密级：_____

浙江大学

硕 士 学 位 论 文

论文题目 面向达尔文 II 类脑计算芯片的仿真训练平台

作者姓名 _____

指导教师 _____

_____

学科(专业) 计算机技术

所在学院 计算机科学与技术学院

提交日期 2020.01.09

A Dissertation Submitted to Zhejiang University for the Degree of Master of Engineering

TITLE: The Simulation and Training Platform for the Neumorphic Hardware: Darwin II

Author: _____

Supervisor: _____

_____

Subject: Computer Technology

College: Computer science and Technology

Submitted Date: 2020.01.09

摘要

随着人工智能发展的高速发展，云计算、物联网、大数据等技术的持续突破，数据量和数据处理量都以指数倍增长，传统的存储器和中央处理器分离的冯诺伊曼框架的内存墙效应越来越成为一种约束，已经不能满足高效计算的需求。基于非冯诺伊曼架构的类脑计算成为了新的选择。类脑计算主要基于第三代神经网络—脉冲神经网络，用于模拟生物神经网络处理、存储和传递信息的方式，具有低功耗、高效等特点。

由于脉冲神经元的“内存和计算二合一”的特点，采用通用的加速硬件如 GPU 并不能发挥脉冲神经网络的高能效特点。因此，基于脉冲神经网络的类脑计算芯片应运而生，成为工业界和学术界的研究重点。然而，由于类脑计算的发展还处于初级阶段，尚未形成统一标准，缺乏统一的仿真训练平台，基于类脑计算芯片的高能效应用开发面临严峻挑战。

本文面向浙江大学自主研发的达尔文 II 类脑计算芯片，研发应用开发的仿真训练平台。从脉冲神经网络的构建、训练和仿真三方面出发，使得开发者能够在达尔文 II 类脑计算芯片上极简单地完成脉冲神经网络应用开发。在网络构建方面，本文提出了面向达尔文 II 类脑计算芯片的脉冲神经网络规范化描述方法，构建了软硬件协同仿真模型库，使神经网络模型的构建有了统一标准。在网络训练方面，同时支持脉冲神经网络的直接训练和间接训练，满足不同应用的训练方法需求。在网络仿真阶段，采用软件仿真与芯片层面仿真相结合的方法，使开发者能够更加准确和直观地评估应用效能的同时大幅度提高仿真速度。本平台避免了开发者直接面对达尔文 II 芯片复杂的硬件约束，对初学者非常友好，极大地降低脉冲神经网络应用开发的难度。此外，仿真训练平台支持标准化扩展接口，能够灵活集成第三方类脑计算芯片。

关键词　达尔文 II，脉冲神经网络，Darwin 神经元模型，网络描述规范，STDP，基于 SNN 的反向传播算法，仿真

Abstract

With the rapid development of artificial intelligence, cloud computing, the Internet of Things(IoT), big data and other technologies make great breakthroughs. The amount of data and data processing have exponentially doubled. Traditional Von Neumann framework is memory and CPU separated, and its memory wall effect is becoming a constraint. Brain-like computing with non-Von Neumann architecture has become a new choice. Brain-like computing is mainly based on the third generation of neural networks-spiking neural networks. This kind of network is mainly constructed by simulating how biological neural networks processing, storing and transmitting information, and has the characteristics of low power consumption and high efficiency.

Because of the spiking neurons has the feature of both memory and computing, the use of general-purpose acceleration hardware such as GPUs cannot take advantage of the energy-efficient characteristics of spiking neural networks. Therefore, brain-like computing chips based on spiking neural networks have emerged at the historic moment, and have become the focus of research in industry and academia. However, as the development of brain-like computing is still in its infancy, a unified standard has not yet been formed, and a lack of a unified simulation and training platform. The development of high-energy effects based on brain-like computing chips faces severe challenges.

This article is aimed at the Darwin II brain-like computing chip, which is independently developed by Zhejiang University, to design a simulation and training platform for application development. Starting from the aspects of construction, training and simulation of spiking neural network, developers can complete the application development of spiking neural network on Darwin II very simply. In terms of network construction, this paper proposes a standardized description method of spiking neural networks for Darwin II, and builds a software and hardware co-simulation model library, so that the neural network model construction has a unified standard. In the aspect of network training, both direct and indirect training

method of spiking neural networks are supported, to meet the needs of training method for different applications. In the network simulation phase, a combination of software simulation and chip-level simulation is used to enable developers to more accurately and intuitively evaluate the effectiveness of applications while greatly increasing the speed of simulation. This platform avoids developers directly facing the complex hardware constraints of Darwin II, and it is very friendly to beginners, which greatly reduces the difficulty of developing spiking neural network applications. In addition, this simulation and training platform supports standardized extension interfaces, which can flexibly integrate third-party brain computing chips.

Keywords Darwin II, Spiking Neural Network, Darwin Neuron Model, Network Description Specification, STDP, SNN-based Back Propagation Algorithm, Simulation

目录

图目录

表目录

第1章 绪论

1.1 类脑计算研究背景

目前，人工智能领域的两个重大瓶颈其实是存储器与中央处理器分离的“冯诺伊曼”架构导致的存储墙效应造成效率低下，和引领半导体发展的摩尔定律预计在未来数年内失效。随着人工智能技术的持续发展，图像视频处理、机器翻译、语音识别、数据挖掘、自然语言处理等智能化应用的高速发展，数据量以及数据处理量都以指数倍增长，传统的计算机设备的计算能力越来越不能满足高效计算的需求。针对这个问题，基于非冯诺伊曼架构的类脑计算将是更好的选择。

相比于“谷歌大脑”使用了 1.6 万个处理器，耗时数天来识别猫脸，“阿尔法狗”利用超过 170 个 GPU，超过 800 万核进行并行计算，下一盘棋的成本是 3000 美元[1]，由大量神经元仅通过突触连接在一起的人脑显得既“聪明”又“节能”。随着神经科学的研究进步，研究者们发现生物神经元是“内存和计算二合一”的单元，既有计算能力，又兼具存储功能，很好地解决了冯诺伊曼架构的存储墙问题。进一步的，科学家们发现人脑具有高容错、高并行以及低功耗等显著优点。因此，研究如何模拟人脑神经元处理信息机制的深度神经网络技术—类脑计算越来越受到世界各国的广泛关注。

早在上世纪 80 年代，就有了人类脑计划的基本概念，即利用计算机建立人类大脑的数据库和模型[2]。1997 年，首个人类脑计划由美国发起，将神经科学和信息科学相结合，进行人类大脑的行为研究以及信息学研究，于 2004 年结束。紧接着，奥巴马政府于 2013 年 4 月宣布开启新一轮美国脑计划—BRAIN(Brain Research through Advancing Innovative Neurotechnologies)[3]，即“创新神经技术脑研究计划”，该计划为期 10 年，旨在更全面深入地理解大脑的功能，预计花费数十亿美元。同年，欧盟委员会也宣布开展“人脑计划(Human Brain Project，HBP)”[4]，预计投入 10 亿欧元，由 87 个研究机构参与，该计划目的是研究大脑的工作机制，为医学进步提供科学和技术基础。我国脑计划的发展也不甘落后。在 2016

年我国先后印发《“十三五”国家科技创新规划》《“十三五”国家信息化规划的通知》，提出并部署了“脑科学与类脑研究”(Brain Science and Brain-Like Intelligence Technology)项目，包括以探索大脑秘密、攻克大脑疾病为导向的脑科学研究以及以建立和发展人工智能技术为导向的类脑研究。

类脑计算的最终目标是研究出新一代的非冯诺伊曼计算体系，兼具节能与高效等优点的通用计算系统。众所周知，人脑在模糊信息处理，如物体识别、自然语言处理、音视频理解等方面的能力远高于计算机。但是，随着人工智能、深度学习技术的发展，在 2015 年，微软亚洲研究院提出 resNet，使用残差学习，在大规模图片数据集 ImageNet 上的分类识别错误率为 4.94%，低于人类的 5.1%[5]；2017 年，DeepMind 团队研发的 AlphaGo[1]在围棋领域战胜人类排名第一的选手柯洁。从这些例子可以看出，电脑在自然语言处理、图像识别等特定领域已经可以与人脑相媲美。

1.2 类脑计算开发平台研究意义

随着计算技术的发展，类脑计算技术引起了人们的广泛关注[6]。如果人工神经网络是模拟人类神经系统的软件，那么用于神经形态计算的类脑计算芯片或者处理器就是一种在硬件中模拟神经细胞的硬件设备。类脑计算的最大优势是其低功耗的特性[7]。因此，它可以有效地应用于对能源效率要求较高的领域。在物联网环境中使用类脑计算芯片可以减少能耗并有效执行需要的人工智能的功能。

通常，类脑计算芯片使用脉冲神经网络(Spiking Neural Network, SNN)模型作为基础，该模型不同于典型的人工网络模型(Artificial Neural Nerwork, ANN)，例如卷积神经网络(Convolutional Neural Network, CNN)或者递归神经网络(Recurrent Neural Network, RNN)。最显著的区别在于，ANN 主要使用具有连续激活值和一组加权输入的理想化计算单元构建，这些单元通常被称为“神经元”。这些（非脉冲）神经元使用可微的非线性激活函数。然而，SNN 使用离散的脉冲来计算和传输信息，并且除了脉冲率外，脉冲时间也很重要。SNN 可以在计算中有效地利用时间编码信息[8]。这种信息处理方式也是事件驱动（event-based）的，

这意味着 SNN 不会进行过多计算，只有在记录到突发活动时，SNN 会产生更多的脉冲。Neil 等[9]还对 SNN 进行了专门的训练以减少近似推理的时间延迟。

人工神经网络的硬件加速工作已经有了长足的发展，包括基于 GPU 的并行计算框架和一些专门为人工神经网络设计研发的加速芯片，中科院计算所的寒武纪系列神经网络处理器[10]。这类芯片用于为特定算法或者特定网络加速时效果较好，但是并不能为具有更高生物相似度的脉冲神经网络加速。目前，有许多可用于类脑计算的芯片和处理器[11]，如 IBM 研发的 TrueNorth 芯片、英特尔的 Loihi 等，这些处理器能够很好地实现脉冲神经网络应用。

T．Bekolay 的等[12]提出了一种名为 Nengo 的基于 Python 的软件工具，可用于构建和仿真大规模的脉冲神经网络应用。Nengo 具有支持各种脉冲神经网络模型的优势，但不支持在类脑计算芯片上对神经网络应用进行调试和运行。Boseon 等[13]提出了一种服务体系架构 NAAL(Neuromorphic Architecture Abstract Layer)，该服务架构通过虚拟化具有不同特征的各种神经形态处理器来实现脉冲神经网络应用，但是目前还处于研究阶段。

但是，由于每个处理器的底层硬件架构不同，对应用的约束不同，在神经元的最大数量，支持的脉冲模型等方面也具有不同的特性。因此，需要针对不同的芯片架构，设计开发对应的软件平台，以充分发挥芯片架构的计算潜力，同时，也有必要实现一种能够支持各种类脑计算芯片的体系结构，用于实现对不同脉冲神经网络模型的仿真或者调试。

1.3 面向类脑计算芯片的开发平台研究现状

类脑计算芯片是一种新的人工智能芯片架构，基于微电子技术和新型神经形态器件，将计算和存储相结合，能够大幅度提升计算性能、降低功耗[14]。目前已有部分基于脉冲神经网络的专用解决方案，包括一些基于脑启发的数字或者模拟电子系统[15]。虽然这些体系结构对于探索大规模神经系统模型的计算特性有很大用处，但是构建低功耗、高效的通用类脑计算芯片依旧是研究热点。目前业界比较成熟的类脑计算芯片包括 IBM 的 TrueNorth 芯片[16]，英特尔的 Loihi 芯片[17]，

清华大学的天机芯片[18]，浙江大学和杭州电子科技大学共同研发的“达尔文”系列芯片[19]等。

IBM 的 TrueNorth 发布于 2014 年，是首个基于脉冲神经网络的“自适应可扩展塑性电子神经形态系统”（SyNAPSE）芯片。TrueNorth 整个系统可以分成两部分：一是 TrueNorth 处理器的硬件，二是 TrueNorth 的软件生态系统。该处理器是功耗约 70mW 可重配置的硅芯片，基本单位是神经核，单片芯片共由 4096 个并行分布式神经核组成，每个神经核支持 256 个神经元的实现，共支持超过 100 万神经元以及 2.56 亿个突触。同时芯片支持扩展，目前共有只包含单片芯片的 NS1e 和由 4×4 芯片阵列组成的 NS16e 两个系统。TrueNorth 提出了一种端到端的软件生态系统，包括神经网络的定义，构建和推理。编程范式包括四个部分[20]：一是 Corelet，用于表示神经突触核心网络，里面封装了除输入输出外的所有详细信息，避免了开发者直接面对底层硬件，如图 1.1 所示；二是一种用于创建、合成和分解 Corelet 的面向对象的 Corelet 语言，以适应 TrueNorth 大规模并行分布的特点[21]；三是一个不断增长的 Corelet 存储库，开发者可以从中组成新的 corelet 并添加回库中；四是端到端的 Corelet 编程环境。但是，TrueNorth 并不支持片上学习。

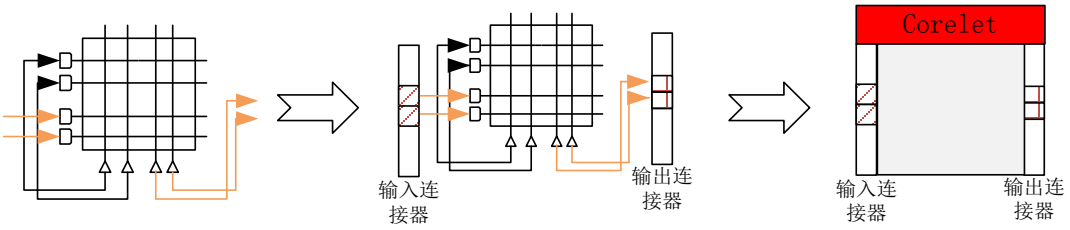

图 1.1 TrueNorth Corelet 示意图

Loihi 由英特尔在 2017 年发布，共有 128 个计算核，每个核支持 1024 个神经元，整个芯片共有超过 13 万个神经元和 1.3 亿个突触。从芯片支持的神经元和突触数量来看，Loihi 相比于 TrueNorth 的规模要小得多。但是 Loihi 支持片上学习，每个核都有一个可编程的学习引擎，支持 STDP 学习规则，并且能够实现异步片上网络。同样，Loihi 也提供了编程工具链，包括基于 python 的脉冲神经网络定义的 API、编译器和构建运行脉冲神经网络的运行库[22]，其架构如图 1.2 所示。

通过 Loihi Python API 定义脉冲神经网络结构，使用编译器（Compiler）有效地将脉冲神经网络映射到 Loihi 的核心上，该映射使用贪心算法（即尽量少地使用核），将输入进行编码，指定训练算法之后即可以开始训练网络。

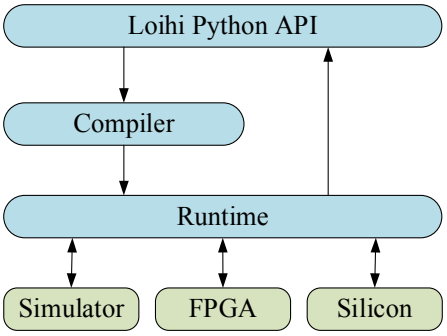

图 1.2 Loihi 编程工具链

2019 年 8 月 1 日，清华大学研发的“天机”芯片登上《Nature》杂志封面。天机芯片采用多核架构，由 156 个功能核（FCore）组成，包含约 2 万个神经元和 1 千万个突触。天机芯片是一款跨范式的计算芯片，最大的创新点在于其同时支持多种神经网络模型，包括脉冲神经网络和传统的人工神经网络（如 CNN，RNN 等），支持机器学习算法和类脑计算[18]。“天机”芯片同样具有软件工具链，能够支持从机器学习编程平台到芯片的自动映射和编译。该软件同时支持在 ANN 和 SNN 不同模式下的应用，使用 FCore 作为 ANN 和 SNN 之间的转换器，通过 FCore 实现不同模式的配置：每个 FCore 包括轴突、树突、胞体和路由器，当轴突和胞体配置成相同的操作模式（单纯 ANN 或者 SNN）时，FCore 为单范式 FCore，以纯 ANN 或者纯 SNN 的模式工作；当轴突和胞体被配置成不同的工作模式时，FCore 为混合 FCore，用于处理 ANN 输入并触发 SNN 输出或者处理 SNN 输入并生成 ANN 输出。

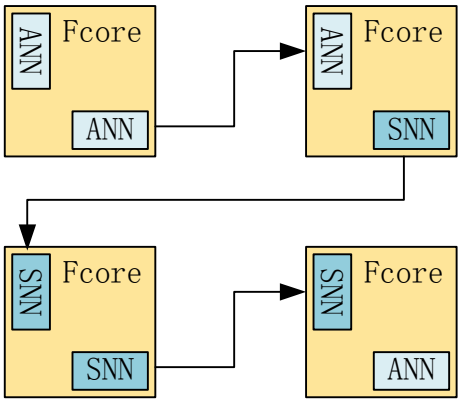

图 1.3 FCore 的工作范式

同时，受到计算机系统层次结构的启发，天机软件工具链包含与主机类似的层次，包括统一的编程抽象和自动编译器的映射。自动编译器可以将一个已训练的模型转化成为一个满足天机硬件约束的等价网络，从而实现将应用程序与目标硬件解耦。

1.4 本文目标

TrueNorth 使用面向对象的开发语言 Corelet 来避免开发者直接面对硬件，同时，支持开发者使用层次化递进的方式来进行脉冲神经网络应用的设计与开发，但是开发者仍然需要考虑到相关的硬件约束，并且 TrueNorth 并不支持在线学习，仅能支持神经网络的前向推理。Loihi 具有一个完整的开发工具链，包括脉冲神经网络的模拟、在线训练等开发流程，但是整个芯片的规模较小，对较大的应用来说需要使用芯片阵列完成，同样的，在开发过程中也需要考虑到其的硬件约束。清华大学的天机芯片首次将脉冲神经网络和传统人工神经网络这两种不同的计算模式相融合，对于单纯的 ANN 或者 SNN 应用，这是一个很好的平台，但是对于 ANN 和 SNN 的融合应用，这也极大地增加了应用开发的难度，开发者需要考虑到 ANN 和 SNN 之间不同模态信息的转换。

