A review of basic software for brain-inspired computing

Article *in* CCF Transactions on High Performance Computing · March 2022

DOI: 10.1007/s42514-022-00092-1

Peng Qu
Tsinghua University
13 PUBLICATIONS **343** CITATIONS
SEE PROFILE

Le Yang
Tsinghua University
4 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Youhui Zhang
Tsinghua University
115 PUBLICATIONS **2,282** CITATIONS
SEE PROFILE

CCF Transactions on High Performance Computing
https://doi.org/10.1007/s42514-022-00092-1

REGULAR PAPER

A review of basic software for brain-inspired computing

Peng Qu[1,2] · Le Yang[1] · Weimin Zheng[1] · Youhui Zhang[1]

Received: 15 July 2021 / Accepted: 23 February 2022

Abstract
Brain-inspired computing, which is inspired by the information processing procedure and the biophysiological structure of the brain, is believed to have the potential to drive the next wave of computer engineering and provide a promising way for the next generation of artificial intelligence. The basic software for brain-inspired computing is the core link to realize the research goals of brain-inspired computing and build the ecological environment of brain-inspired computing applications. This paper reviews the status of the three major kinds of basic software for brain-inspired computing. Namely, the toolchains for neuromorphic chips, the software simulation frameworks, and the frameworks that integrate spiking neural networks (SNNs) and deep neural networks (DNNs). Afterward, we point out that a "general-purpose" hierarchical and HW/SW decoupled basic software framework would be beneficial to both the (computational) neuroscience and brain-inspired intelligence fields. And the notion “general-purpose” refers to the decoupling of software and hardware and supports the integration of computer science and neuroscience related research.

Keywords Brain-inspired computing · Basic software · Neuromorphic toolchains · SNN simulation · Decoupling software and hardware

1 Introduction

Brain-inspired computing is often used to refer to the computational theory, architecture, hardware design, and even models and algorithms that are inspired by the information processing procedure and biophysiological structure of the brain. At present, numerical simulations based on brain-inspired computing have become one of the most important research methods in the neuroscience field, after experimental and theoretical methods (Jordan et al. 2018). In the field of artificial intelligence, brain-inspired computing is also considered to be one of the promising paths to the next generation of artificial intelligence (Marblestone et al. 2016; Hassabis et al. 2017; Pei et al. 2019; Roy et al. 2019; Richards et al. 2019). And brain-inspired architecture design is also one of the main development directions of computer architecture in the post Moore Law era (Waldrop 2016).

The basic software for brain-inspired computing is the core link to realize the research goals of brain-inspired computing and build the ecological environment of brain-inspired computing applications (Zhang et al. 2020). The basic software for brain-inspired computing needs to meet the development requirements of various applications in this interdisciplinary field and needs to efficiently drive all kinds of brain-inspired computing chips.

For neuroscience and computational neuroscience researchers, the features of the basic software for brain-inspired computing that they focus on include the efficient and reasonable analysis and interpretation of experimental data to build computational neural models (from single neuron to large-scale neural circuits), and the use of biologically-feasible plasticity rules to optimize them (the main computing paradigm in the brain-inspired computing is spiking neural network, or SNN for short), in order to improve the understanding and knowledge of the infrastructure and underlying mechanism of biological intelligence (Zhang and Xu 2021).

✉ Peng Qu
shen_yhx@163.com

Youhui Zhang
zyh02@mail.tsinghua.edu.cn

1 Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China

2 State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi, Jiangsu, China

For brain-inspired intelligence application researchers and developers, the optimization methods they prefer to are not limited to biologically feasible, so they pay attention to the integration of current effective technologies in the field of deep learning (such as the error backpropagation algorithm and the variants) into SNN, which is called D-SNN (i.e. deep spiking neural network), and attaches importance to the hybrid modeling and application of deep neural network (DNN) and SNN (Pei et al. 2020; Marblestone et al. 2016). That is, they pursue excellent computing performance and application intelligence.

The common needs of these two types of R&D personnel include convenient and accurate neural network modeling and description, efficient simulation (or running) of neural networks that give full play to the potential of the underlying hardware (Brette et al. 2007; Hazan et al. 2018; Yavuz et al. 2016). At the same time, they pay attention to SNN optimization methods of multi-level abstraction and multi-scale plasticity, so as to continuously draw new inspirations from the development of neuroscience, which in turn is conducive to a better understanding of the biological nervous system.

