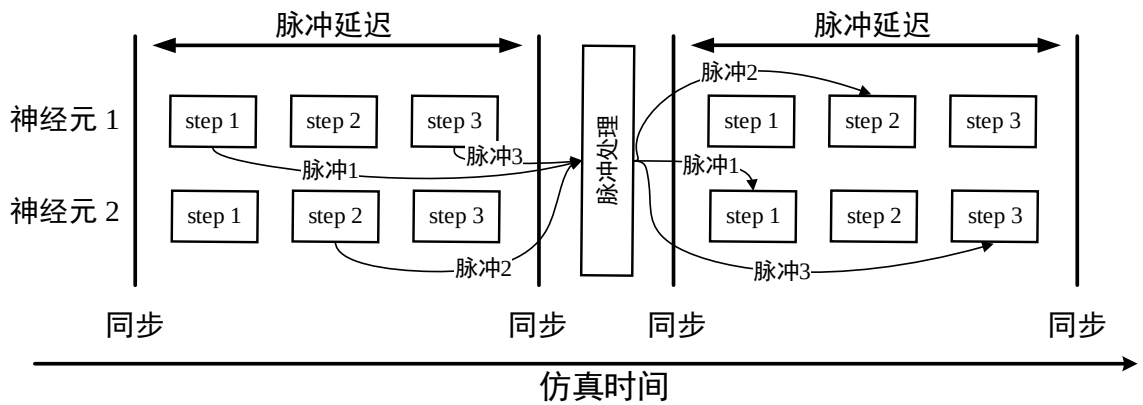

图 3-5 仿真步骤

3.3.2 NEST 仿真器负载量化分析

通过上一小节对 NEST 仿真机制的分析，能大致掌握 NEST 仿真的基本工作流程，然而对 NEST 实际运行时的内存开销、计算时间开销、通信时间开销等特性还不够清晰，接下来将对这些特性进行深入研究。在对 NEST 进行分析研究时，需要对一些参数值进行假设以便有一个更好的定量分析，假设原始的基于 CPU 的 NEST 仿真器能以分布式的方式在 P 个 MPI 进程中执行，并且在每个 MPI 进程中又能执行 T 个 OpenMP 线程，并假设整体网络模型包含 N 个神经元，突触总数量 S，仿真时间为 T_{set}。值得注意的是，这里的 N 个神经元并不一定都是相同类型的，即这些神经元的内存、计算时间开销可能都不同，还有每个神经元的突触数量也并不一定就相同，以下将对 NEST 仿真器的三个部分内存开销、计算时间开销、通信时间开销量化并建立模型。

（1） NEST 内存负载模型

因为实际内存分配的最小单位是进程，一个进程内所有线程共享被分配的内存，因此第 k 个进程的总内存开销为 M^k，第 k 个进程的神经元数量为 N^k，M 可以由公式(3-10)计算得到：

$$M^k = \sum_i M_n{}^i + \sum_i \sum_j M_s{}^{ji} \tag{3-10}$$

其中 $i \in [1, N^k]$、$j \in [1, S^i]$、$k \in [1, P]$，$M_n{}^i$ 表示该进程内第 i 个神经元的内存开销，S^i 表示该进程内第 i 个神经元的拥有的突触数量，$M_s{}^{ji}$ 为该进程内第 i 个神经元的第 j 个突触的内存消耗。又由于计算平台要维持 NEST 运行本身会占据一定的内存，假设所需的基本内存为 M_r，所以整体总的开销可以表示为以下(3-11)公式，：

$$M(P,N) = \sum_k M^k + M_r \tag{3-11}$$

把公式(3-10)带入公式(3-11)得到公式(3-12):

$$M(P,N) = \sum_k \sum_i M_n{}^{ik} + \sum_k \sum_i \sum_j M_s{}^{jik} + M_r \tag{3-12}$$

这里隐藏了一个等式条件，那就是 P 个进程的神经元的总数还是 N，突触总数还是 S，即 $N = \sum_k N^k$，$S = \sum_k \sum_i S^{ik}$，那么整体的内存负载模型公式(3-12)可以进一步化简为公式(3-13)，这里的变量 i, j 取值范围发生了改变：$i \in [1, N]$、$j \in [1, S]$。

$$M(P,N) = \sum_i M_n^i + \sum_j M_s^j + M_r \tag{3-13}$$

通常对于同一类型的神经元和突触来说，内存开销都是一致的，有的网络往往只有一种神经元模型和一种突触模型，那么(3-13)等式可以进一步的简化为等式(3-14)，其中 M_n 为单个神经元内存开销和 M_s 为单个突触内存开销。

$$M(P,N) = N \times M_n + S \times M_s + M_r \tag{3-14}$$

（2） NEST 计算时间负载模型

NEST 仿真器的计算时间开销模型主要由神经元更新的时间和突触更新的时间，总

的时间开销$T_{\cos t}$可以由所有神经元更新完成时间T_n加上所有突触更新完成时间$T_{\cos t}$，总的计算模型如公式(3-15)所示：

$$T_{\cos t} = T_n + T_s = \sum_i \sum_w t_n^{wi} + \sum_j \sum_m t_s^{mj} \tag{3-15}$$

其中第i个神经元第w次更新时间为t_n^{wi}，第j个突触的第m次的更新时间为t_s^{mj}，其中$i \in [1, N]$、$j \in [1, S]$。至于w的取值范围的上限为神经元更新的次数Num_{neuron}，可以由公式(3-16)确定，即仿真设定时间除以最小时间来确定：

$$Num_{neuron} = \frac{T_{set}}{dt} \tag{3-16}$$

而m取值范围的上限就是突触更新次数$Num_{synapse}$，因为突触更新是事件触发型，因此它跟神经元得脉冲激发频率有直接关系，假设该频率为α，那么$Num_{synapse} = \alpha \times T_{set}$。