同时，这些所有的开发应用软件平台都是针对特定的芯片开发的，并不是一个通用的软件平台。对于达尔文 II 类脑计算芯片，由于其特定的硬件设计及约束，目前也并没有合适的软件平台解决方案来实现脉冲神经网络的构建、训练、仿真

等任务。因此，本文的主要目标就是开发一套面向达尔文 II 类脑计算芯片的仿真训练平台，包括脉冲神经网络的描述和构建（神经元模型选择、网络描述）、训练和仿真（训练算法选择、软硬件协同仿真）。

本文开发平台的输入为开发者自定义的一个脉冲神经网络，可以是训练完成的网络，也可以是未经过训练的网络，平台会自动检查该网络是否满足达尔文 II 芯片的硬件约束条件。对于已满足硬件约束的网络，开发者可选择训练方式对其训练，之后使用软件仿真器对网络功能进行仿真，软件仿真结束后开发平台会进行网络到芯片的自动化映射配置并进行芯片层面的仿真，这样极大地减少了开发者需要考虑的硬件条件，降低了开发者的应用开发难度。

1.5 论文组织结构

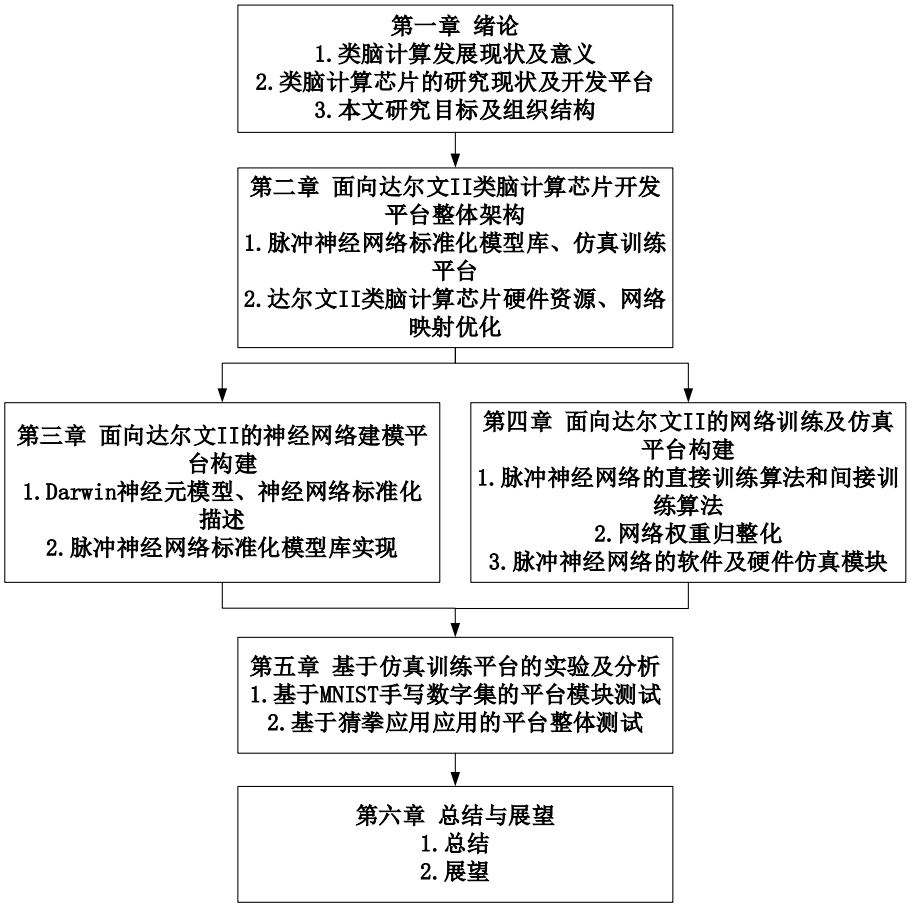

图 1.4 论文组织结构

如图 1.4 所示为本文组织结构。本文第一章主要介绍了基于脉冲神经网络的类脑计算的研究背景及意义，总结了目前类脑计算芯片的发展现状以及各自的开发平台和应用开发工具。同时，提出了本文的研究目标，为新一代的达尔文 II 类脑计算芯片设计一个对应的仿真训练平台，使开发者可以不直接面对硬件约束进行类脑计算相关应用的开发，降低开发者的开发难度。

第二章首先对仿真训练平台的总体框架进行介绍，然后对平台的每个功能模块进行具体说明，包括脉冲神经网络标准化模型库和仿真训练模块，最后提出了脉冲神经网络到达尔文 II 类脑计算芯片的映射及优化方法。

第三章和第四章为第二章的具体展开。第三章主要实现面向达尔文 II 芯片的脉冲神经网络建模平台的构建，提出了 Darwin 神经元模型和神经网络的描述规范，最后对脉冲神经网络标准化模型库的实现进行了说明。第四章主要实现仿真训练平台中的训练和仿真平台的构建，对仿真训练平台支持的训练算法进行介绍和说明，对仿真模块的实现进行说明。

第五章主要进行基于仿真训练平台的实验，验证平台的可用性。通过手写数字识别对平台分模块进行验证。通过猜拳应用对整个软件平台进行全过程说明。

第六章对全文进行总结，并展望该开发平台的未来发展方向。

1.6 本章小结

本章主要介绍了基于脉冲神经网络的类脑计算的研究背景及意义，总结了目前类脑计算芯片的发展现状以及各自的开发平台和应用开发工具。同时，提出了本文的目标，为新一代的达尔文 II 类脑计算芯片设计一个对应的仿真训练平台，最后对文章结构进行概述。

第2章 面向达尔文 II 类脑计算芯片的开发平台整体架构

2.1 平台整体架构

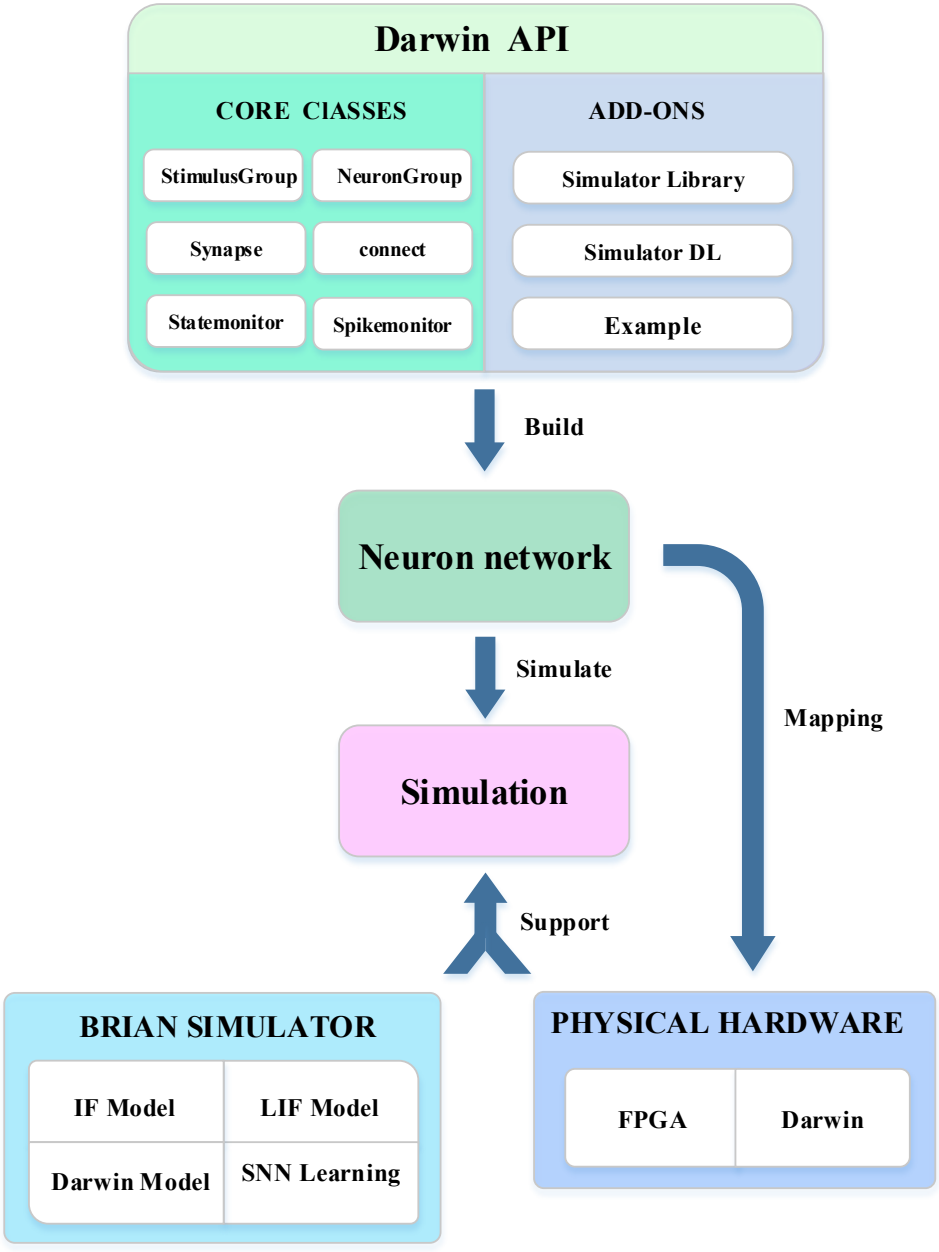

图 2.1 平台总体框架

达尔文芯片II的仿真训练平台可实现以下功能：一是脉冲神经网络的构建，包括神经网络的结构描述、神经元模型选择等；二是脉冲神经网络的仿真，包括脉冲源的产生、神经网络的软件仿真以及在达尔文II芯片上的仿真等；三是脉冲神经网络的训练，包括直接训练和间接训练，直接训练是指使用脉冲神经网络的学习算法对网络进行训练，间接训练是指将已经训练好的ANN转化成为脉冲神经网络。达尔文芯片II的仿真训练平台总体架构如图2.1所示，整个平台可分为四部分：

A．脉冲神经网络标准化模型库（DARWIN API）

脉冲神经网络的标准化模型库既支持达尔文II类脑计算芯片，也可以支持其他标准。在本文中，主要指达尔文API，由两部分组成，一是仿真平台的核心类，主要包括StimulusGroup、NeuronGroup、Synapse、connect、Statemonitor、Spikemonitor等与脉冲神经网络构建相关的模块。二是扩展和模型部分ADD-ONS，主要包括与核心类相关的基础库simulator library；与深度学习相关的simulator DL，其中包含ANN的训练、ANN转SNN等脉冲神经网络间接训练模块；以及仿真平台开发示例simulator example等。

B．仿真训练平台

脉冲神经网络的软件仿真以及直接训练部分是基于brain2的，仿真平台支持多种脉冲神经元模型，包括IF（Integrate and Fire）模型、LIF（Leaky Integrate and Fire）模型以及自定义的基于达尔文II芯片的Darwin神经元模型。同时，仿真平台也支持多种脉冲神经网络的学习算法，与传统人工神经网络类似，包括监督学习算法和非监督学习算法。

C．硬件资源（PHYSICAL HARDWARE）

平台的物理实现包括用于芯片和计算机通信的FPGA以及主芯片达尔文II。FPGA用于仿真训练的加速。达尔文II芯片用于实际脉冲神经网络应用的开发调试与运行加速。

D．网络优化映射平台（Mapping）

网络映射模块主要实现将开发者设计好的脉冲神经网络映射到芯片上的功

能，能够自动产生满足芯片约束的配置文件。整个mapping的目标是尽量使脉冲神经网络在芯片上低功耗、高效地运行。

平台的工作流程为：首先使用 Darwin API 的核心类以及相关库建立神经网络，若网络需要训练，则选择合适的训练方式，若不需要，则可以直接进行网络仿真，可以在软件仿真器Brain2上进行。同时，也可以使用mapping模块对神经网络的配置信息进行编译，将其自动化映射到达尔文II芯片上运行或者进行芯片层面的仿真。本文的工作主要针对面向达尔文II芯片的神经网络的构建、训练、仿真等方面展开。

2.2 脉冲神经网络标准化模型库

在本文中，脉冲神经网络标准化模型库主要由达尔文API构成。达尔文API主要包括两方面的内容，一是仿真训练平台核心类（SIMULATOR CORE CLASSES），二是扩展和模型部分（ADD-ONS）。

仿真训练平台核心类，主要包括 StimulusGroup、NeuronGroup、Synapse、connect、Statemonitor和Spikemonitor等模块。StimulusGroup主要用于产生脉冲神经网络的输入激励，将网络的输入数据编码成为一系列的由离散的脉冲时间点组成的脉冲时间序列。NeuronGroup是脉冲神经元类，主要包括达尔文II芯片支持的两种脉冲神经元模型。Synapse和connect主要用于突触的建模和连接的构建，脉冲神经元之间的连接称为突触，包括兴奋型突触和抑制型突触。本文设计了Statemonitor 和 Spikemonitor 两个记录器来记录脉冲神经网络运行过程中相关变量的状态，Statemonitor用于记录神经元的状态，包括膜电位以及状态变量的值，Spikemonitor用于记录脉冲信息，包括脉冲发放时间等。

扩展和模型部分主要用于支持仿真训练平台核心类的实现和扩展，包括simulator library、simulator DL和simulator example等三个模块。simulator library中主要包括与仿真训练平台核心类相关的基础库，包括神经元模型的实现、脉冲输入编码方法的实现。达尔文神经元模型共有两种，一种是单向衰减的神经元模型，另一种是双向衰减的神经元模型。脉冲神经网络的编码方式主要有两种，一

种是时序编码，另一种是频率编码。Simulator DL 中主要是与深度学习相关的模块以及脉冲神经网络的间接训练算法的实现。深度学习相关模块主要是封装了 tensorflow 开源框架中的人工神经网络的构建和训练方法，脉冲神经网络的间接训练方法主要是将训练好的人工神经网络转化成为功能相同的脉冲神经网络。

2.3 仿真训练平台

仿真训练平台的实现模块为 BRAIN BACKED，主要用于脉冲神经网络的直接训练和软件层面仿真。其中包括了脉冲神经网络的构建、脉冲神经元的软件模型、脉冲神经网络的直接训练以及脉冲神经网络的软件仿真。

BRAIN BACKED 主要是基于脉冲神经网络模拟器 Brain2 实现，支持脉冲神经元和突触动态特性的自定义。在 Brain2 中，本文主要实现了 LIF（Leaky Integrate-and-Fire）模型、IF（Integrate-and-Fire）模型和 Darwin 模型。同时，该平台也支持脉冲神经网络的构建，构建方法包括两种：一是开发者直接传入脉冲神经网络的结构文件，通过指定神经元模型和网络的超参数实现网络的构建；二是直接在代码编辑器中编程实现网络构建。

脉冲神经网络的直接训练方法是直接训一个脉冲神经网络，而不是间接训练方法中的转化方法。这种方法是通过神经元产生脉冲的情况，直接对神经元之间的连接权重做出改变。包括监督学习算法和非监督学习算法。

网络的仿真主要包括两方面的仿真：一是软件平台的仿真，二是芯片层面的仿真。软件平台的仿真主要是对网络进行功能仿真，这样能够使开发者对自己构建的网络的功能有一个更直观的认识。芯片层面的仿真主要是通过仿真训练这一软件平台将层次化的脉冲神经网络映射到达尔文 II 芯片上，再选择合适的脉冲输入编码方式，在芯片上运行网络得到运行结果。理论上来说，芯片层面的仿真结果应该与软件平台的仿真结果完全一致。

2.4 硬件资源

达尔文 II 芯片为浙江大学自主研发的一款基于脉冲神经网络的类脑计算芯

片，支持脉冲神经网络的自定义配置，并且前向运行这个网络。图 2.2 所示为达尔文 II 的芯片架构示意图，片上网络的拓扑结构为 2 维网格型拓扑结构，每个节点包含一个路由器和一个神经拟态处理单元（Neuromorphic Processing Unit，NPU），路由器与 NPU 直接相连，负责数据的传递分发。图 2.2 所示为 8×8 大小的示意图，实际芯片共有 28×28 个节点。

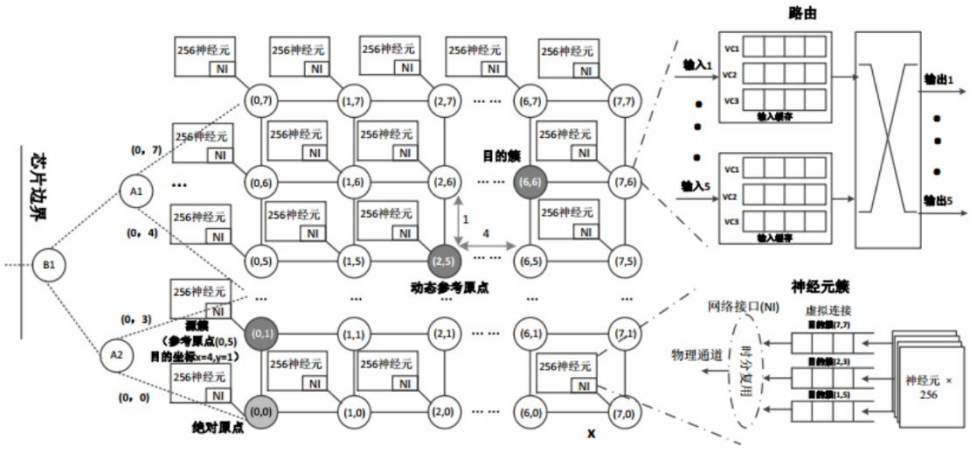

图 2.2 达尔文 II 芯片架构图（8×8 大小）

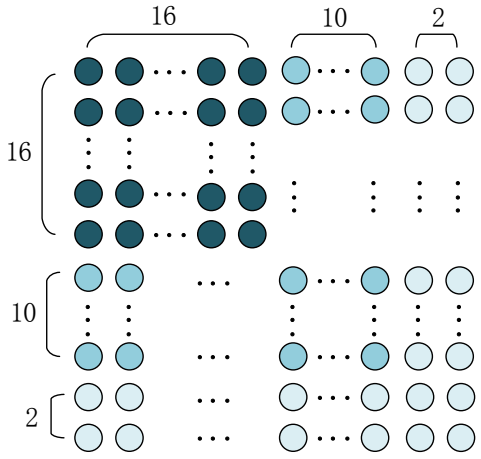

图 2.3 达尔文 II 芯片资源架构

达尔文 II 芯片的每个节点都有一个神经拟态核，每一个神经拟态核都包含一个片上存储单元，但是，为了更高效地利用芯片资源，每一个核的片上存储单元的大小是不一样的。如图 2.3 所示，为达尔文 II 芯片的资源架构图，整个芯片共