Last but not least, for brain-inspired computing chip designers and developers, basic software for brain-inspired computing plays three roles. First, as an intermediate layer between the hardware and applications, it is necessary to make the constraints of brain-inspired hardware as "transparent" to developers as possible, so that the former can support all kinds of brain-inspired computing applications to the greatest extent. Second, the back-end interfaces of basic software to various types of hardware should be flexible and fully reflect the needs and characteristics of brain-inspired computing, which is a basis for the optimization and adaptation of various brain-inspired chips. Finally, the basic software for brain-inspired computing can provide key and concise computing features, memory access patterns, and temporal features of applications for hardware design and is the prerequisite for the codesign of brain-inspired software and hardware.

The rest of this paper is organized as follows: Sect. 2 outlines the three types of the basic software for brain-inspired computing. Section 3 discusses the neuromorphic chips and the toolchains of them in detail. Section 4 talks about the software simulation frameworks. Section 5 shows the emerging work that integrates SNNs and DNNs. Finally, we analyze the status quo of the basic software for brain-inspired computing and the development forecast in Sect. 6.

2 Three types of the basic software for brain-inspired computing

The most widely used computing model in brain-inspired computing is spiking neural network, and its major computing procedure is as follows: firstly, we need to solve the dynamic equations of the membrane potential which is usually a differential equation to get the current value of the membrane potential. Secondly, if the membrane potential reaches the threshold voltage, the neuron will issue a spike, and the voltage will stay at the reset voltage for a refractory period if necessary. Thirdly, we need to deliver the issued spikes to their target synapses according to the connectionism. And finally, the synapses that received spikes will update the internal state of their target neurons according to the synapse model. The neuron models define the dynamic of the membrane potential and other internal states. There are quite a few neuron models, such as leaky-integrate-and-fire (LIF) (Burkitt 2006), Izhikevich (2003), Hodgkin and Huxley (1990), and so on (Brette et al. 2007). Different models provide different levels of biological reality and computational efficiency.

Take the traditional LIF model as an example, the membrane potential of typical LIF model can be defined as:

$$C_m \frac{dv(t)}{dt} = -\frac{C_m}{\tau_m}\left[v(t) - V_{reset}\right] + i_e(t) + i_i(t) + I_{offset} + i_{injection}(t)$$

Here, $v(t)$ is the membrane potential of the neuron, C_m is the neuron capacity, τ_m is the membrane time constant, V_{reset} is the resistant voltage, $i_e(t)/i_i(t)$ is the excitatory/inhibitory input currents, I_{offset} is the constant input currents, and $i_{injection}(t)$ is the dynamic input currents. The input currents are defined as:

$$\begin{cases} i_e(t) = C_m \sum_{k=1}^{N_e} W_{e,k} S_{e,k}(t) \\ i_i(t) = C_m \sum_{k=1}^{N_i} W_{i,k} S_{i,k}(t) \end{cases}$$

Here, $W_{e,k}/W_{i,k}$ is the synaptic weight of the kth excitatory/inhibitory synapse while $S_{e,k}/S_{i,k}$ is the corresponding input spike.

For simple models, such as IF and some LIF variants, we can solve the equations of the membrane potentials and get the analytical solutions. But for more complicated models, numerical methods are necessary. The forward Euler (Carnevale and Hines 2006) method which is rather simple while providing acceptable accuracy is used by most software and hardware implementations. Some software simulators even use more complicated methods, such as backward Euler and Crank–Nicholson methods (Carnevale and Hines 2006) to provide more accurate and stable numerical solutions.

Solving the dynamic equations of the membrane potential (including calculating the input currents) is often complicated and time-consuming. Moreover, due to the sparsity of spikes, the computation in the whole procedure is usually sparse, event-driven, and may have irregular memory