单次神经元的更新时间t_n还可以分成不应期的更新时间开销(t_{n_unref})、响应期无脉冲的更新时间开销(t_{n_ref})和响应期产生脉冲的更新时间(t_{n_spike})三种，通常情况下t_{n_unref}与t_{spike}都为0，但t_{spike}会在发射脉冲时产生计算时间，t_{n_unref}会在产生脉冲之后的一小段时间内产生计算时间，即公式(3-15)可以进一步细分为：

$$T_{\cos t} = \sum_i \sum_w (t_{n\_ref}^{wi} + t_{n\_unref}^{wi} + t_{n\_spike}^{wi}) + \sum_j \sum_m t_s^{mj} \tag{3-17}$$

上述这种分析计算方法是针对单进程的，如果使用多进程，那么显然公式(3-17)就需要改变了。NEST 为了尽可能的缩短分布式仿真的模拟时间，会在给各进程分配神经元的时候采用轮询的方式来尽可能得实现均匀分配，每个进程将会被分到大约数量相同的工作负载，即每个进程大约处理N/P个神经元、S/P个突触，则第k个进程的计算耗时$T_{\cos t}^k$可以为以下，其中$i \in [1, N/P]$、$j \in [1, N/S]$：

$$T_{\cos t}^k = \sum_i \sum_w (t_{n\_ref}^{wi} + t_{n\_unref}^{wi} + t_{n\_spike}^{wi}) + \sum_j \sum_m t_s^{mj} \tag{3-18}$$

最后所有进程总的计算时间开销$T_{\cos t}$取决于计算时耗最大的进程，如公式(3-19)所示：

$$T_{\cos t} = \max(T_{\cos t}^k) \ , \ k \in [1, P] \tag{3-19}$$

（3） NEST 通信时间负载模型

对于多进程的 SNN 网络仿真，除了计算耗时外，还有一定的通信耗时，而整个仿真过程中总的通信时耗与各进程通信的次数$Number_{com}$与每次的通信量有关系，可以由以下(3-20)等式获得：

$$\begin{cases} Number_{com} = \dfrac{T_{set}}{delay} \\ Num_{com}^f = delay \times Num_{synpase}^f \times Data_s \end{cases} \tag{3-20}$$

其中Num_{com}^f第f次的通信量，$Num_{synpase}^f$为第f次产生的脉冲数量，$Data_s$表示产生单次脉冲所引起的数据量，又假设带宽为C，t_{com}^f表示第f次的通信时间，那么总的通信时间T_{com}为每一次通信时间的总和，$f \in [1, Number_{com}]$：

$$\begin{cases} t_{com}^{f} = \dfrac{Num_{com}^{f}}{C} \\ T_{com} = \sum_{f} t_{com}^{f} \end{cases} \tag{3-21}$$

3.4 神经元数值求解量化方法研究

膜电位等的数值更新将会直接影响脉冲神经网络的的仿真时间、精度以及脉冲激发频率等，而不同的数值求解方法、不同的时间步长将直接影响到膜电位等的更新，因此寻求一个能以更小的计算代价获得更高的仿真精度对脉冲神经网络性能的加速有较大的意义，因此接下来将围绕着数值求解方法展开相关的量化工作。有许多不同的方法可以用于求神经元模型膜电位更新方程的近似解，本节将讨论其中三种较为常用的方法：前向欧拉法、四阶龙格-库塔法和指数欧拉法，并将使用以下符号：在时间步 n 时，任何量 γ 的值将被表示为 y^n 。

3.4.1 神经元更新方程数值求解方法

假设存在这样一个方程：

$$\frac{d\xi}{dt} = \psi(\xi, t) \tag{3-22}$$

其中 $\psi(\xi,t)$ 是 ξ 和 t 的任意函数，接下来将对该方程的进行三种不同近似求解法。

（1）向前欧拉法（FE，Forward Euler Method）

近似方程解的最简单方法之一是前向欧拉法，假设知道在 t 时刻对应 ξ 的值，那么当时间增量 Δt 时，在 $t+\Delta t$ 时刻，ξ 的值可以通过公式(3-23)近似的获得：

$$\xi^{n+1} = \xi^{n} + \psi(\xi^{n}, t)\Delta t \tag{3-23}$$

该方法非常的简单，但只有一阶精度，这意味着它的误差与时间步长成正比。

（2）四阶龙格-库塔法（RK4，Rung-Kutta Method）

用四阶龙格-库塔法可以得到一个较好的解，该方法具有四阶精度[$error \propto (\Delta t)^4$]，类似的，$t+\Delta t$ 时刻，ξ 的值可以通过公式(3-24)近似的获得：

$$\xi^{n+1} = \xi^{n} + \frac{\Delta t}{6}(\zeta_1 + 2\zeta_2 + 2\zeta_3 + \zeta_4) \tag{3-24}$$

其中 $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ 由公式(3-25)获得：

$$\begin{cases} \zeta_1 = \psi(\xi^{n},\ t) \\ \zeta_2 = \psi(\xi^{n} + \dfrac{1}{2}\zeta_1\Delta t,\ t + \dfrac{1}{2}\Delta t) \\ \zeta_3 = \psi(\xi^{n} + \dfrac{1}{2}\zeta_2\Delta t,\ t + \dfrac{1}{2}\Delta t) \\ \zeta_4 = \psi(\xi^{n} + \zeta_3\Delta t,\ t + \Delta t) \end{cases} \tag{3-25}$$

这种方法很容易推广到耦合的多变量系统。从这些方程可以明显看出，该方法比前欧拉