有三种不同大小的片上存储单元，对应芯片的不同区域。从芯片的左上角开始，大小为16×16的深蓝色区域内，每个神经拟态核的片上存储空间为32KB，共有256个；在大小为26×26的区域内的其他节点上的片上存储空间为16KB，共有420个；芯片剩余核的片上存储空间为8KB，共有108个。这样根据片上存储单元的大小不同，可将芯片分成三个部分，脉冲神经网络的映射也会基于这三个部分进行。

达尔文II芯片的每个神经拟态核其实只实现了一个物理神经元，但是通过时分复用的方式，每个神经拟态核最多可以支持实现256个逻辑神经元。因此，整块芯片共能够支持28×28×256=200704个神经元。

每个神经元会与其他神经元相连，每条连接用数据包表示，存储在源神经元的片上存储单元中。数据包可分为两类，第一类为头部数据包，用来说明该源神经元连接到的节点坐标以及数据包的长度，第二类为权重数据包，表示源神经元与目的神经元的连接权重，具体格式说明如表2.1所示。

表2.1 数据包格式表

数据包类型	位数	描述
头部数据包	15:12	目的节点的X坐标
	11:8	目的节点的Y坐标
	7:0	数据包长度
权重数据包	15:8	目的神经元编号
	7:0	连接权重

每个神经拟态核的片上存储空间都需要存储地址和数据这两大内容。在本文的芯片设计中，每个神经元数据存储单元的起始地址用32bit的数据表示，并且后一个神经元的起始地址同时也表示前一个神经元数据存储单元的结束。因此，在每个神经拟态核中，若共实现了N个逻辑神经元（N<=256），则需要用来存储地址和头部数据包的空间总共需要[(32+16)×N+32]bit，剩余空间用于存储连接。

对于不同大小的片上存储单元，可存储的连接数不同。假设片上存储空间为 X KB，X 取值为 32，16，8，则对应的连接数量可通过公式 2.1 计算。

$$conn = (X \times 1024 \times 8 - 48N - 32)/2 \tag{2.1}$$

若每个神经拟态核实现 256 个神经元，则对于片上存储为 32KB 的核，共可以存储 15614 条连接，对于片上存储为 16KB 的核，共可以存储 7422 条连接，对于片上存储为 8KB 的核，共可以存储 3326 条连接。因此，对于一个标准的达尔文 II 芯片，即每个神经拟态核都实现 256 个神经元，共可以支持 7473632 条连接。

2.5 网络映射优化

网络映射就是在保持网络结构和网络行为一致的条件下，将层次化的脉冲神经网络在以节点为单位的达尔文 II 芯片上实现。在网络映射的过程中，主要需要考虑的是硬件资源约束以及网络运行的实时性。首先是网络实现的硬件资源评估，由于达尔文 II 芯片的硬件约束，每个节点支持的神经元数量以及连接数量有限，因此我们需要根据给定脉冲神经网络的结构计算其映射所需要的资源，判断达尔文 II 芯片能否满足资源要求。在满足硬件资源需求的情况下进行网络映射的同时，需要考虑到应用运行的实时性和能耗。由于在芯片运行过程中，能耗和时耗主要发生在芯片不同节点之间的数据传输中，所以在网络映射中，需要使数据传输的距离尽可能短，以降低网络运行能耗并且提高网络运行速度。

在应用的硬件资源评估过程中，本文主要分成两步。第一步为模糊计算，判断芯片的资源总数能否满足网络映射需要的资源总数。若芯片的资源总数都不能满足网络的映射需求，那么这个网络是不能配置到芯片上的。若资源总数能够满足网络需求，则可以进行第二步的精确计算计算。第二步直接计算网络的每一层需要的每种神经拟态核的数量，对网络的硬件资源需求得到一个较为准确的数据。

第一步计算脉冲神经网络的资源需求总量，主要是总的神经元数量的计算和总连接数的计算，如公式 2.2 和 2.3 所示。

$$N_{neu} = \sum_{i=1}^{L} n_{neu,i} \tag{2.2}$$

$$CONN = \sum_{i=1}^{L-1} \sum_{j=0}^{n_{neu,i}} conn_{ij} \tag{2.3}$$

其中，公式 2.2 中的$n_{neu,i}$表示网络第 i 层的神经元数量，L 表示网络层数，计算得到的N_{neu}为网络中的神经元总数；公式 2.3 中的$conn_{ij}$表示第 i 层的第 j 个神经元的连接数量，计算的到的 CONN 为该神经网络的连接总数。根据达尔文 II 芯片的资源限制，计算得到的N_{neu}应小于 200704；由于硬件支持的连接数量需要根据每个神经拟态核实现的逻辑神经元数量计算得到，这里给出的连接限制是基于 256 个逻辑神经元的，连接数应不超过 7473632 条。

硬件资源评估的第二步就是根据脉冲神经网络的结构，具体算出每一层需要的芯片节点数量，即得到网络映射的方案。在芯片资源分配和网络映射的过程中，本文设计了如下分配原则：

a. 神经网络的每一层尽量不放在芯片的同一节点中，除非两层网络能够使用一个节点实现；

b.原则上先使用片上存储为 32KB 的神经拟态核，再使用其他节点；

c.尽可能少地使用节点。

d.每一层的网络节点之间的距离尽可能近。

神经网络的映射可以抽象化为二维装箱问题，但是限制条件有所区别。上述分配原则中，原则 c 为装箱问题的最终优化目标，即尽可能少地使用箱子；原则 a 主要是为了简化装箱问题，采用分治思想，可以把网络的每一层映射看作一个小的装箱问题；原则 b 则是节点使用的具体优先级说明；原则 d 主要是映射过程中节点使用的优化，用于提高网络运行的实时性，降低网络运行的功耗。

脉冲神经网络的结构是多种多样的，如图 3.6。对于一些规律连接的网络，我们可以直接得到每一层的每个神经元的连接数 c；还有一些网络层连接是无规律的，对于这种网络，为了简化计算，本文首先计算网络每一层神经元的平均连接数c_{ave}，然后通过一个超参数 reverse 的设置，来给连接数一定的冗余量，最终

无规律连接层每个神经元的平均连接数 c 如公式 2.4 所示。

$$c = c_{ave} \times (1 + reverse) \tag{2.4}$$

那么，我们可以对每一层网络所需要的节点数进行一个比较精确的计算。假设芯片的每个节点可支持最多 256 个神经元和 Link 条连接，那么，实际每个节点最多实现的逻辑神经元的数量如公式 2.5 所示，Link 的取值范围为{15614，7422，3326}；网络每一层所需的节点数量由公式 2.6 计算得到。

$$num_{neu} = \min(\left\lfloor \frac{Link}{c} \right\rfloor, 256) \tag{2.5}$$

$$num_{node} = n_{neu,i} / num_{neu} \tag{2.6}$$

网络总节点的计算是不包括输入层的，因为在脉冲神经网络中，输入层相当于脉冲发生器，一般是对外部输入的数据进行脉冲编码，因此，不会在芯片内部实现。

2.6 本章小结

本章首先对脉冲神经网络的仿真训练平台进行了一个总体性的介绍。平台主要分为四个部分，包括脉冲神经网络标准化模型库、仿真训练平台、硬件资源和网络优化映射，能够实现脉冲神经网络的构建、训练、仿真等功能，支持开发者从零开始构建脉冲神经网络应用。然后，介绍了达尔文 II 芯片的硬件架构及硬件资源约束，提出了脉冲神经网络到达尔文 II 芯片的映射原则。本文的研究重点是标准化的脉冲神经网络模型的构建方法，开发脉冲神经网络训练以及软硬件协同仿真平台，并对平台进行评估。

第3章 面向达尔文II的神经网络建模平台构建

3.1 脉冲神经元模型

3.1.1 脉冲神经元概述

传统的人工神经元模型主要包含两个功能，一是对前一层神经元传递的数值进行加权和，二是使用非线性激活函数对加权和进行计算得到输出值。前者用于模仿生物神经元之间的信息传递方式，后者用来提高神经网络的非线性计算能力，也就是神经网络的表达能力。但是这种神经元模型只是表征性的模仿了生物神经元，实则并没有生物相似度。与传统的人工神经元相比，脉冲神经元从神经科学的角度出发，对生物神经元进行建模，具有很高的生物相似性。

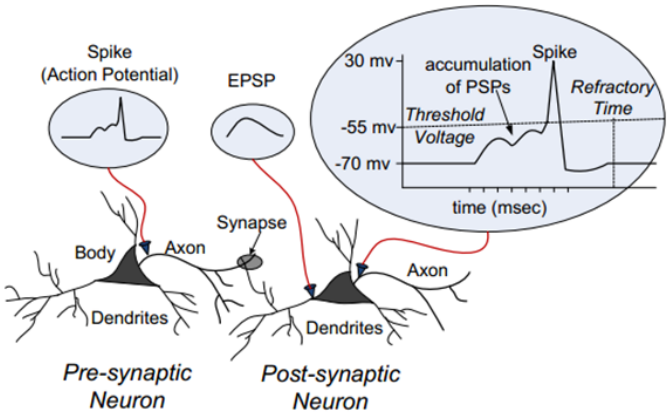

图 3.1 生物神经元工作原理

图 3.1 为生物神经元工作原理示意图。一个生物神经元由树突（Dendrites）、轴突（Axon）和神经细胞体（Soma）组成，树突负责接收其他神经元的传入信息（电压信息）并将信息传递给神经细胞体，神经细胞体累计这些输入，若累计输入达到一定阈值，则会产生一个输出信号（电压信息），通过轴突传递给其他神经元。神经系统中的信号以细胞膜的电位变化传导，两个神经元之间的连接称为突触（Synapse），神经元之间的信息传递通过突触完成。发送信号的神经元被称为突触前神经元（Pre-synapse neuron），接收信号的神经元为突触后神经元

（Post-synapse neuron）。

突触在信息传递过程中存在延时，当突触后神经元接收到兴奋性突触后电位（Excitatory Postsynaptic Potential）时，膜电位会增加，当膜电位达到阈值后该神经元会发放一个脉冲并且将膜电位恢复到一个低于阈值的水平，并且在此后一段时间内对接收到的脉冲输入不做出反应，这一段时间称为静息期（Refractory Time）。

通过对生物神经元进行不同抽象层次的建模，可以得到不同生物相似度的神经元模型，一般来说，脉冲神经元模型的生物相似度越高，则其的计算复杂度也越高[23]，如图 3.2 所示为不同抽象层次的神经元模型，模型的运算复杂度随生物精确性的提高而提高。

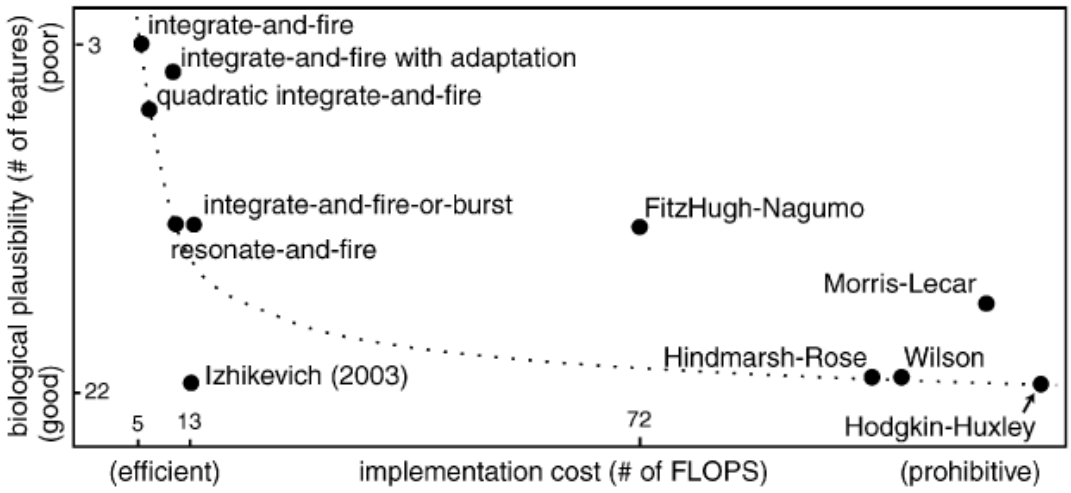

图 3.2 脉冲神经元模型与计算复杂度

Hodgkin-Huxley 模型[24]具有最高的生物相似度，使用一组非线性微分方程来描述细胞膜上的离子通道的开闭情况，但是也是由于它极高的生物相似度，其神经元的运算量也非常高，对于大规模神经网络的应用难以达到实时性要求。因此，一系列简化的神经元模型应运而生，主要有 Morris-Lecar 模型[25]、Izhikevich 模型[26]等，由两到三个非线性微分方程描述。此外，还有最简单的 LIF（Leaky Integrate-and-Fire）模型[27]及其衍生模型，虽然其生物相似度最低，但是计算非常简单，利于大规模应用的实现，因此，这也是目前使用最为广泛的神经元模型。

本文中，主要使用的是 LIF 模型、IF（Integrate-and-Fire）模型以及达尔文 II 芯片支持的 Darwin 模型。

3.1.2 IF（Integrate-and-Fire）模型

IF 模型是最简单的脉冲神经元模型，仅考虑了 Hodgkin-Huxley 模型中的漏电流，可用公式 3.1 所示的微分方程描述。由于 IF 模型的计算量极小且易于实现，它在于多脉冲神经网络中被应用。

$$i_{IF}(t) = C_{IF}\frac{dV_{IF}}{dt} \tag{3.1}$$

C_{IF}表示电容器的电容，通过解微分方程，可以得到神经元膜电位的表达式如公式 3.2 所示。

$$V_{IF}(t) = V_r + \frac{1}{C_{IF}}\int_0^{t-t_{pre}} i_{IF}(t-s)ds \tag{3.2}$$

V_r表示神经元的重置电位，即神经元发放脉冲后的电位，t_{pre}为前一个脉冲的发放时间。

3.1.3 LIF 模型

IF 模型虽然是最简单的，但是其生物相似度并不高，目前更为广泛应用的模型是 LIF 模型，它同样具有计算简单的优点，并且比 IF 模型具有更高的生物相似度。图 3.3 为 LIF 模型的等效模拟电路图。通过一个膜电容 Cm 和一个膜电阻 Rm 的并联来模拟神经元细胞膜的非绝缘性，因此膜电位存在着衰减。Vrest 为神经元的静息电位，Vth 是神经元的阈值电压，当膜电位超过 Vth 时，会发放脉冲。

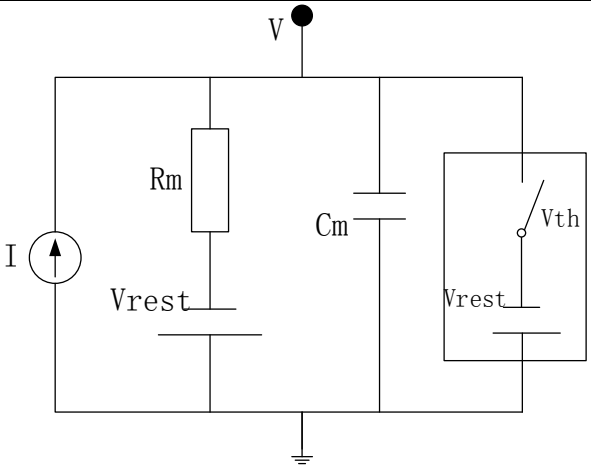

图 3.3 LIF 模型等效模拟电路图

当突触前神经元产生一个脉冲传递给突触后神经元时，会在突触后神经元的细胞膜上产生相应的电流 I，由于电容和电阻是并联的，因此产生的电流会分成两个部分，一部分用于对电容器 Cm 的充电，一部分被膜电阻 Rm 消耗掉。因此，LIF 模型可以用如公式 3.3 所示的微分方程描述。

$$\tau_m \frac{dV}{dt} = -(V - V_{rest}) + R_m I \tag{3.3}$$

其中，$\tau_m = R_m C_m$，为膜时间常量，V 为神经元膜电位，R_m 为膜电阻的电阻值，C_m 为膜电容的电容值。若以 V_{rest} 作为方程的初始条件，通过解得微分方程可得：

$$V(t) = V_{rest} e^{-\frac{t-t_{pre}}{\tau_m}} + \frac{1}{C_m}\int_0^{t-t_{pre}} e^{\frac{t-t_{pre}}{\tau_m}} \cdot I(t-s)ds \tag{3.4}$$

t_{pre} 为上一个脉冲的发放时间。

LIF 模型的神经元行为如图 3.4 所示。当突触后神经元接收到兴奋性突触后电位时，膜电位上升，在没有接收到输入时，膜电位会以指数形式衰减，当膜电位超过阈值时，神经元会发放脉冲并恢复到静息状态。