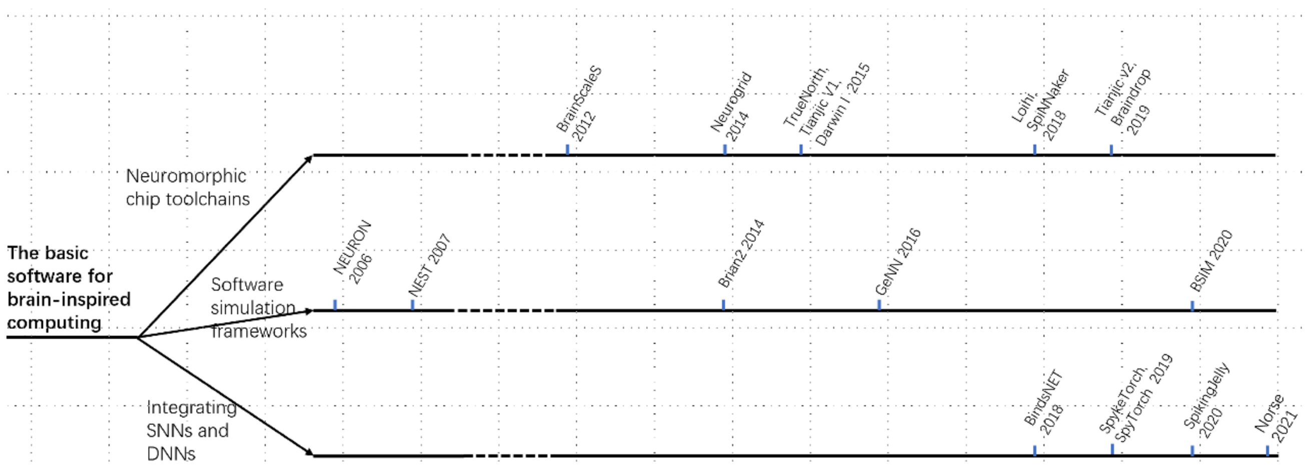
The basic software for brain-inspired computing
Neuromorphic chip toolchains
Software simulation frameworks
Integrating SNNs and DNNs
BrainScaleS 2012
Neurogrid 2014
TrueNorth, Tianjic V1, Darwin I 2015
Loihi, SpiNNaker 2018
Tianjic v2, Braindrop 2019
NEURON 2006
NEST 2007
Brian2 2014
GeNN 2016
BSIM 2020
BindsNET 2018
SpykeTorch, SpyTorch 2019
SpikingJelly 2020
Norse 2021

Fig. 1 Three types of basic software for brain-inspired computing

access patterns. Thus, several dedicated hardware/software implementations are proposed to efficiently simulate SNNs.

According to the different development sources and paths, the basic software for brain-inspired computing can be divided into three categories (Fig. 1):

The first is to develop neuromorphic chips and provide their own software toolchains, including IBM's TrueNorth chip (Akopyan et al. 2015), Intel's Loihi chip (Davies et al. 2018), SpiNNaker (Brown et al. 2018), and BrainScaleS system (Schemmel et al. 2012) (both are supported by the Human Brain Project[1]), and Stanford University's NeuroGrid (Benjamin et al. 2014), etc.

This type of research often uses end-to-end software and hardware co-design solutions, making toolchains and even application models specific to the target chips. In addition, there have been emerging research attempts to avoid being bound to a certain type of chip by building a "general-purpose" domain-specific language or development framework that connects software and hardware (Zhang et al. 2020; Davision et al. 2009; Richmond et al. 2014; Aimone et al. 2019a, b; Lagorce and Ryad 2015). Such research is generally in the early stages of design; because of many unsolved problems, there is a lack of general support for actual chips.

The second is SNN simulation frameworks derived from the field of computational neuroscience, with the goal of understanding biological systems, including NEST (Gewaltig and Diesmann 2007), NEURON (Carnevale and Hines 2006), Brain2 (Stimberg et al. 2014), GeNN (Yavuz et al. 2016), etc.

The characteristics of this type of work are to support the description of the dynamic process of more detailed neuronal activities, support multiple types of synaptic models and multiple types of synaptic plasticity, so as to be able to simulate biological neural networks in detail, while they require users to have a certain degree of the foundation of computational neuroscience.

Compared with the development frameworks widely used in the field of deep learning, the usage is more complicated: the frameworks are usually developed in C language, resulting in a lack of cross-platform capabilities; they lack support for the in-depth optimization of various back-end hardware, including general-purpose graphics processing units (GPGPUs) that are widely used in numerical simulations (Qu et al. 2020).

The third is the work (Hazan et al. 2018; Pehle and Pedersen 2021; Mozafari et al. 2019) that has emerged in recent years to integrate SNN representation/computation features into the widely used deep learning development framework (Paszke et al. 2019). The main audience of this kind of work is brain-inspired intelligent application developers, and the purpose is to integrate the convenience of deep learning framework development (including the BP algorithm and its variants) with the characteristics of SNN and make full use of various resources in the deep learning field.

This kind of work is basically in the early stage, and there is still a long way to go to expand the deep learning framework from the perspective of function enhancement and performance optimization. Some studies (Gidon et al. 2020; Li et al. 2020) have found that the detailed structures (such as dendrites) in biological neurons are not just for connection, they can perform operations, that is, the neuron itself may also be a multi-layer network, which requires the ability to build the fine dynamics model of synapses and is a potential challenge for the existing framework's modeling method based on tensors and tensors' computing. Moreover, the backends of existing deep learning frameworks [including the neural network compilers (Chen et al. 2018)] are often

[1] https://www.humanbrainproject.eu.

designed for deep learning acceleration chips. Whether it is suitable for brain-inspired chips is an open question.