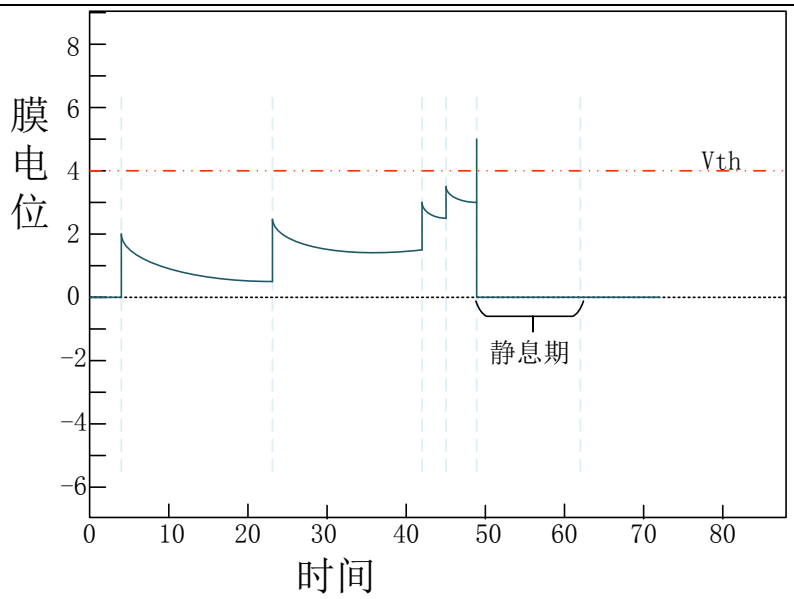

图 3.4 LIF 神经元行为

3.1.4 Darwin 神经元模型

由于硬件约束，在达尔文 II 芯片上实现标准的 LIF 模型较为困难，因为利用硬件实现指数计算的成本非常高。因此，综合考虑神经元模型精度和硬件资源消耗，在这两者均能满足较好需求的情况下，本文对 LIF 模型进行了改进，提出了 Darwin 神经元模型，使用线性衰减代替 LIF 模型中的指数衰减，在具有一定生物相似性的前提下，使得硬件实现更为简单。同时，Darwin 模型可分为两种类型，第一类是单纯线性衰减的 Darwin 模型，本文称之为 DarwinI，可以由公式 3.5 描述。

$$V(t) = V(t-1) + leak + \sum_t W \cdot \delta(t) \tag{3.5}$$

V(t)和 V(t-1)分别表示当前时刻和前一时刻的膜电位；leak 表示每一个单位时间的膜电位衰减，leak 可为正数、负数和零，当 leak 为零时，相当于 IF 模型，当 leak 为负数时，其行为接近 LIF 模型的神经元行为，当 leak 为正时，在网络没有输入的情况下，神经元也会发放脉冲；$\sum_t W \cdot \delta(t)$ 表示该神经元在当前时刻收到了的所有脉冲信号的加权和，表示对膜电位的累积。

第二类是双向衰减的 Darwin 神经元模型，本文称之为 DarwinII，可以用公式

3.6 描述：

$$V(t)=\begin{cases} V(t-1)-leak+\sum_t W\cdot\delta(t) & V(t-1)>0 \\ V(t-1)+\sum_t W\cdot\delta(t) & V(t-1)=0 \\ V(t-1)+leak+\sum_t W\cdot\delta(t) & V(t-1)<0 \end{cases} \tag{3.6}$$

V(t)和 V(t-1)依旧分别表示当前时刻和前一时刻的膜电位；leak 表示每一个单位时间的膜电位衰减，此时，leak 只能为正数，当 V(t-1)大于零时，膜电位向下衰减，当 V(t-1)小于零时，膜电位向上增加，即膜电位始终向零靠拢；$\sum_t W\cdot\delta(t)$ 依旧表示该神经元在当前时刻收到了的所有脉冲信号的加权和，表示对膜电位的累积。

两种类型的 Darwin 神经元模型的行为如图 3.5 所示。图 3.5（a）和图 3.5（b）对应单向衰减的 DarwinI 模型，（a）为 leak 为负数，（b）为 leak 为零，图 3.5（c）对应双向衰减的 DarwinII 模型，可以看出，膜电位始终向零衰减。

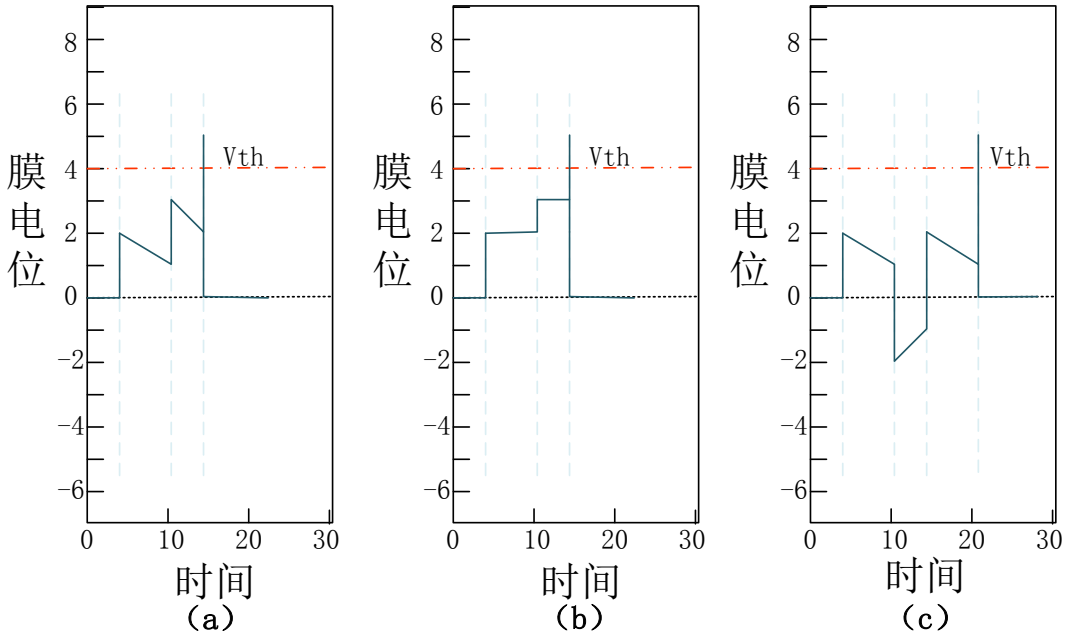

图 3.5 Darwin 神经元行为

3.2 面向达尔文 II 的脉冲神经网络描述

3.2.1 脉冲神经网络连接表示

一个网络模型中最重要的是网络结构，不同的网络结构能够实现的功能大不相同。与传统的人工神经网络一样，脉冲神经网络也是模拟生物神经元的连接方式来实现特定的功能，因此，常见的人工神经网络模型也会在脉冲神经网络中出现，例如 Deep SNN，CSNN（Convolutional SNN）等[28]。目前广泛应用的人工神经网络模型中，主要存在如图 3.6 所示的一些连接方式。

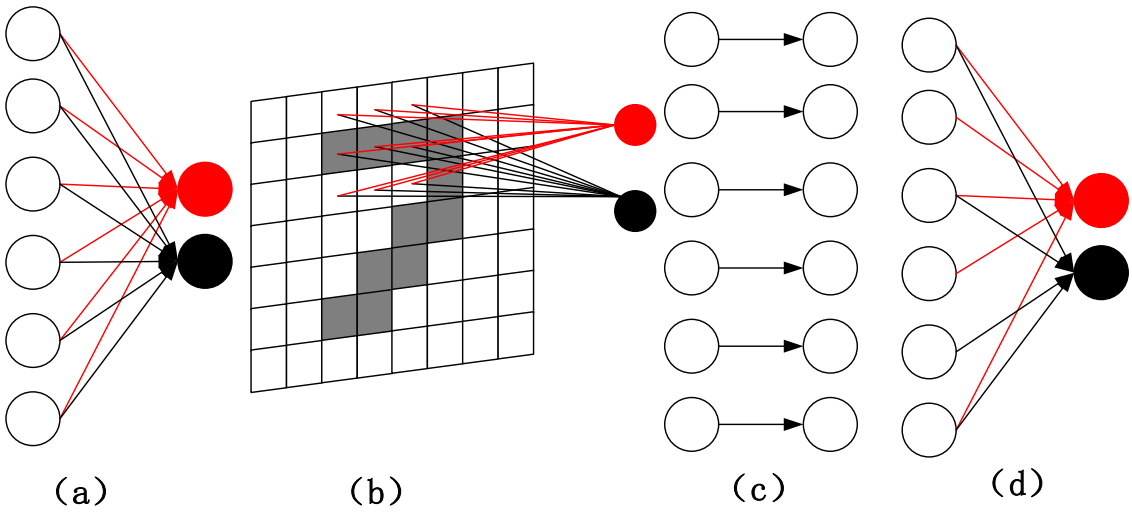

图 3.6 常见的网络连接方式

图 3.6 中，(a) 为全连接层，(b) 为卷基层，(c) 为一对一连接层，这三种连接都是有规律的连接，(d) 为无规律的随机连接层。并且，根据连接的稀疏程度，本文将这些连接分为稀疏连接和稠密连接，显然，一对一连接和卷积连接为稀疏连接，全连接为稠密连接。根据这两种连接方式的不同，本文设计了两种不同的连接表示方法以高效地表示不同的网络层。

在脉冲神经网络中，由于突触传输信息存在延时，严格意义上来说，不同的突触延时不同。但是，为了简化设计，本文认为同一层网络的突触具有相同的延时，因此，网络的延时可以由开发者单独指定，在这两种连接表示方法中，均不涉及延时的表示。

第一种连接表示方法是采用邻接矩阵。矩阵的行表示源神经元，矩阵的列表示目的神经元，矩阵中存储的信息表示这两个神经元之间的连接权重，当连接权重为0时，表示这两个神经元之间没有连接。如图3.7（a）所示为一个邻接矩阵M，表示一个4X5的网络连接，M[1][2]表示1号源神经元和2号目的神经元的连接权重为7。这种连接表示方法更适用于稠密连接。

(a)邻接矩阵 (b)三元组

图3.7 网络连接表示方法

第二种连接表示方法为元组表示，将每一条连接都表示成一个单独的三元组，三元组的元素为源神经元编号，目的神经元编号和神经元之间的连接权重。如图3.7（b）所示为三元组表示方法，第一条记录表示0号源神经元与1号目的神经元的连接权重为2。这种表示方法更适用于稀疏连接。

在实际应用中，两种表示方法都适用，但是，开发者仍然需要综合考虑自己的应用，选择更加高效的表示方法。

3.2.2 脉冲神经网络规范化描述

表 3.1 网络超参数说明

参数	取值类型	说明
neurontype	0，1	0：单向衰减的 Darwin 模型； 1：双向衰减的 Darwin 模型。
Vth	整型数组	每一层的阈值电压，与网络层数相同
leak	整型	神经元电压的衰减
leaksign	-1，1	-1：不衰减 1：衰减
reset_mode	0，1	神经元电压重置模式，如公式 3.7 所示
netDepth	整型	包括输入层在内的网络层数
layerWidth	整型数组	网络每一层的神经元数目
delay	整型数组	网络每一层的延时，共有 0，1，2 三种选择

一个完整的脉冲神经网络，除了指定网络结构，还需要指定神经元模型以及其他超参数。神经元模型必须为达尔文 II 芯片支持的几种神经元模型，网络需要指定的超参数设计如表 3.1 所示。

$$V(t)=\begin{cases}0 & reset\_\bmod e=0\\ V(t)-Vth & reset\_\bmod e=1\end{cases} \tag{3.7}$$

3.2.3 神经元模型实现

Darwin 神经元模型基于 Brain2 实现。在 Brain2 中，使用一组微分方程来表示神经元膜电位的动态特性，如公式 3.8 所示为衰减的 Darwin 模型，v 表示神经元的膜电位，tau 为时间常数，I 的单位为伏特，表示神经元膜电位的衰减，对应于表 3.1 中的 leak。

$$\begin{array}{l} dv/dt = ((-70*mV - v) + I)/tau : volt \\ I : volt \\ tau : second \end{array} \quad 3.8$$

公式 3.9 为不衰减的 Darwin 模型，v 表示神经元的膜电位，tau 为时间常数，I 为输入电流。

$$\begin{array}{l} dv/dt = I/tau : 1 \\ I : 1 \\ tau : second \end{array} \quad 3.9$$

表 3.1 中的参数 Vth 和 reset_mode 在构建神经元时指定。如公式 3.10 所示，构建了 1 个神经元，eqs 为定义神经元的微分方程，同时指定了阈值电压和神经元重置模式。

G=NeuronGroup(1,eqs,threshold='Vth',reset='reset_mode') 3.10

3.3 仿真训练平台核心类

3.3.1 神经元模型类（NeuronGroup）

神经元模型类主要构建脉冲神经网络的神经网络层，每个层需要指定神经元的数量以及神经元的类型，包括 IF、LIF 和 DarwinII 神经元，以及其他超参数。神经元模型类的设计如下：

Class NeuronGroup：
Parameters： *Num*：表示神经元的数量； *NeuronType*：表示神经元的类型，可以选择 IF、LIF、DarwinI，DarwinII； *Reset_mode*：神经元发出脉冲后重置膜电位的方式； *Leaksign*：表示神经元是否衰减； *Leak*：神经元的衰减常数； *Threshold*：阈值电压； *Delay*：延时

3.3.2 突触和连接

突触（Synapse）为突触模型，用于定义神经元之间连接的类型。两个神经元之间的连接称为突触（Synapse），发送信号的为突触前神经元（pre-synapse neuron），接收信号的为突触后神经元（post-synapse neuron），如图 3.8 所示，j 表示突触前神经元，i 表示突触后神经元。

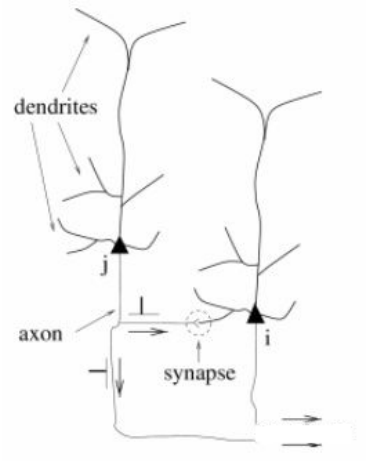

图 3.8 神经元连接示意图

本文使用 Synapse 类定义神经元之间的连接类型，使用连接（Connect）类建立连接，可以实现的连接种类包括卷积连接，池化连接，全连接等。具体定义如下：

Class Synapse：

Parameters：

model: 表示突触的模型。

On_pre: 表示突触前膜神经元产生脉冲时的动作，如‘v_post += w’表示当突触前膜神经元产生脉冲时，突触后膜神经元的电压值增加 w。

Class Connect：

Parameters：

sou: 表示连接的源神经元。

tar: 表示连接的目的神经元。

Sou_index: 源神经元的索引。

Tar_index: 目的神经元的索引。

weight: 连接的权重。

delay: 源神经元与目的神经元脉冲传递的延迟。

Synapsestype: 突触的类型。

3.3.3 状态记录器

许多认知模型的基础是积分器，即从数学上讲，网络的输出为输入的积分。同样的，脉冲神经网络中的积分器就是脉冲神经元。为了更好地观察和分析网络的运行状态，本文设计了两个记录器用于保存网络中相关变量的中间状态，分别是 StateMonitor 和 SpikeMonitor。

StateMonitor 用于记录神经元的状态，主要记录的信息包括每个时刻神经元的膜电位以及状态变量的值等，输出文件内容分为需要记录的神经元编号及其对应的参数状态，设计如下：

Class StateMonitor：

Parameters：

NeuIdx：需要记录的神经元编号。

Variable：需要记录的参数。

Output_file：输出文件的保存位置。

同时，StateMonitor 也可以实时输出所需记录的神经元的膜电位图，如图 3.9 所示，为记录的两个神经元的膜电位变化图。

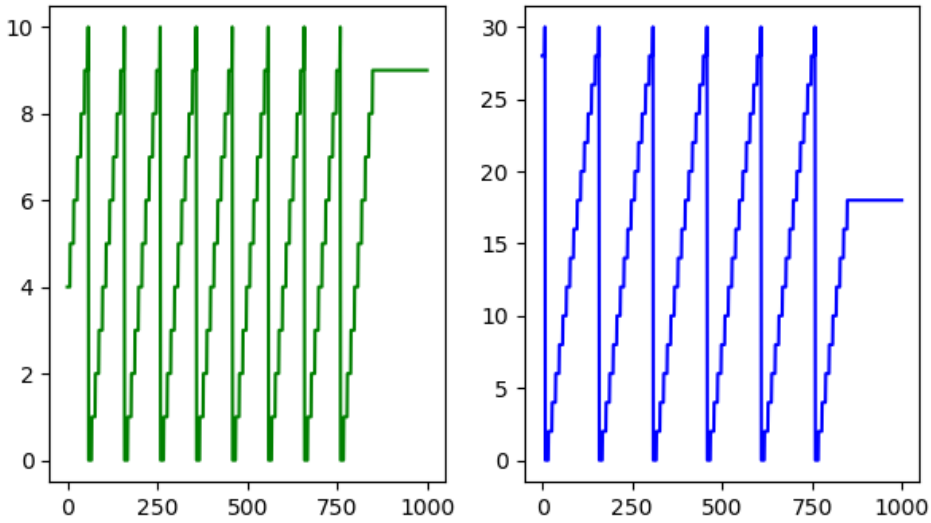

图 3.9 StateMonitor 记录的神经元膜电位变化图

SpikeMonitor 用于记录脉冲信息，主要包括脉冲发放的时间等，输出文件包括需要记录的神经元及其对应的脉冲发放时间以及数量，设计如下：

Class SpikeMonitor：
Parameters： *NeuIdx*：需要记录的神经元编号。 *Output_file*：输出文件的保存位置。

3.4 本章小结

本章主要从神经元建模以及神经网络连接的角度描述了面向达尔文 II 芯片的神经网络建模。首先是提出 Darwin 神经元模型，然后提出了两种神经网络结构的表示方法以及面向达尔文 II 芯片的神经网络的具体描述规范。最后，对于脉冲编码以及仿真训练平台核心类的具体实现做了说明。

第4章 面向达尔文 II 的网络训练及仿真平台构建

本平台封装了 Brain2 来实现脉冲神经网络的构建、直接训练和仿真。Brain2 是一个非常方便的脉冲神经网络的模拟器，支持神经元和突触动态特性的自定义。我们在 Brain2 上自定义实现了达尔文 II 芯片支持的两种 Darwin 神经元模型。当输入一个符合要求的脉冲神经网络后，可以使用 Brain2 对网络进行构建以及后续操作，包括网络的训练及仿真。

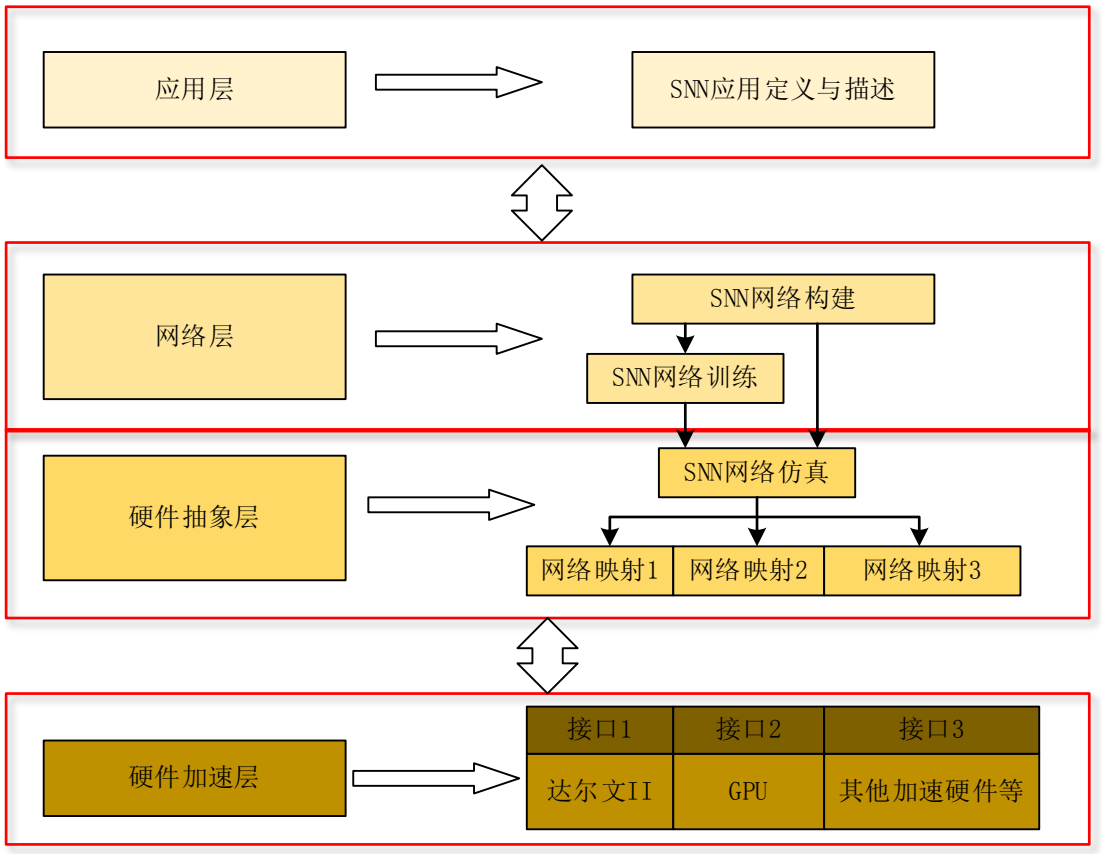

图 4.1 仿真训练平台抽象

如图 4.1 所示为仿真训练平台抽象图，整个平台可抽象为应用层、网络层、硬件抽象层和硬件加速层四个层次。应用层主要面向开发者，实现脉冲神经网络应用的定义与描述，在第三章中已进行具体说明。网络层主要是实现脉冲神经网

络的构建和训练。硬件抽象层主要是基于硬件加速器对该应用的软件实现，包括软件仿真以及针对特定硬件加速器的网络映射方法。硬件加速层主要是针对脉冲神经网络的加速实现，通过不同的接口，可以支持不同的硬件加速器，如达尔文 II、GPU 以及其他硬件加速芯片等，支持不同加速器的灵活扩展。需要指出的是，本章主要实现基于达尔文 II 类脑计算芯片这一硬件加速器训练及仿真。

4.1 脉冲神经网络的直接训练方法

由于脉冲神经网络是基于事件驱动的，使用离散的脉冲来计算和传递信息，脉冲神经元的传递函数通常是不可微的，从而使用传统的反向传播算法训练较困难。但是，目前仍有一些较为有效的训练学习算法。目前主要的学习训练算法有非监督学习、监督学习、强化学习和演化算法等几大类[31]。

4.1.1 突触可塑性学习算法

突触可塑性学习算法(Spike-Timing-Dependent Plasticity, STDP)[32]是一种非监督学习算法，基于赫布法则（Hebbian Rule），能够实现突触权重的自适应调整。STDP 根据特定神经元输入和输出脉冲的相对时序来调整神经元之间的连接强度：如果两个神经元同时兴奋，那么他们之间的突触连接会增强，即，在一个时间窗口内，当突触前神经元的脉冲发放早于突触后神经元（约等于 10ms），那么，他们之间的突触连接权重就会增加，被称为长期增强(Long-Term Potential,LTP)；如果突触前神经元的脉冲发放迟于突触后神经元时，那么认为时间事件之间不存在因果关系，他们之间的突触权重将会降低，被称为长期抑制(Long-Term Depression,LTD)。“长期”一词用于区分实验中观察到的几毫秒范围内非常短暂的影响。

突触可塑性学习算法强调了脉冲发放时序不对称的重要性。最常见的 STDP 学习规则可用公式 4.1[32]描述。

$$W(s)=\begin{cases}A_{+}\exp(-s/\tau_{+}) & s\geq 0\\ -A_{-}\exp(s/\tau_{-}) & s<0\end{cases} \tag{4.1}$$

W 为突触权重，A_{+}、A_{-}通常为常数，表示学习率。τ_{+}、τ_{-}为时间窗口的时

间常数。公式 4.1 的上半部分描述的是 LTP 的过程，下半部分描述 LTD 的过程。增强和衰减的强弱效果由衰减指数调控，衰减指数的大小由突触前和突触后发放脉冲的时间差以及时间常数τ_+、τ_-决定。

对于突触 j，其总的权重改变量为每一对突触前神经元和突触后神经元权重改变量之和，如公式 4.2[32]所示。

$$\Delta w_j = \sum_{m=1}^{N}\sum_{n=1}^{N} W(t_i^{(n)} - t_j^{(m)}) \tag{4.2}$$

图 4.2 所示为学习率均为 0.5，时间常数均为 10ms 的 STDP 学习窗口，突触前神经元在t_j时刻发放脉冲，突触后神经元在t_i时刻发放脉冲，图中可以明显看出 STDP 学习算法如何根据脉冲发放时间改变神经元之间的连接强度。

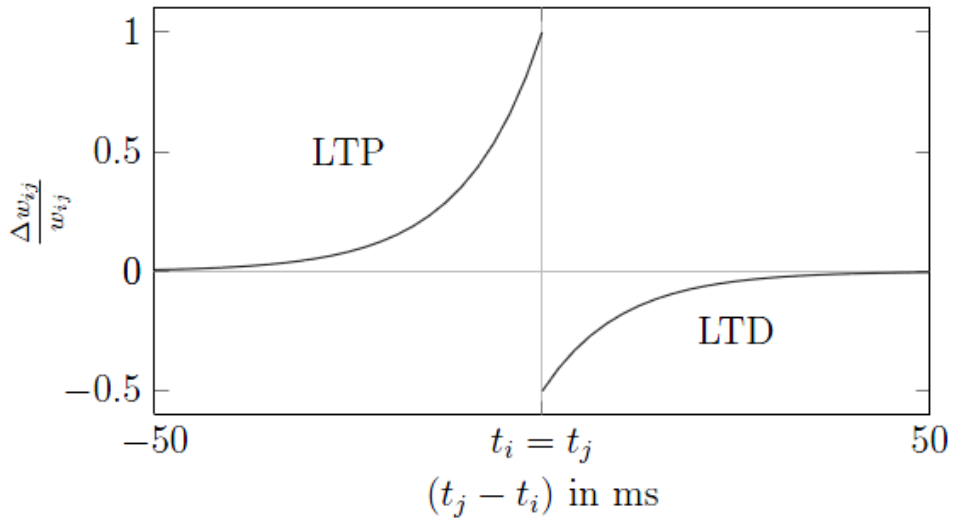

图 4.2 STDP 学习窗口

4.1.2 监督式突触学习算法（Tempotron）

脉冲发放时间在多种不同种类的神经元的突触学习中起着至关重要的作用。Tempotron 和 STDP 一样，也是一种突触学习算法，不同的是，Tempotron 是一种监督式的突触学习算法，由 Robert 等[33]提出，适用于时空脉冲模式的信息学习，学习目标是使神经元实际输出的膜电位与期望输出的膜电位尽可能接近。

Tempotron 学习算法基于 LIF 神经元模型，认为神经元后突触膜电位

（Postsynapstic Potentials，PSPs）是所有与之相连的突触前神经元传入脉冲的加权和，如公式 4.3[33]所示，其中 t_i 表示第 i 个脉冲的传入时间；$K(t-t_i)$ 是归一化后的 PSPs，计算公式如 4.4 所示，τ 和 τ s 分别表示膜电位和突触电流的衰减时间常数，因数 V0 将 PSPs 内核归一化到 1。当 V(t)超过脉冲发放阈值时，神经元会输出一个脉冲，V(t)会被重置为神经元的静息电位 Vrest。

$$V(t)=\sum_i \omega_i \sum_{t_i} K(t-t_i)+V_{rest} \tag{4.3}$$

$$K(t-t_i)=V_0(\mathrm{e}^{\frac{-(t-t_i)}{\tau}}-e^{\frac{-(t-t_i)}{\tau_s}}) \tag{4.4}$$

在 Tempotron 中，每个输入模式都属于 0 或者 1 两种类别之一。假设输入脉冲序列共包括 N 个脉冲，Tempotron 的任务是通过发放脉冲来响应对应的模式，并且在发生错误时，则改变神经元之间的连接权重。当应该输出 0，但是输出神经元发放了脉冲时，Tempotron 会更新与该神经元相连的上一层神经元的权重，更新的目的是抑制该输出神经元的膜电位，使其不发放脉冲，因此，与该输出神经元相连的输入神经元的权重将会减小，更新量计算如公式 4.5[33]所示。

$$\Delta\omega_i=\lambda\sum_{t_i<t_{\max}} K(t_{\max}-t_i) \tag{4.5}$$

其中，t_{max} 为突触后膜电位 V(t)达到最大值的时间，λ 是一个大于零的常数，表示学习率。

在 Tempotron 中，输出神经元 0 和 1 两种状态，非常适用于解决二分类任务，对于多分类任务，可以采用二进制编码的方式输出类别。例如，若输出类别共有 10 类，则可以用 4 位二进制数表示，脉冲神经网络有 4 个输出节点。Tempotron 成功实现了单脉冲的时空模式分类，但是并不适用于大规模的脉冲神经网络应用。

4.1.3 基于 SNN 的反向传播算法

反向传播算法（Backpropagation）[34]是用来训练 ANN 的最常见也是最基础的算法，在 ANN 训练中得到了非常好的效果。其算法思想主要是从网络所犯错误中学习，通过定义目标函数，对网络中所有的权重计算函数梯度，从而找到目

标函数的最优更新方向。反向传播算法基于连续可微函数，由于 SNN 中传递的是离散的脉冲信息，基于SNN的反向传播算法需要有所改动，以适应SNN的计算特性。

传统的反向传播算法使用链式法则计算损失函数对网络每一层权重θ的偏导数，然后在负梯度方向更新权重，从而达到网络学习的目的，使损失函数的值尽可能小。反向传播算法的基本流程分为五步[34]：

a. 前向计算网络每一层的输入及输出，直到网络的输出层l_{nl},网络每一层的输入用$z^{(l)}$表示，l代表层数，每一层的输出入公式4.6所示，其中f为每一层的激活函数；

$$a^{(l)} = f^{(l)}(z^{(l)}) \tag{4.6}$$

b. 计算输出层的偏差，如公式4.7所示，其中y表示期望输出的标签向量；

$$\delta^{(n_l)} = \frac{\partial L(a^{(n_l)}, y)}{\partial z^{(n_l)}} = \frac{\partial L(a^{(n_l)}, y)}{\partial a^{(n_l)}} \cdot f'(z^{(n_l)}) \tag{4.7}$$

c. 从$l = n_l - 1$，$n_l - 2$，……，2,反向计算每一层的梯度，如公式4.8所示，$W^{(l)}$为第l层的权重向量；

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) \cdot f'(z^{(l)}) \tag{4.8}$$

d. 计算每一层的偏导数；

$$\begin{aligned} \nabla_{W^{(l)}} L &= \delta^{(l+1)} (a^{(l)})^T \\ \nabla_{b^{(l)}} L &= \delta^{(l+1)} \end{aligned} \tag{4.9}$$

e. 更新参数，如公式4.10所示，η为学习率。

$$\begin{aligned} W^{(l)} &= W^{(l)} - \eta \nabla_{W^{(l)}} L \\ b^{(l)} &= b^{(l)} - \eta \nabla_{b^{(l)}} L \end{aligned} \tag{4.20}$$

以上计算流程和公式都是针对 ANN 的数值计算设计的，而对于基于事件的脉冲神经网络来说，需要对计算偏导数的方式作出改进。

对于一个LIF神经元，它的膜电位更新方式可以用公式4.11和4.13描述[35]，V为膜电位，t_p和t_{p-1}为当前和前一个脉冲产生的时间，$w_i^{(p)}$为第i个突触的权重，

动态权重w_{dyn}用于静息期控制。

$$V(t_p) = V(t_{p-1})e^{\frac{t_{p-1}-t_p}{\tau}} + w_i^{(p)} w_{dyn} \tag{4.11}$$

$$w_{dyn} = \begin{cases} (\Delta t / T_{ref})^2 & \Delta t < T_{ref} \\ 1 & otherwise \end{cases} \tag{4.12}$$

$$V(t) = V(t) - Vth \tag{4.13}$$

因为脉冲信号是离散的，并不能用于计算梯度，所以在基于SNN的反向传播算法[36]中，并不直接采用脉冲信号进行反向传播，而是计算每个脉冲事件对神经元膜电位的累积影响。

$$\begin{aligned} x_k(t) &= \sum_p \exp(\frac{t_p - t}{\tau}) \\ a_i(t) &= \sum_q \exp(\frac{t_q - t}{\tau}) \end{aligned} \tag{4.14}$$

如公式4.14所示，$x_k(t)$表示第k个突触输入的脉冲对目标神经元膜电位的累积效应，$a_i(t)$表示第i个神经元发出脉冲后对自身膜电位的影响，其中t_p、t_q均小于t。这样，除了脉冲发放的时间点，公式4.14中的变量为连续变量，可用于反向传播计算，忽略静息期的影响，公式4.11可被改写为公式4.15。

$$V_i(t) = \sum_{k=1}^{m} w_{ik} x_{ik}(t) - Vth_i a_i(t) + \sigma Vth_i \sum_{j=1, j\neq i}^{n} k_{ij} a_j(t) \tag{4.15}$$

公式4.15的右边三项分别表示输入，膜电位重置和侧向抑制，k_{ij}表示第j个神经元对第i个神经元的侧向抑制强度，σ为小于1的常数，表示侧向抑制的预期效果。在所有神经元都具有相同的时间常数τ的情况下，由于当前层的输出a_i即为下一层的输入x_k，因此公式4.15为通过链式法则推导基于SNN的反向传播算法提供了基础。将公式4.15中的$V_i(t)$设为零，可以推导得到反向传播算法的转换方程，公式4.16。

$$a_i \approx \frac{s_i}{Vth_i} + \sigma \sum_{j=1, j\neq i}^{n} k_{ij} a_j(t), \text{其中} s_i = \sum_{k=1}^{m} w_{ik} x_k \tag{4.16}$$

直接对公式4.16进行微分计算，可以得到基于SNN的反向传播计算公式，

如公式 4.17-4.21。

$$\frac{\partial a_i}{\partial s_i} \approx \frac{1}{Vth_i} \tag{4.17}$$

$$\frac{\partial a_i}{\partial w_{ik}} \approx \frac{\partial a_i}{\partial s_i} x_k \tag{4.18}$$

$$\frac{\partial a_i}{\partial Vth_i} \approx \frac{\partial a_i}{\partial s_i}(-a_i + \sigma \sum_{j \neq i}^{n} k_{ij} a_j) \tag{4.19}$$

$$\frac{\partial a_i}{\partial k_{ih}} \approx \frac{\partial a_i}{\partial s_i}(\sigma Vth_i a_h) \tag{4.20}$$

$$\frac{\partial a_i}{\partial x_k} \approx \frac{\partial a_i}{\partial s_i} \frac{1}{1-\mu\sigma}(w_{ik} - \frac{\mu\sigma Vth_i}{1+\mu\sigma(n-1)} \sum_{j=1}^{n} \frac{w_{jk}}{Vth_j}) \tag{4.21}$$

通过将公式 4.17-4.21 中的导数插入到标准误差反向传播算法中，可获得一种有效的 SNN 学习规则，在梯度的推导中仅考虑了抑制性横向连接的一阶效应。对于抑制性横向连接，参数变化的影响会随着连接距离的增加而迅速衰减。因此，一阶近似可节省大量计算成本，而不会损失准确性。

4.1.4 脉冲神经网络训练（SNN Learning）

脉冲神经网络的直接训练算法主要在 BRAIN BACKED 中实现。目前，主要支持两种直接训练算法：STDP 和基于 SNN 的反向传播算法。开发者不需要了解算法的具体实现细节，而是像 TensorFlow 中提供的训练 API 一样，提供正确的标签并且定义损失函数即可开始网络的训练。API 设计如下：

Class STDPOptimizer：

Parameters：

Learning_rate_pos：网络的连接增强学习率，相当于 A_+；

Learning_rate_pos：网络的连接减弱学习率，相当于 A_-；

Tau：时间窗口常数；

Vth：脉冲神经元阈值电压。

Time_window：网络训练时间。

Class BPoptimizer：

Parameters：

Learning_rate：网络的学习率；

Tau：时间窗口常数；

Vth：脉冲神经元阈值电压。

Sigma：侧向抑制常数。

Label：样本的实际标签。

4.2 脉冲神经网络的间接训练方法

4.2.1 人工神经网络裁剪

所谓的间接训练，即首先使用深度学习中的网络训练方法对传统人工神经网络进行训练，然后将训练完成的网络转化成为脉冲神经网络。其中主要包括深度学习相关的模块，包括 ANN 的构建、训练，以及脉冲神经网络的转化。具体流程如图 4.3 所示。

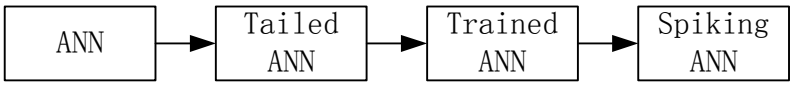

图 4.3 脉冲神经网络间接训练流程图

第一步是构建和训练传统人工神经网络，主要基于 TensorFlow 库完成。

Tensorflow 库是一个端到端开源计算平台，开发者可以轻松地使用其中的 keras 等高阶API构建神经网络模型，具有高度的灵活性以及可移植性。同时，Tensorflow 可以将预测模型结构和目标函数相结合，自动计算相关微分导数，为基于梯度的学习算法提供了有力支持。本文的传统人工神经网络的训练主要基于反向传播算法。