3 Neuromorphic chips and the toolchains of them

To fully exploit the computing power of brain-inspired computing, quite a few neuromorphic chips are proposed (Akopyan et al. 2015; Davies et al. 2018; Brown et al. 2018; Schemmel et al. 2012; Benjiamin et al. 2014; Neckar et al. 2019; Pei et al. 2019), and most of them also provide their own software toolchains (Amir et al. 2013; Lin et al. 2018; Rhodes et al. 2018; Müller et al. 2020; Deng et al. 2020).

TrueNorth (Akopyan et al. 2015) is a synchronous and asynchronous hybrid fully digital neuromorphic chip produced by IBM. It consists of 540 million transistors which provide 4096 neurosynaptic cores, each containing 256 neurons and 64,000 synapses. And it can support about 1 million neurons and 256 million synapses in total. The chip has a peak computing performance of 58 GSOPS (giga-synaptic operations per second) and a peak computing power consumption of 400 (GSOPS/W). The neuron model used by TrueNorth is a simplified LIF model (Cassidy et al. 2013). To fully exploit the potential of TrueNorth, IBM developed a series of specific tools. The most important tools are Compass (Preissl et al. 2012), a simulator of the TrueNorth architecture, and Corelet (Amir et al. 2013), a new programming paradigm that consists of a program abstraction, an object-oriented language, a library, and an end-to-end programming environment. These tools make it possible to program with the models and connection patterns used by TrueNorth directly and they can compile the programs and map them to the chip afterward.

Loihi (Davies et al. 2018) is an asynchronous neuromorphic chip that supports synaptic plastic policy produced by Intel. It provides the support of several important features such as hierarchical connectivity, dendritic compartments, synaptic delays, programmable synaptic learning rules, and so on. Loihi is a grid-like multi-core spiking neural network processor, and each chip contains 128 neuromorphic cores, 3 X86 processor cores, an off-chip interface for expansion, and an asynchronous network-on-chip for message transmission. It has 2.07 billion transistors and 33 MB SRAM in total. Each neuromorphic core can simulate 1024 neural parts (dendrites or somas) through time-division multiplexing. The Intel Lab also provides the Loihi toolchain (Lin et al. 2018), which consists of three parts: an intuitive Python-based API for specifying SNNs, a compiler, and runtime for building and executing SNNs on Loihi and several target platforms (Loihi silicon, FPGA, and functional simulator).

The SpiNNaker (Spiking Neural Network Architecture) (Brown et al. 2018) is a massively parallel computing platform designed to simulate large-scale spiking neural networks in real-time. It plans to simulate 1 billion neurons and 1 trillion synapses using 65,536 MPSoC (Multi-Processor System-on-Chip) chips (a total of 1,179,648 processors). The MPSoC chip uses GALS (Globally Asynchronous Locally Synchronous) design scheme, and each chip integrates 18 homogeneous ARM968 integer processor cores, routers, DMA controllers, and other peripheral devices. As its computation is completed by ARM cores, several software packages that help simulate models described in other software are proposed, e.g., SPyNNaker (Rhodes et al. 2018) for PyNN (Davison et al. 2009) and nengo_spinnaker[2] for Nengo (Bekolay et al. 2014).

Besides these neuromorphic chips that use pure digital circuits, quite a few designs also try to make use of the analog circuits to achieve higher power efficiency, better integration, and more parallelism.

BrainScaleS (Brain-inspired multiscale computation in neuromorphic hybrid systems) (Schemmel et al. 2012) is a successor of the FACETS project.[3] Its hardware system is a wafer-scale system that consists of the BWS (BrainScaleS Wafer-Scale System) and a group-based multi-purpose communication system. And the BWS is composed of 384 interconnected digital-analog mixed ASICs named HICANN (High Input Count Analog Neural Network chip). Each HICANN provides 256×2 neural arrays and two 256×224 synaptic arrays. A dedicated software stack named BrainScaleS Operating System (Müller et al. 2020) is provided to support PyNN interface. Thus, BrainScaleS could share the same programming interface with software simulators.

Neurogrid (Benjiamin et al. 2014) is a neuromorphic computing system designed to simulate large-scale neural models. It is a digital/analog hybrid system, and it uses analog circuits for information processing and digital circuits for information transmission. By integrating 16 NeuroCore chips, each of which consists of 256×256 neuron arrays, the Neurogrid system can simulate an SNN that consists of 1 million neurons and 6 billion synapses in real-time. The Neurogrid system comes along with a dedicated software stack which is composed of a user interface (UI), a hardware abstraction layer (HAL), and driver components (Driver). The UI helps the users define the models of the SNNs, the HAL maps the model description to the circuits, and the driver program the mapping onto Neurocores.

Braindrop (Neckar et al. 2019) is another mixed-signal neuromorphic system. And it is designed to be programming at a high level of abstraction. In Braindrop, the core is equipped with 4096 neurons, 64 KB of weight memory, 1024 accumulator buckets, and 1024 synaptic filters.

2 https://github.com/project-rig/nengo_spinnaker.

3 https://facets.kip.uni-heidelberg.de.

Braindrop also provides a backend software whose computation objects are nearly isomorphic to the hardware to support the Nengo (Bekolay et al. 2014) software.

Compared with foreign research, domestic neuromorphic chip research started a bit late, but they are developing rapidly.

Tianjic (Pei et al. 2019) is a hybrid heterogeneous neuromorphic chip that can accommodate both SNNs and DNNs proposed by the Center for Brain-Inspired Computing Research, Tsinghua University. Tianjic chip adopts a many-core architecture, and its basic unit is the multimodal neural computing core. Each core contains 256 neurons and the overall chip contains approximately 4000 neurons and 10 million synapses. As for DNNs, Tianjic can achieve about 1.3TFPLOS. Tianjic also provides a unified model description framework to cover the software stack (Deng et al. 2020).

The Darwin Mouse[4] is another brain-inspired computer built by Zhejiang University. It is equipped with 792 Darwin II neuromorphic co-processors (Ma et al. 2017) and can emulate and support near 120 million neurons and 100 billion synapses.

Moreover, quite a few researchers (Prezioso et al. 2015; Snider et al. 2011; Yu et al. 2011; Jackson et al. 2013; Yao et al. 2020) are trying to make use of emerging non-volatile memory technologies (e.g., memristors) to assist the development of neuromorphic chips. Some of them (Prezioso et al. 2015) even try to simulate the neurons and synapses using memristors directly. However, these researches are still at very early stages and they often do not provide a sophisticated software stack.

All in all, most neuromorphic chip designs usually come along with an end-to-end software toolchain or are hardware-software co-design solutions. Thus, this type of software is tightly coupled with the hardware and it is usually quite difficult to make the software suitable for multiple neuromorphic chips.

4 Software simulation frameworks

Besides neuromorphic chips, there is also a lot of software that tries to simulate the procedure of brain-inspired computing on traditional computers (Carnevale and Hines 2006; Gewaltig and Diesmann 2007; Mureşan and Ignat 2004; Stimberg et al. 2014; Wilson et al. 1988; Yavuz et al. 2016; Qu et al. 2020), working as important simulation tools for computational neuroscience fields.

The NEURON simulation environment (Carnevale and Hines 2006) is a flexible and powerful simulator for models of individual neurons and networks of neurons. It provides tools for conveniently building, managing, and using models. Unlike many other simulators, Neuron uses backward Euler and Crank–Nicholson (Carnevale and Hines 2006) methods other than the forward Euler method to achieve more accuracy and stability. It can simulate the behaviors of the neurons with rich reality and detail, from representing the ion channels to representing the spread of electrical signals in a branched dendritic architecture.

NEST (Gewaltig and Diesmann 2007) is a simulator for spiking neural network models. NEST supports neuron and synaptic models with different levels of biological detail, focusing on the dynamics, size, and structure of the biological system. NEST provides more than 50 neuron models and more than 10 synaptic models, including typical neuron models such as Izhikevich and Hodgkin-Huxley, as well as different variants of the spike-timing-dependent plasticity (STDP) model. NEST is very flexible in defining neuron networks and provides convenient and effective commands, allowing checking and modifying the neuron and connection status at any time during the simulation process. NEST is implemented by C++ and provides a Python interface called PyNEST. NEST can expand from single-core laptops to multi-node supercomputers efficiently. NEST has a large-scale developer community, in which many examples are shared to help users build their own simulation projects.

Brian2 (Stimberg et al. 2014) is a tool for SNN simulation, which is aimed at the application of neuromorphic computation. Brian2 is written in Python and is easy to learn and develop for users. Brian2 is flexible and easy to expand, and convenient for users to use differential equations to define neuron and synapse models and simulation methods. Brian2 uses its own general description method which is easy to generate code for different platforms without modifying the core code, enhancing adaptability and scalability.