将普通的人工神经网络转化成为脉冲神经网络也存在着一些难点[37]。ANN 主要基于数值计算，若网络的激活函数为 tanh()，其输出值得范围为-1 到 1，网络传递的数值中存在正负，但是 SNN 是基于脉冲事件传递信息的，因此如何在 SNN 中表示 ANN 中的负数是难点之一。虽然在 SNN 中可以使用抑制神经元来表示负值，但是这样不仅会导致神经元数量增加，也会使脉冲神经网络的结构更为复杂。其次，在 ANN 的网络层中，偏置项可以为正数或者负数，在 SNN 中同样很难表示。最后，若网络为卷积神经网络，其中存在空间最大池化层（即使用输入特征图中一小片区域的最大值作为该区域的输出），在 SNN 中则需要两层网络才能实现相同的功能。

因此，在构建 ANN 的过程中，需要对其进行裁剪，使其更易于转化成为脉冲神经网络。首先，保证在 ANN 的构建和训练过程的数值都为非负数，主要方法是在数据预处理之后加入绝对值计算，使得输入网络的参数为非负数，网络的激活函数选择线性整流单元 ReLU，如公式 4.5 所示，这样能够保证网络的每个单元输出为非负数。其次，在 ANN 的构建和训练过程中和直接去除偏置项，实验证明，去除偏置项后对网络的效果几乎没有影响[37]。最后，使用空间线性池化代替最大池化层，空间线性池化是指将空间的输入值求平均而不是取最大值。

$$f(x) = \max(0, x) \tag{4.5}$$

4.2.2 ANN 到 SNN 的转化

ANN 到 SNN 的转化是一个非常直接的过程。将上文所述经过裁剪的 ANN 训练完成后，在保持 ANN 网络结构和网络权重不变的情况下，将 ANN 的神经元替换成为 Darwin 脉冲神经元，即完成了 ANN 到 SNN 的转化。转化过后，ANN 中

的权重即表示 SNN 中神经元之间的连接强度。

如图 4.4 所示，为一个卷积神经网络（convolutional neural network，CNN）转化成为脉冲神经网络的示意图。图 4.4（a）为裁剪过后的卷积神经网络，可以看出，整个网络中去掉了偏置项，并且使用了 ReLU 作为激活函数，使用了平均池化层代替了最大池化层。图 4.4（b）为转化过后的脉冲神经网络，整个网络结构保持不变，但是在网络输入预处理之后加入了脉冲产生层，主要是对输入数据进行编码，用脉冲序列的形式输入；在网络的最后加入了脉冲计数器，统计在网络运行的时间窗口内每个输出神经元产生的脉冲。

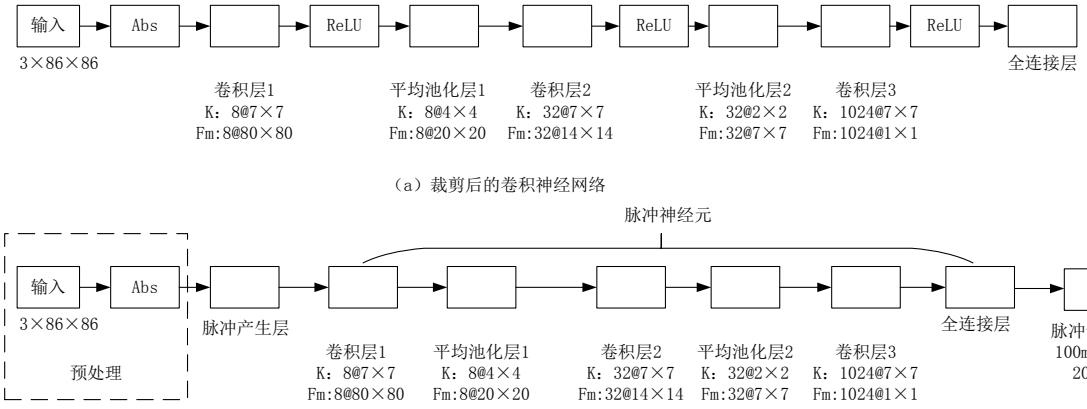

图 4.4 CNN 到 SNN 转化示意图

4.2.3 深度学习基础库（Simulator DL）

Simulator DL 主要实现脉冲神经网络的间接训练功能。本平台封装了 tensorflow 训练库，并且设计实现了一些方法自动提取 ANN 中的网络结构并生成 SNN 中的连接，连接主要由三元组表示，主要支持的连接提取层有卷积层、池化层、全连接层、一对一连接层等，设计如下：

Class GenConnection：

Function gen_conv_conn：

Parameters：

Sou: 源神经元层；

Des：目的神经元层；

Filters：卷积核；

Pads：边缘补偿；

Stride：卷积操作的步长。

Function gen_avepooling_conn:

Parameters：

Sou：源神经元层；

Des：目的神经元层；

Kernel_size：卷积核的大小；

Pads：边缘补偿；

Stride：池化操作的步长。

Function gen_fc_conn：

Parameters：

Sou：源神经元层；

Des：目的神经元层；

Weights：网络两层之间的连接权重矩阵。

Function get_o2o_conn:

Parameters：

Sou：源神经元层；

Des：目的神经元层；

Weights：网络两层之间的连接权重矩阵。

4.3 面向达尔文 II 的网络权重归整化

本文的脉冲神经网络是面向达尔文 II 芯片的。如第 2 章中所述，达尔文 II 芯片的权重由 8bit 的有符号数表示，能够表示的数值范围为[-128，127]。然而，无论是通过直接训练方法还是间接训练方法得到的脉冲神经网络的权重都是浮点数，因此，为了使其在达尔文 II 芯片上能够运行，需要将浮点数表示的权重转化成为整数表示。因为脉冲神经网络中传递的是脉冲信息，因此，权重的缩放配合超参数电压阈值的调整并不会改变脉冲神经网络的行为。

权重归整化可分为三步。首先是获取权重的绝对值的最大值，如公式 4.22 所示；其次，计算能够进行权重缩放的最大倍数，如公式 4.23 所示；最后选择合适的缩放倍数对网络权重进行缩放取整。

$$w_{\max} = \max(|w_i|) \tag{4.22}$$

$$S = \lfloor 127 / w_{\max} \rfloor \tag{4.23}$$

权重归整化可分为两类，一是对网络的每一层进行权重归整化，二是对网络整体进行归整化，但是这两类归整化的具体实现方法没有区别。对于网络每一层进行的权重归整化，也需要对网络每一层调整电压阈值。通过调整电压阈值，可以使网络的行为基本保持不变。

4.4 面向达尔文 II 的网络仿真

4.4.1 脉冲神经网络构建

一个完整的脉冲神经网络包括神经元模型以及网络结构，同时脉冲神经网络的描述要符合前文的描述规范。

本平台支持用两种方式实现脉冲神经网络的构建。第一种方式是采用平台的框架，由开发者提供网络的连接文件，并指定神经元的类型和网络的一些超参数（超参数见表 3.1），当这些文件和超参数符合要求时，平台会自动构建开发者需要的脉冲神经网络。这种方式可以支持开发者在本平台直接运行自己训练好的网络，并且对不会使用或者不熟悉 Brain2 的开发者非常友好。

网络连接文件表示网络层与层之间的连接，使用三元组表示。文件格式为二进制文件，文件内容为源神经元与目的神经元的编号以及它们之间的连接。具体格式如下，示例如图 4.5。

Connections = [(src1,dst1,weight1,delay1),

(src2,dst2,weight2,delay2),……]

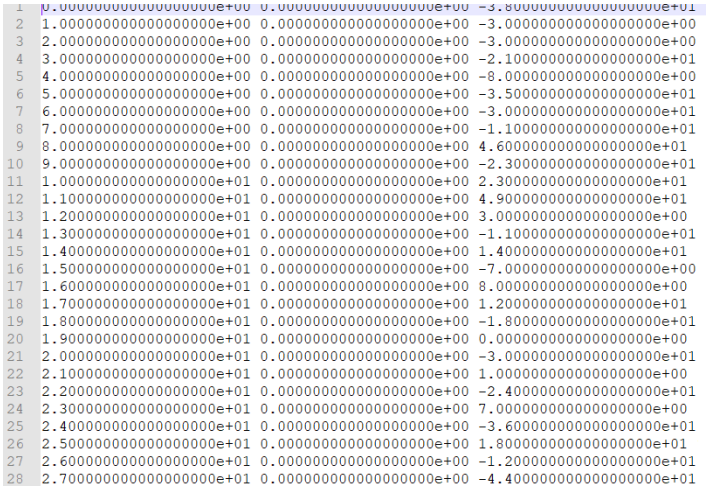

图 4.5 连接文件示意

第二种方式主要针对对 Brain2 有使用经验的开发者，开发者可以直接在代码编辑器中自己编程实现网络的构建。

4.4.2 神经网络到达尔文 II 的优化映射

在进行芯片层面的仿真之前，需要将层次化的神经网络映射到节点化的达尔文 II 芯片上。针对 2.3 中提出的资源分配原则中的 b，在网络映射过程中，我们会优先使用片上存储空间为 32KB 的芯片节点。那么，对于给定的一个脉冲神经网络，它的节点计算会出现两种情况：第一种情况是整个网络需要的节点总数不超过 32KB 的节点数量；第二种情况是整个网络分配需要使用到其他存储空间的节点。

针对这两种情况，我们把芯片划分为 4 个区域，如图 4.6 所示。

区域 ①大小为芯片左上角 16×16 的正方形区域，里面所有的节点均为片上存储为 32KB 的节点；区域 ②为芯片右上角大小为 16×12 的区域，共有 192 个

节点，片上存储为 32KB 和 16KB 的节点分别为 190 个和 32 个；区域 ③为芯片右下角大小为 12×16 的区域，节点组成和区域 ②相同；区域 ④为芯片左下角 12×12 的正方形区域，片上存储为 32KB 和 16KB 的节点分别为 120 个和 24 个。

对于第一种比较简单的情况，网络所需的节点总数不超过 256 个，只需要直接将网络配置到区域 ①的位置。但是当网络的某些层不能配置在区域 ①，或者区域 ①中只能放下网络某一层的一部分时，我们依次使用区域 ②③④直到完成网络的映射，或者发现网络无法映射。

在网络硬件资源评估结束后，我们可以得到网络每一层需要的节点数量。根据计算得到的每一层网络所需的节点数量，包括每种不同片上存储的节点的数量，遵循区域 ①②③④的使用顺序，我们可以得到网络的一个随机映射方案，这样就将按层分布的脉冲神经网络转化成为按芯片节点分布的网络了。

根据第二章中提出的映射原则中的 d，为了使外部的输入数据的传输距离尽量短，本文将网络第一层按如图 4.6 所示的方案进行初始化。图中实线箭头的方向为节点的分配方向，从区域 ①的左边第一列开始依次使用节点，虚线箭头表示这一列节点被分配完，重新开始新一列节点的分配。

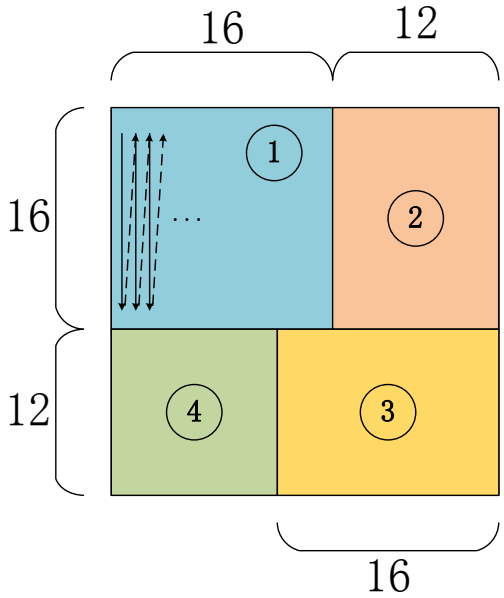

图 4.6 网络第一层初始化方案

在初始化第一层之后，按照从前往后的顺序依次对网络的每一层进行初始化，初始化的原则是与上一层的节点距离尽量短，这样能够保证数据尽快传输并且功耗较低。对于当前需要分配节点的层，在已知前一层网络节点的坐标时，需要在当前区域的剩余未使用节点中找到与前一层网络节点距离最小的节点，距离计算如公式 4.24 所示。

$$dist_i^l = \sum_{j \in node_{l-1}} M_{ji} \tag{4.24}$$

$dist_i^l$ 表示，在剩余可用节点中的第 i 个节点与 l-1 层使用的所有节点的距离之和，M_{ji} 表示节点 j 和节点 i 之间的距离。在后续每一层的分配过程中，每次都会计算剩余可用节点与前一层节点的距离，并根据距离对每一个剩余节点进行排序，选择距离较短的节点，剩余节点的使用示意图如图 4.7，节点中的数字表示节点使用的优先级。按照这种节点分配方式，我们可以得到一个基本能够实时的功耗较低的网络映射方案。

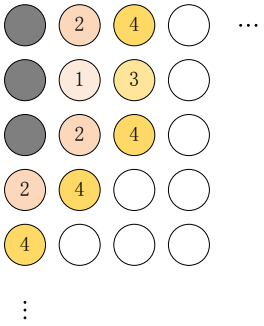

图 4.7 剩余节点的使用顺序

4.4.3 输入激励类（StimulusGroup）

StimulusGroup 用于产生脉冲神经网络的输入激励。与传统的人工神经网络不同，脉冲神经网络是事件驱动型（event-based）的网络，以脉冲传递信息，因此，脉冲神经网络的输入和输出都是一系列的脉冲串。脉冲序列是由离散的脉冲时间点组成的时间序列，经过特定的脉冲序列编码方法，可以将图片、音频等数据或者外界刺激编码成为特定的脉冲序列。

目前脉冲神经网络的输入编码方式主要有频率编码（rate coding）和时序编

码（temporal coding）两种[29]。

频率编码是指将信息编码在脉冲发放的频率上,具有非常高的抗干扰性，但是信息密度较低。最早的的频率编码是平均脉冲发放率编码,即计算时间 t 到 t+Δt 这一段时间内脉冲数的平均值，如公式 3.8 所示,其中 *nsp* 表示在这一段时间内的脉冲发放数量。但是，在实验过程中，通常会进行 k 次实验，使用这 k 次实验总的脉冲发放数量除以实验次数，得到 t 到 t+Δt 这一段时间内的平均脉冲发放数量，然后再除以时间长度以得到脉冲发放率,如公式 3.9 所示，其中 nsp_k 表示 k 次实验中总的脉冲发放数量。由于生物神经元的特性，当受到较强刺激时会发放更多的脉冲，但是，脉冲编码采用了平均值编码，很多细节信息容易被忽略。

$$r(t)=\frac{nsp(t:t+\Delta t)}{\Delta t} \tag{3.8}$$

$$r(t)=\frac{nsp_k(t:t+\Delta t)}{k\Delta t} \tag{3.9}$$

时序编码依赖于每一个脉冲发放的具体时间，将神经元需要传递的信息编码在脉冲发放时间中。与频率编码不同，时序编码可以实现更高级别的信息密度和更高的处理速度，因为每个脉冲计数均无需确定平均值，但是，时序编码的抗干扰性较弱，时间分辨尺度为毫秒级别，一个较小的干扰可能会产生重大影响。由 8ms，18ms，38ms 组成的脉冲序列和由 8ms，18ms，25ms，38ms 组成的脉冲序列是完全不同的。S.Thorpe[30]等人对人类视觉系统中的处理速度进行了一些研究，表明脉冲序列中的第一个峰值可能包含大多数信息。最简单的时序编码是二进制编码，即将 n 个时间点的脉冲序列编码成为由 n 个 0 和 1 组成的二进制串，若脉冲串的第 k 个数字为 1，则表示在神经元在第 k 个时间点发放脉冲。常见的时序编码方式还有等级顺序编码（Rank Order Coding）、相位编码（Phase Coding）等。

本系统同时支持脉冲编码和时序编码，可以实现如下功能：一是可输入给定脉冲序列、固定频率的脉冲序列；二是可实现对数据的编码，包括频率编码和时序编码等常用编码方式。StimulusGroup 的设计如下：

Class StimulusGroup：

Parameters：

Method：‘exact’表示明确设置的脉冲；‘encode’表示需要对数据进行编码。

Sti_method：当 Method 参数为 exact 时有效，表示确切的脉冲种类，可选‘possion（泊松脉冲）’、‘index_time’。

Num：当 Method 参数为 exact 时有效，表示激励神经元的数量。

Freq：当 Method 为 exact 且 Sti_method 为 possion 时有效，表示泊松脉冲的频率。

Indexes：当 Method 为 exact 且 Sti_method 为 index_time 时有效，表示产生脉冲的神经元编号。

Times：当 Method 为 exact 且 Sti_method 为 index_time 时有效，表示产生脉冲的神经元对应的脉冲时间，Times 长度应与 Indexes 长度相同。

Data：当 Method 为 encode 时有效，表示需要编码的数据。

Encode_method：当 Method 为 encode 时有效，表示脉冲编码的方式，可选 random，possion 等。

Maxfreq：当 Methode 为 encode 且 Encode_method 为 possion 时有效，表示脉冲编码的最大频率。

Dur：当 Method 为 encode 且 Encode_method 为 random 时有效，表示脉冲编码的时间。

4.4.4 神经网络仿真

网络的仿真主要包括两方面的仿真：一是软件平台的仿真，二是芯片层面的仿真。

在仿真开始阶段，需要开发者提供网络的输入激励。本平台支持两种网络输

入激励的方式。一种是开发者使用平台提供的 API，指定数据编码方式，包括泊松编码、随机数编码等。在本平台中，随机数编码方式（random）主要如公式 4.25 所示。其中，$rand()$ 函数是一个随机数产生器，能够产生均匀分布的[0，1]之间的随机数，I_i 表示输入的第 i 个数据，为归一化后的数据，c 为一个常数因子。使用一个随机数与输入图片的像素进行比较，当像素值大于随机数时，这个像素点在当前时刻会发放脉冲。

$$rand() < cI_i \tag{4.25}$$

另一种方式为时序编码方式，即开发者直接提供输入激励文件作为仿真模块的输入。输入激励文件用于确定的网络输入表示，需要开发者将数据编码成为一系列的脉冲串。文件格式为二进制文件，文件内容为脉冲神经元 ID 和其对应的脉冲时间序列。具体格式如下，示例如图 4.8 所示：

spikeTrains = [[neuID1,[time_list_1]],

[neuID2,[time_list_2]],……]