GeNN (Yavuz et al. 2016) is a software toolkit for SNN simulation. GeNN transforms different descriptions of neuron simulation into unified C++ code through code generation and generates C++ executable files connecting with NVIDIA GPU. GeNN supports SpineML (Richmond et al. 2014), Python (PyGeNN) (Knight et al. 2021), and Brian2 (Brian2GeNN) (Stimberg et al. 2020) interfaces. Code generation also enables GeNN to support different platforms such as Linux, Mac OS X, and Windows. GeNN mainly optimizes the performance of large-scale neural networks and has greatly improved the simulation speed. Although code generation limits the flexibility of simulation (the network structure cannot be adjusted at runtime), these methods enhance the overall execution performance and efficiency.

BSIM (Qu et al. 2020) is a GPGPU-enabled SNN simulator that focuses on performance. It proposes quite a few

[4] https://www.chinadaily.com.cn/a/202009/07/WS5f55828ba310675eafc57d2d.html.

optimization methods to make full use of the inherent characteristics of SNNs and reaches much better performance compared with other simulators. It also supports the widely used PyNN interfaces (Davison et al. 2009).

Software simulation frameworks play an important role in the field of brain-inspired computing, but they are designed for commercial hardware like CPUs and GPUs, and they usually do not support different types of neuromorphic chips.

5 The frameworks that integrate SNNs and DNNs

With the rapid development of deep learning, there is an increasing research interest that tries to combine the ideas from both neuroscience and computer communities to make further achievements (Hazan et al. 2018; Pei et al. 2019). Thus, quite a few software researches are proposed to integrate SNNs and DNNs.

BindsNET (Hazan et al. 2018) is an SNN framework to implement machine learning and reinforcement learning tasks. BindsNET is developed based on PyTorch framework and supports SNN machine learning applications running on CPU and GPU. Different from the SNN network simulator focusing on biological completeness, BindsNET simplified and abstracted neural operations, using user-friendly and brief grammar coding, enabling users to quickly use and develop bio-inspired algorithms by themselves. BindsNET can solve large-scale machine learning problems and is suitable for rapid prototyping. The disadvantage of BindsNET lies in its inability to implement complex networks and lack of support for delayed synapses.

Norse (Pehle and Pedersen 2021) is an SNN framework focusing on deep learning. Norse is developed based on the sparsity and event-driven features of biological neural networks to solve the von Neumann bottleneck, optimize computing speed, and improve energy efficiency. Norse is implemented based on the PyTorch framework and implements many neurons and synaptic models. Norse also integrates many encoding and decoding algorithms, data set integration, and specific examples to support deep learning applications. Norse can easily expand from a laptop to multiple nodes on an HPC cluster.

SpyTorch[5] proposes a surrogate gradient approach SuperSpike to solve the training problem in SNN's deep learning. Synaptic plasticity is the most important part of the learning process of SNN, but this process is difficult to solve by calculating the gradient which is the traditional deep learning method. SpyTorch proposed a new gradient calculation method called SuperSpike that can transform the spike of the synapses in SNN into a smoother curve, and associate neuromorphic calculations with machine learning, thereby reducing learning costs and increasing simulation efficiency.

SpikingJelly[6] is a framework based on PyTorch that uses Spiking Neural Network for deep learning. SpikingJelly framework supports both clock-driven and event-driven. At the time-driven level, SpikingJelly uses a surrogate gradient approach to replace the gradient of the spiking function. SpikingJelly regards the spiking neuron as an activation function and embeds it into the network built by PyTorch. At the event-driven level, the state of neurons is updated by events. The activities of different neurons can be calculated asynchronously, and there is no need to keep the clock synchronized. SpikingJelly realizes event-driven by implementing Tempotron neurons. SpikingJelly provides several examples of deep learning, including MNIST, encoder, conversion from ANN to SNN, LSTM, etc.

SpykeTorch (Mozafari et al. 2019) is a convolutional spiking neural network simulator based on PyTorch, which supports spike-timing-dependent plasticity (STDP) and reward-modulated STDP learning rules. The SpykeTorch module is inherited from the PyTorch module. It uses tensor calculations to replace the original single neuron calculations, and it is easy to perform neural network simulations on the GPU. SpykeTorch focuses on the simulation and implementation of the STDP and provides multiple examples to help understand the STDP mechanism and simulation process.

These frameworks are usually under active development and not stable enough to adapt to different software and hardware interfaces.

6 Status analysis and development forecast

The basic software for brain-inspired computing is developing rapidly, but there is still quite a bit of shortcoming. Although some have envisioned that the hardware-specific interfaces began to stabilize (Aimone et al. 2019a, b), the existing basic software designs are either tightly coupled with hardware or not "general-purpose" enough to support multiple software and hardware interfaces easily. Thus, its status quo cannot meet the rapid interdisciplinary development needs. We believe that a "general-purpose" hierarchical and HW/SW decoupled basic software framework (Zhang et al. 2021) is needed to overcome the above problems. "General-purpose" here has two meanings:

First, decoupling of software and hardware.