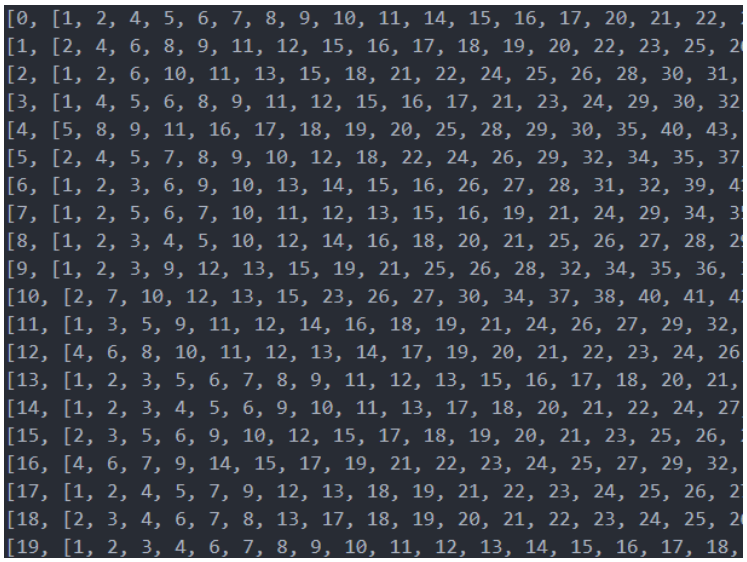

图 4.8 输入文件示例

软件平台的仿真主要是进行一个功能仿真。当构建的脉冲神经网络符合映射要求之后，开发者可使用平台对网络进行软件仿真，通过选择合适的输入数据编码方式和指定网络的一些超参数，开发者可以得到这个脉冲神经网络应用的输

出。这样有助于开发者对网络实现的功能有一个更直观的认识，当发现错误时也能够直接在软件仿真时就解决。

芯片层面的仿真主要是通过软件平台将层次化的神经网络转化成为按节点分布的网络。在映射过程中，会产生对应的网络配置文件。理论上来说，当网络中的超参数相同，网络的输入和开发者选择的输入编码方式也相同时，芯片运行的结果应与软件平台的仿真结果完全一致，若不一致，则需要首先排查芯片的配置是否正确，对芯片进行调试。

由于对于大规模的神经网络应用，达尔文芯片的配置需要较长的时间才能完成，通过软件仿真模块，可以避免用户在网络在芯片上运行出错后再发现错误，这是非常浪费时间的。而通过软硬件同时仿真对比，也可以更快地定位问题出现在网络构建还是芯片运行中。

4.5 本章小结

本章主要提出了基于达尔文II芯片的脉冲神经网络训练方法和仿真。训练方法主要包括直接训练和间接训练。直接训练是指使用STDP和基于SNN的反向传播算法直接训练一个脉冲神经网络，间接训练是指将训练好的人工神经网络转化成为脉冲神经网络。然后，提出一种权值归整化方法将训练好的脉冲神经网络转化成为能够在达尔文II芯片上运行的符合芯片要求的网络。最后，介绍了仿真模块的实现。

第5章 基于仿真训练平台的实验及分析

5.1 仿真训练平台工作流程

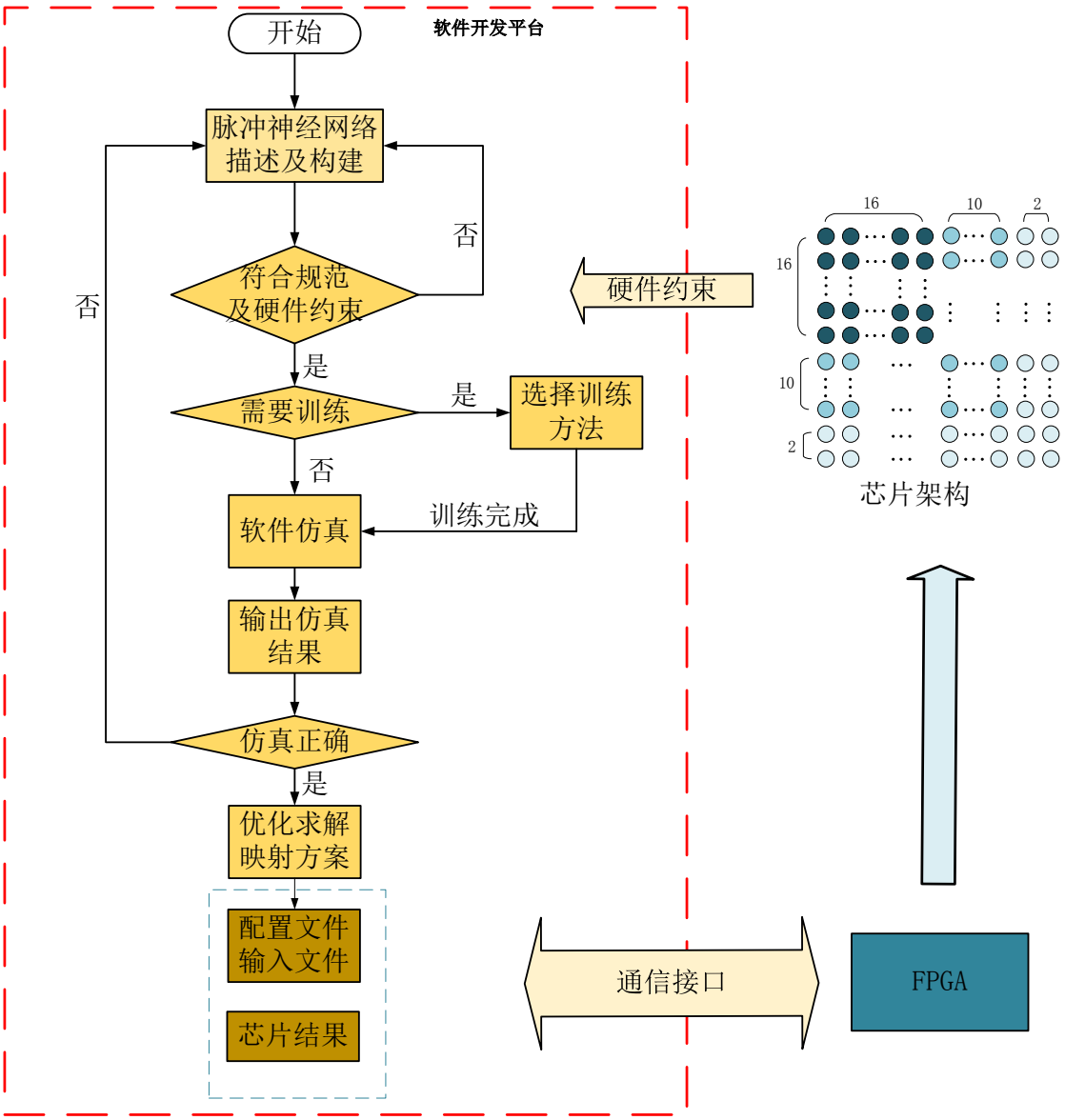

图 5.1 仿真训练平台工作流程

如图 5.1 所示为面向达尔文 II 类脑计算芯片仿真训练平台的工作流程图。首先平台需要一个符合描述规范以及符合硬件约束的脉冲神经网络。当不符合规范或者符合硬件约束时，需要开发者重新定义网络。软件平台支持对脉冲神经网络

的训练，同时也支持直接运行已经训练好的脉冲神经网络。当需要对网络进行训练时，开发者可以选择合适自己网络的训练方法（训练方法在第四章中已有详细说明），目前支持非监督训练算法 STDP 和基于 SNN 的反向传播算法。训练完成后需要对网络进行软件仿真。对于不需要训练的网络，则直接进入软件仿真模块。若仿真结果不正确或者不符合开发者的预期，开发者可以选择重新构建网络或者重新训练网络；若仿真结果正确，则通过平台的自动化映射模块，生成芯片的配置文件和输入文件，进入芯片层面的仿真。平台与芯片通过中间的 FPAG 转发模块进行通信，通过 FPGA 模块将配置文件传输给芯片，芯片即可对网络进行配置。最后将输入数据传入芯片，即可在芯片上运行这个网络，得到网络的运行结果。这个运行结果指的是网络输出层的每个神经元对应的输出脉冲序列，需要开发者自行对这些脉冲输出进行解码，得到需要的结果表示。

5.2 模块测试

如前文所述，整个仿真训练平台可分为脉冲神经网络构建模块、网络训练模块、仿真模块。

脉冲神经网络构建模块主要用于神经网络的构建，包括网络结构、神经元模型选择、网络其他超参数设置。神经元模型选择与其他超参数的设置可以参考表 3.1。在网络结构方面，需要注意的是目前软件平台并不支持循环神经网络的配置运行，因此，在网络结构检查中，主要是针对神经网络中存在循环的情况进行检测。网络中存在循环的情况主要有两种，一是层内的循环，即层内神经元之间有互相连接，二是层间的循环，即后一层网络中的神经元会连向前一层网络。对于检测到的这两种情况，若开发者不作出修改，系统将按照无循环的方式运行。

在本节中，构建了基于手写数字识别的脉冲神经网络，对网络构建模块、网络训练模块和仿真模块进行了具体的测试。

手写数字数据集 MNIST[38]是一个入门级的图片分类数据集，共包括 0 到 9 这十个数字（如图 5.2 所示）的 70000 张手写图片，其中 6000 张为训练数据，10000 张为测试数据。

图 5.2 MNIST 手写数字集图片

5.2.1 脉冲神经网络构建模块

本文构建了一个卷积脉冲神经网络，即网络的结构与卷积神经网络相同，如图 5.3 所示。网络由两个卷积层、两个池化层和一个全链接层组成。两个卷积层的卷积核为 8×3×3 和 16×3×3，两个池化层为平均池化，池化层的核大小为 2×2，最后的输出层包括 10 个神经元，分别代表 0 到 9 这十个类别。本节分别对两种脉冲神经网络构建的方法进行了测试，两种方式都必须符合第三章中提出的网络描述规范。

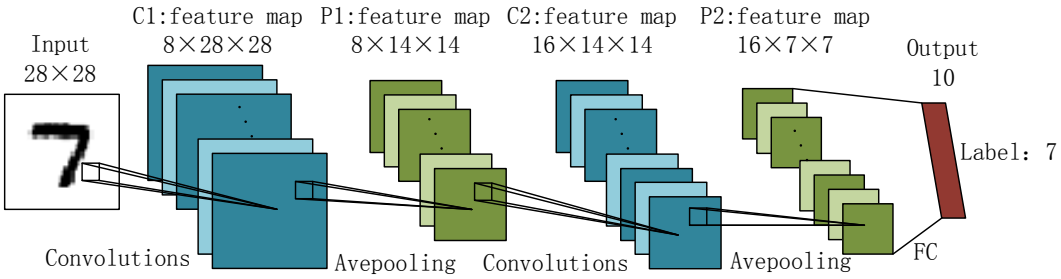

图 5.3 手写数字识别网络结构图

如图 5.4 所示为开发者通过提供网络结构文件构建如图 5.3 所示的脉冲神经网络。在图 5.4 中，（a）部分主要是指定神经元模型的各种参数，可以看出选择的是 DarwinI 模型，并且膜电位不衰减；（b）部分主要指定网络的参数，包括网络层数，每层网络的神经元数量、阈值电压和延时；（c）部分为开发者提供的符合规范的网络连接文件，平台通过解析文件能够构建对应结构的网络。

```
start
choose neuron type: 1 for DarwinI, 0 for DarwinII
0
set reset mode: 0 for reset to zero, 1 for reset to (v-vth)
0
set leaksign: -1 for IF, 1 for LIF
-1
set leak
0
```
（a）

```
input netDepth:
6
input layerWidth:
784,6272,1568,3136,784,10
set vth for every layer
200,200,200,200,200
set deley: choose 0 or 1 or 2
1
```
（b）

```
input connection files' path
in_conv1,conv1_pool1,pool1_conv2,conv2_pool2,pool2_out
network construction done
```
（c）

图 5.4 通过提供网络结构文件构建网络

```
net = Network()
Conv_1 = NeuronGroup(Num=[8, 28, 28], Neuontype='DarwinI',
        Reset_mode=0, leaksign=-1, leak=0,threshold=200,delay=1)
Pool_1 = NeuronGroup(Num=[8, 14, 14], Neurontype='DarwinI',
        Reset_mode=0, leaksign=-1, leak=0,threshold=200,delay=1)
Conv_2 = NeuronGroup(Num=[16, 14, 14], Neurontype='DarwinI',
        Reset_mode=0, leaksign=-1, leak=0,threshold=200,delay=1)
Pool_2 = NeuronGroup(Num=[16, 7, 7], Neurontype='DarwinI',
        Reset_mode=0, leaksign=-1, leak=0,threshold=200,delay=1)
Output = NeuronGroup(Num=10, Neurontype='DarwinI',
        Reset_mode=0, leaksign=-1, leak=0,threshold=200,delay=1)
net.add([Conv_1, Pool_1, Conv_2, Pool_2, Output])
```
（a）

```
S1 = Synapse(model='volt', on_pre='v_post += w')
Input_to_Conv1 = gen_conv_conn(sou=Input, des=Conv1, filters=8, kernel_size=[3, 3],
        stride=[1, 1, 1, 1],padding='valid', weight=weight['w_conv1'], synapse_type=S1)
Conv1_to_Pool1 = gen_avepooling_conn(sou=Conv1, des=Pool1, kernel_size=[2, 2],
        stride=[1, 2, 2, 1], synapse_type=S1)
Pool1_to_Conv2 = gen_conv_conn(sou=Pool1, des=Conv2, filters=16, kernel_size=[3, 3],
        stride=[1, 1, 1, 1], padding='valid', weight=weight['w_conv2'], synapse_type=S1)
Conv2_to_Pool2 = gen_avepooling_conn(sou=Conv2, des=Pool2, kernel_size=[2, 2],
        stride=[1, 2, 2, 1], synapse_type=S1)
Pool2_to_Output = gen_fc_conn(sou=Pool2, des=Output, weight=weight['w_fc'], synapse_type=S1)
net.add([Input_to_Conv1, Conv1_to_Pool1, Pool1_to_Conv2, Conv2_to_Pool2, Pool2_to_Output])
```
（b）

图 5.5 编程实现网络构建

图 5.5 所示为编程实现网络构建，其中（a）部分表示每一层神经元的构建，包括每一层的神经元数量以及其余超参数；（b）部分表示网络结构的构建。

5.2.2 脉冲神经网络训练模块

如第四章所述，本平台支持的脉冲神经网络训练算法主要分为两类：一是直接训练，二是间接训练训练。其中，直接训练算法主要包括无监督的 STDP 算法和基于 SNN 的反向传播算法。间接训练算法是指先训练裁剪后的 ANN，然后保持网络结构不变，将权重归整化到 8 位有符号整数，将其转化成为 SNN。

因此，本文首先对相同网络结构的卷积神经网络进行训练，学习率为 0.005，训练 epoch 为 40 次，得到训练好的权重。然后采用第四章中描述的转化方法，对网络中的结构参数进行提取转化，将网络转化成为脉冲神经网络。

其次，分别使用 STDP 和反向传播算法对脉冲神经网络进行训练。STDP 的连接增强和连接减弱学习率分别为 0.006 和 0.003，时间窗口常数设为 10ms，阈值电压设为 5；在反向传播算法中，时间窗口常数设为 20ms，学习率为 0.01，侧向抑制常数为 0.03，阈值电压为 17。对于输入图片的编码，两种训练算法都采用随机数编码，将归一化后的图片像素值作为参考值，网络运行时间为 100ms。

表 5.1 不同训练算法的准确率

算法	准确率	权重归整化后的准确率
卷积神经网络	99.92%	--
间接训练算法	98.91%	97.92%
STDP	86.74%	84.31%
反向传播算法	97.51%	95.47%

如表 5.1 所示为不同训练算法的准确率对比以及将权重归整化后的准确率对比。直接训练的卷积神经网络的准确率为 99.92%，将网络转化成为脉冲神经网络后，可以发现网络的准确率有所下降，约为 98.91%，造成这种现象的原因主要是单向衰减的 Darwin 神经元模型的脉冲发放率并不能完全表示 ANN 中神经元的激活值。ANN 到 SNN 转化可行的理论基础是使用脉冲神经元的脉冲发放率来模拟人工神经元的激活值，但是，通过转化方法，调整脉冲神经网络的各种超参数，

并不能精确地表示 ANN 中的激活值。

对于所有的训练算法，在进行权重归整化操作后，网络的分类准确率都有所下降。这主要是权重归整化操作带来的准确率下降。经过权重归整化操作后，部分神经元的脉冲发放率有可能会被改变，不需要发放脉冲的神经元可能会有少量的脉冲发放，需要发放脉冲的神经元的脉冲发放率可能会增高或者降低。但是，这种改变并没有对网络的整体功能造成非常大的影响，网络精度损失并不大，在可接受的范围内。

在三种训练算法中，间接训练算法和反向传播的准确率较高，STDP 的准确率较低。虽然反向传播算法并不能用生物科学的角度解释，但是在应用实现中还是能达到很好的效果。STDP 算法的准确率最低，只有 86.74%，可能是因为本平台实现的是最基础的而且是简化过后的 STDP，对数据的学习能力没有那么强。

5.2.3 脉冲神经网络仿真模块

仿真模块主要是对网络的功能进行测试。在这部分的测试中，采用了由卷积神经网络转化而来的脉冲神经网络。

网络的激励由输入激励类产生，选择的输入编码方式为 random 编码，运行时间为 100ms，如图 5.6 所示。

```
# 创建激励
Input = StimulusGroup(Method='encode', Encode_method='randon', Num=[28, 28], Data=image,
                      Maxfreq=1000 * Hz, Dur=100 * ms)
# 将激励加入到网络中
net.add(Input)
```