Traditional computers are called general-purpose computers because they are based on the theory of Turing

[5] https://github.com/fzenke/spytorch.

[6] https://github.com/fangwei123456/spikingjelly.

completeness. Turing completeness ensures that any program written in a programming language can be converted into an equivalent instruction sequence on any Turing complete processor (this process is called "compilation"). That is, all computable problems (Turing computability) can be supported, so they are "universal", which also ensures the decoupling characteristics of software and hardware of traditional computers. Inspired by this, some work (Zhang et al. 2020) has proposed "neuromorphic completeness", which theoretically ensures that any Turing computable function can be converted into a model on the neuromorphic complete hardware, which enables the independent development of brain-inspired computing software and hardware (i.e., make feasible the decoupling of software and hardware). In this way, the system hierarchy of brain-inspired computing composed of the software layer, the compiler layer, and the hardware layer can ensure that the applications, basic software, primitive sets, and hardware designs are compatible with each other while developing independently.

Second, a basic software platform that supports the integration of computer science and neuroscience related research.

From the perspective of the evolutionary history of deep learning to a large extent the booming development and application expansion of DNN originated from the second upsurge of neural networks. One of the key technological breakthroughs was the BP algorithm proposed in 1986 for multilayer perceptron.

Since then, with the emergence of high-performance computing platforms represented by GPGPU (including the corresponding programming models and environments, such as CUDA[7]) and the development of machine learning frameworks represented by Tensorflow (Abadi et al. 2016)/ PyTorch (Paszke et al. 2019), deep learning research and development has a uniform and efficient platform, which has greatly attracted many developers (including many non-computer professionals) and laid the foundation for the proposal and development of a large number of innovative algorithms, models and applications.

Now, in the field of neuroscience, a large number of experimental data related to brain circuits/activities have been accumulated,[8] which has stimulated research enthusiasm for the development of large-scale network models in biophysics. Further, research methods that combine biological neural activities with specific intelligent behaviors are one of the critical paths to the next generation of AI.

However, brain-inspired computing lacks a basic platform like the counterpart in the field of deep learning, including a unified platform that can support the integration of computer science and neuroscience related research at the same time.

[7] https://developer.nvidia.com/cuda-downloads.

[8] https://portal.brain-map.org.

Although the micro-architecture of brain-inspired computing chips is quite different from the traditional general-purpose processors, it is of great significance to learn from the successful development experience of traditional computer systems and deep learning, to develop general-purpose brain-inspired computing software in the hierarchical and decoupling technology route.

Acknowledgements Thank Beijing Innovation Center for Future Chips for providing the information about the neuromorphic chips.

Funding Supported by the National Natural Science Foundation of China (62072266). Supported by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (Grant no. 2020A07). Supported by Tsinghua University-China Mobile Communications Group Co., Ltd. Joint Institute (Grant no. 62050340). Supported by Beijing National Research Center for Information Science and Technology (BNR2022RC01003).

Availability of data and material (data transparency) Not applicable.

Code availability (software application or custom code) Not applicable.

Declarations

Conflict of interest No.

References

Abadi, M., Barham, P., Chen, J. et al.: TensorFlow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (2016)

Aimone, J.B., Severa, W., Vineyard, C.M.: Composing neural algorithms with Fugu. Proc. Int. Conf. Neuromorph. Syst. **3**, 1–8 (2019a). https://doi.org/10.1145/3354265.3354268

Aimone, J.B., Severa, W., Vineyard, C.M.: Composing neural algorithms with Fugu. In: Proceedings of the International Conference on Neuromorphic Systems, 1–8 (2019b).

Akopyan, F., Sawada, J., Cassidy, A., et al.: TrueNorth: design and tool flow of a 65 maw 1 million neuron programmable neurosynaptic chip. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(10), 1537–1557 (2015). https://doi.org/10.1109/TCAD.2015.2474396

Amir, A., Datta, P., Risk, W.P., et al.: Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores. In: The 2013 International Joint Conference on Neural Networks (2013). https://doi.org/10.1109/IJCNN.2013.6707078

Bekolay, T., James, B., Eric, H., et al.: Nengo: a Python tool for building large-scale functional brain models. Front. Neuroinform. **7**, 1–48 (2014). https://doi.org/10.3389/fninf.2013.00048

Benjamin, B.V., Gao, P., McQuinn, E., et al.: Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. Proc. IEEE **102**(5), 699–716 (2014). https://doi.org/10.1109/JPROC.2014.2313565