图 5.6 网络激励产生

在仿真测试中，通过输出层每一个神经元输出的脉冲数量来判断图片的类别，输出脉冲数量最多的神经元的编号即为图片识别结果。如图 5.7 所示为软件仿真结果的界面展示，图片的标签为 7，网络的识别结果也为 7。

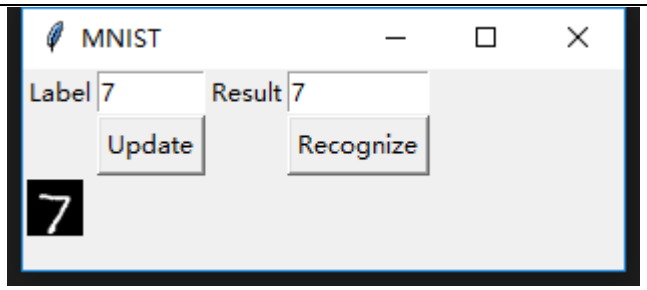

图 5.7 软件仿真结果展示

在进行芯片层面仿真之前，需要对网络进行映射，映射原则与方法在前文中已有说明。通过计算可知网络能够在片上存储空间为 32KB 的节点中完成配置，每一层所需的节点数量依次分别为 25 个、15 个、13 个、4 个和 1 个，节点分配如图 5.8 所示。

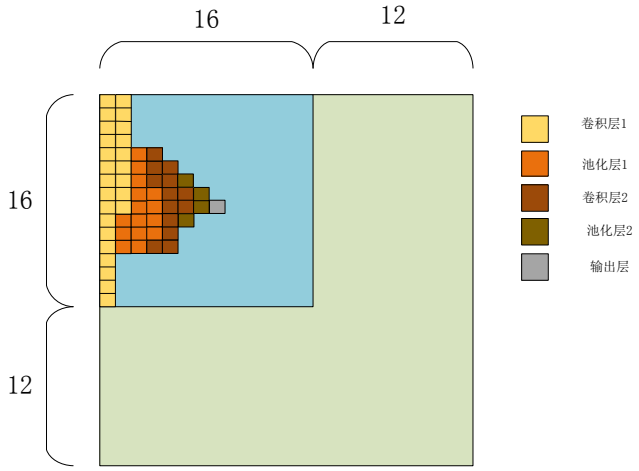

图 5.8 网络映射图

图 5.9 为芯片层面仿真的输出结果，通过对其解析，能得到与软件仿真相同的结果。

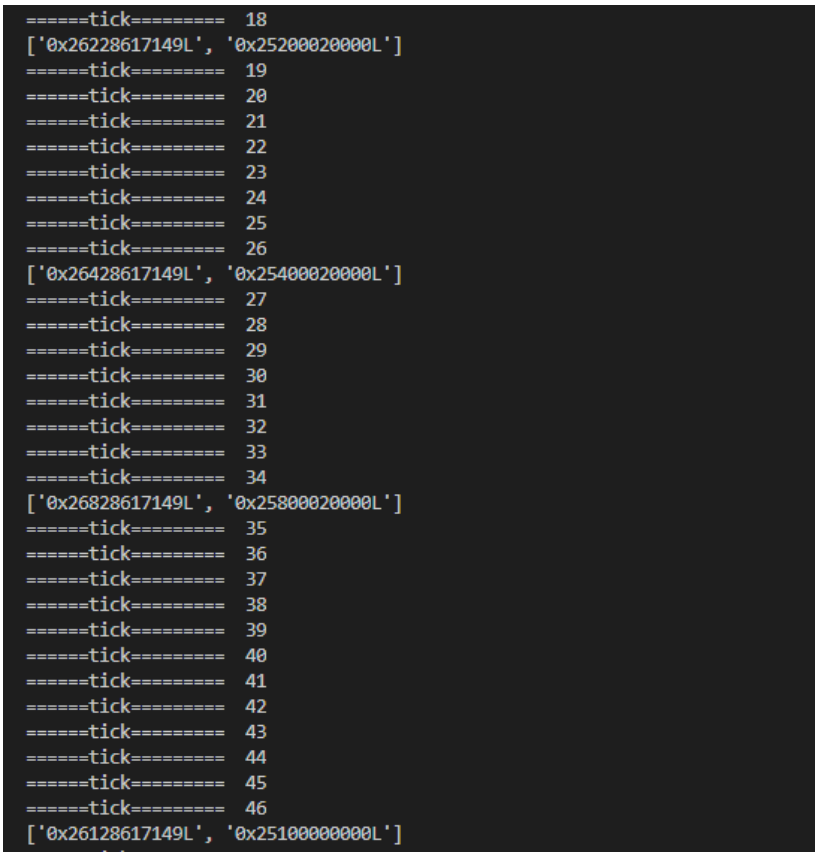

图 5.9 芯片仿真结果

5.3 猜拳应用整体分析

猜拳应用主要是实现了人与计算机的石头剪刀布，无论人出石头、剪刀或者布，计算机总能实时地出一个能够赢过人的手势。

应用的数据集来源于我们自己采集的手势图片，如图 5.10 所示。共三类手势，手势采集的背景为黑色背景，每一类手势分别采集了 5000 张图片，包括不同姿势的手势和不同人的手势。其中训练集和测试集的比例为 4：1， 也就是说，每一类有 4000 张图片用于训练，1000 张图片用于测试。

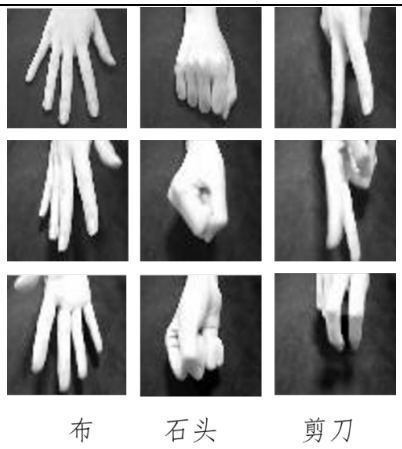

图 5.10 猜拳应用数据集

本文构建了一个全连接的网络用于手势识别的分类，如图 5.11 所示。输入图片为 64×64 的灰度图像，除输入层外，网络还有两层隐藏层和一层输出层。对于这个网络，采用了间接训练的方式，将其从 ANN 转化成为 SNN。在网络映射中，隐藏层 1 需要映射到 32 个片上存储为 32KB 的节点上，隐藏层 2 需要映射到 2 个节点上，输出层需要一个节点。对图片的脉冲编码依旧使用随机数编码，网络运行时间为 100 的单位时间，神经元模型为不衰减的 DarwinI 模型。

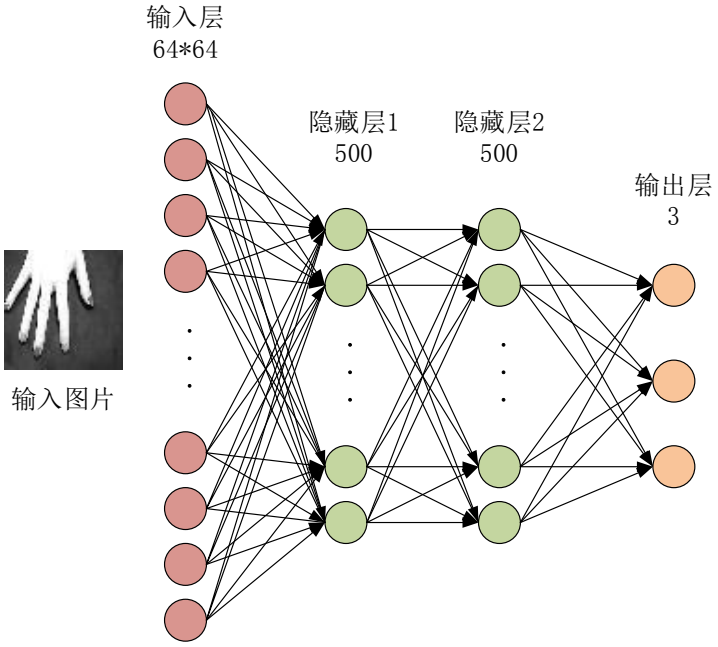

图 5.11 猜拳应用网络结构

整个应用的实验现场如图 5.12 所示，包括仿真训练软件平台、达尔文 II 芯片平台、摄像头采集模块等。摄像头采集模块会不停地采集实验者的手势信息，当实验者的手势发生变化时，会将捕捉到的手势图，通过随机数编码的方式，编码成为一系列的脉冲输入仿真训练平台，平台通过 FPGA 将输入的脉冲激励发送给芯片。当芯片接收到激励时，网络开始运行，运行结束后将芯片的输出的结果返回给仿真训练平台。平台通过解析返回的结果，并将能够赢过实验者的手势实时地输出到结果展示界面。通过这个应用，也证明了本文提出的仿真训练平台的可用性。

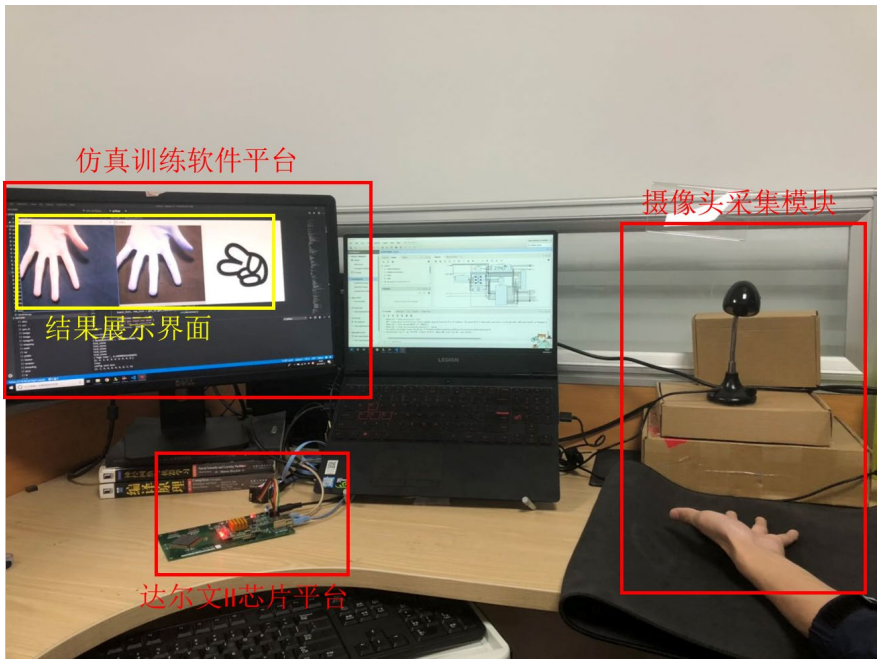

图 5.12 实验现场图

5.4 本章小结

本章首先对仿真训练平台的总体运行流程进行了说明。然后，将平台分为脉冲神经网络构建模块、网络训练模块、仿真模块等模块，使用基于卷积的脉冲神经网络的手写数字识别例子，对平台进行模块测试。最后，基于此平台，完成了一个实时的猜拳应用，证明了此软件开发平台的可用性。

第6章 总结与展望

6.1 总结

随着人工智能发展的日趋深入，云计算、物联网、大数据等技术的持续突破，数据量和数据处理量都以指数倍增长，传统的存储器和中央处理器分离的冯诺伊曼框架已经越来越不能满足高效计算的需求，因此，基于非冯诺伊曼架构的类脑计算成为了新的选择。

目前业界已有很多款相对较成熟的类脑计算芯片以及其配套的软件开发平台。但是针对基于脉冲神经网络的类脑计算芯片来说，每一种软件平台都有其独特性，是针对某一块芯片的具体架构开发的，并没有一个通用平台适应不同芯片架构的开发。因此，面向达尔文 II 芯片这款类脑计算芯片，本文设计了适合其应用开发的仿真训练平台，能够避免开发者直接面对芯片硬件的架构和约束，直接方便地进行应用开发。

本文从脉冲神经网络的构建、训练和仿真三方面出发，研究了仿真训练平台的实现。整个平台共由四个主要模块构成：脉冲神经网络标准化模型库、仿真训练平台、硬件资源和网络优化映射平台，分别用于实现不同功能。本文的主要成果有以下三部分：

一是提出了面向达尔文 II 类脑计算芯片的脉冲神经网络规范化描述方法，构建了软硬件协同仿真模型库，这一部分主要在脉冲神经网络标准化模型库中实现，包括 Darwin 神经元模型的构建、网络结构以及其超参数的标准化描述等。这一部分统一了脉冲神经网络的构建标准。

二是在脉冲神经网络训练方面，同时支持直接训练方法和间接训练方法。其中，直接训练方法包括非监督算法--STDP 和监督算法--基于 SNN 的反向传播算法；间接训练方法是指将训练完成的 ANN 在保持权重和网络结构不变的情况下，使用脉冲神经元替换人工神经元，将其转化成为脉冲神经网络的方法。多种训练方法能够满足不同脉冲神经网络训练的需求。

三是在网络仿真阶段，采用软件仿真和芯片层面仿真相结合的方法。软件仿真主要是对脉冲神经网络应用的功能进行仿真，并对芯片仿真结果提供参考标准。芯片仿真结果应与软件仿真结果完全相同，通过与软件仿真结果的对比，也可发现芯片仿真是否发生错误。通过这两种仿真方法的结合使用，开发者能够更加准确和直观地评估应用效能的同时大幅度提高仿真速度。

6.2 展望

目前本平台实现了从脉冲神经网络构建到芯片运行返回结果的整体流程，但是在这过程中还有许多可以优化的地方。目前网络输入数据的编码和输入芯片的过程还是比较耗时的一个部分，未来可以提出更优化的数据编码方案。在网络训练方面，主要是基于软件平台完成脉冲神经网络训练，将来可以考虑直接在芯片中完成网络的训练。

参考文献

[1] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484.

[2] Jr J W K . Mapping the brain and its functions: Integrating enabling technologies into neuroscience research[J]. 1969.

[3] Bargmann C I, Newsome W T. The brain research through advancing innovative neurotechnologies (BRAIN) initiative and neurology[J]. JAMA neurology, 2014, 71(6): 675-676.

[4] Amunts K. The human brain project: neuroscience perspectives and German contributions[J]. e-Neuroforum, 2014, 20(2): 43-50.

[5] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1026-1034.

[6] Monroe D. Neuromorphic computing gets ready for the (really) big time[J]. Communications of the ACM, 2014, 57(6): 13-15.

[7] Cardarilli G C, Cristini A, Di Nunzio L, et al. Spiking neural networks based on LIF with latency: Simulation and synchronization effects[C]//2013 Asilomar Conference on Signals, Systems and Computers. IEEE, 2013: 1838-1842.

[8] 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.

[9] Neil D, Pfeiffer M, Liu S C. Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks[C]//ACM Symposium on Applied Computing. Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016.

[10] Liu D, Chen T, Liu S, et al. Pudiannao: A polyvalent machine learning accelerator[C]//ACM SIGARCH Computer Architecture News. ACM, 2015, 43(1): 369-381.

[11] Kim Y, Zhang Y, Li P. A reconfigurable digital neuromorphic processor with

memristive synaptic crossbar for cognitive computing[J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2015, 11(4): 38.

[12] 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.

[13] Hong B, Cho J, Kim B, et al. A study on supporting spiking neural network models based on multiple neuromorphic processors[C]//Proceedings of the Conference on Research in Adaptive and Convergent Systems. 2019: 131-132.

[14] 黄铁军, 施路平, 唐华锦, 等. 多媒体技术研究: 2015——类脑计算的研究进展与发展趋势[J]. 2016.

[15] Chicca E, Stefanini F, Bartolozzi C, et al. Neuromorphic electronic circuits for building autonomous cognitive systems[J]. Proceedings of the IEEE, 2014, 102(9): 1367-1388.

[16] Akopyan F, Sawada J, Cassidy A, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(10): 1537-1557.

[17] 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.

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

[19] Shen J, Ma D, Gu Z, et al. Darwin: a neuromorphic hardware co-processor based on spiking neural networks[J]. Science China Information Sciences, 2016, 59(2): 1-5.

[20] Amir A, Datta P, Risk W P, et al. Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores[C]//The 2013 International Joint Conference on Neural Networks (IJCNN). IEEE, 2013: 1-10.

[21] 王雨辰, 胡华. 类脑计算新发展——“TrueNorth”神经元芯片[J]. 计算机科学, 2016, 1.

[22] Lin C K, Wild A, Chinya G N, et al. Programming spiking neural networks on Intel’s Loihi[J]. Computer, 2018, 51(3): 52-61.

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

[24] Dayan P, Abbott L F. Theoretical neuroscience[M]. Cambridge, MA: MIT Press, 2001.

[25] Morris C, Lecar H. Voltage oscillations in the barnacle giant muscle fiber[J]. Biophysical journal, 1981, 35(1): 193-213.

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

[27] Gerstner W. A framework for spiking neuron models: The spike response model[M]//Handbook of Biological Physics. North-Holland, 2001, 4: 469-516.

[28] Cao Y, Chen Y, Khosla D. Spiking deep convolutional neural networks for energy-efficient object recognition[J]. International Journal of Computer Vision, 2015, 113(1): 54-66.

[29] Krause T U, Würtz P D D R. Rate coding and temporal coding in a neural network[D]. M. Sc. thesis, Electrical Engineering, Univ. of Bochum, Germany, 2014.

[30] Thorpe S, Fize D, Marlot C. Speed of processing in the human visual system[J]. nature, 1996, 381(6582): 520.

[31] 陶建华,陈云霁. 类脑计算芯片与类脑智能机器人发展现状与思考[J]. 中国科学院院刊(7):803-811.

[32] Markram H, Lübke J, Frotscher M, et al. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs[J]. Science, 1997, 275(5297): 213-215.

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

[34] Hecht-Nielsen R. Theory of the backpropagation neural network[M]//Neural networks for perception. Academic Press, 1992: 65-93.

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

[36] Lee J H, Delbruck T, Pfeiffer M. Training deep spiking neural networks using

backpropagation[J]. Frontiers in neuroscience, 2016, 10: 508.

[37] Diehl P U, Zarrella G, Cassidy A, et al. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware[C]//2016 IEEE International Conference on Rebooting Computing (ICRC). IEEE, 2016: 1-8.

[38] Lecun Y L , Bottou L , Bengio Y , et al. Gradient-Based Learning Applied to Document Recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

攻读硕士学位期间主要的研究成果

致谢