Brette, R., Rudolph, M., Carnevale, T., et al.: Simulation of networks of spiking neurons: a review of tools and strategies. J. Comput. Neurosci. **23**, 349–398 (2007). https://doi.org/10.1007/s10827-007-0038-6

Brown, A.D., Chad, J.E., Kamarudin, R., et al.: SpiNNaker: event-based simulation—quantitative behaviour. IEEE Trans. Multiscale Comput. Syst. **4**(3), 450–462 (2018). https://doi.org/10.1109/TMSCS.2017.2748122

Burkitt, A.N.: A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. Biol. Cybern. **95**, 1–19 (2006). https://doi.org/10.1007/s00422-006-0068-6

Carnevale, N.T., Hines, M.L.: The NEURON Book. Cambridge University Press, Cambridge, Cambridge (2006). https://doi.org/10.1017/CBO9780511541612

Cassidy, A.S., Merolla, P., Arhur, J.V. et al.: Cognitive computing building block: a versatile and efficient digital neuron model for neurosynaptic cores. In: The 2013 International Joint Conference on Neural Networks (2013). https://doi.org/10.1109/IJCNN.2013.6707077

Chen, T., Moreau, T., Jiang, Z., et al.: TVM: an automated end-to-end optimizing compiler for deep learning. In: Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation, pp. 579–594 (2018)

Davies, M., Srinivas, N., Lin, T.-H., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018). https://doi.org/10.1109/MM.2018.112130359

Davison, A., Brüderle, D., Eppler, J., et al.: PyNN: a common interface for neuronal network simulators. Front. Neuroinform. **2**, 1–11 (2009). https://doi.org/10.3389/neuro.11.011.2008

Deng, L., Wang, G., Li, G., et al.: Tianjic: a unified and scalable chip bridging spike-based and continuous neural computation. IEEE J. Solid-State Circuits **55**(8), 2228–2246 (2020). https://doi.org/10.1109/JSSC.2020.2970709

Gewaltig, M.-O., Diesmann, M.: NEST (neural simulation tool). Scholarpedia **2**(4), 1–1430 (2007)

Gidon, A., Zolnik, T.A., Fidzinski, P., et al.: Dendritic action potentials and computation in human layer 2/3 cortical neurons. Science **367**, 83–87 (2020). https://doi.org/10.1126/science.aax6239

Hassabis, D., Kumaran, D., Summerfield, C., et al.: Neuroscience-inspired artificial intelligence. Neuron **95**(2), 245–258 (2017). https://doi.org/10.1016/j.neuron.2017.06.011

Hazan, H., Saunders, D.J., Khan, H., et al.: BindsNET: a machine learning-oriented spiking neural networks library in python. Front. Neuroinform. **12**, 1–89 (2018). https://doi.org/10.3389/fninf.2018.00089

Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. Bull. Math. Biol. **52**, 25–71 (1990). https://doi.org/10.1007/BF02459568

Izhikevich, E.M.: Simple model of spiking neurons. IEEE Trans. Neural Netw. **14**(6), 1569–1572 (2003)

Jackson, B., Rajendran, B., Corrado, G., et al.: Nanoscale electronic synapses using phase change devices. ACM J. Emerg. Technol. Comput. Syst. **9**(2), 1–12 (2013). https://doi.org/10.1145/2463585.2463588

Jordan, J., Ippen, T., Helias, M., et al.: Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. Front. Neuroinform. **12**(2), 1–21 (2018). https://doi.org/10.3389/fninf.2018.00034

Knight, J.C., Komissarov, A., Nowotny, T.: PyGeNN: a Python library for GPU-enhanced neural networks. Front. Neuroinform. **15**, 1–10 (2021). https://doi.org/10.3389/fninf.2021.659005

Lagorce, X., Ryad, B.: Stick: spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony. Neural Comput. **27**(11), 2261–2317 (2015). https://doi.org/10.1162/NECO_a_00783

Li, X., Tang, J., Zhang, Q., et al.: Power-efficient neural network with artificial dendrites. Nat. Nanotechnol. **15**, 776–782 (2020). https://doi.org/10.1038/s41565-020-0722-5

Lin, C.-K., Wild, A., Chinya, G.N., et al.: Programming spiking neural networks on Intel's Loihi. Computer **51**(3), 52–61 (2018). https://doi.org/10.1109/MC.2018.157113521

Ma, D., Shen, J., Gu, Z., et al.: Darwin: a neuromorphic hardware co-processor based on spiking neural networks. J. Syst. Architect. **77**, 43–51 (2017). https://doi.org/10.1016/j.sysarc.2017.01.003

Marblestone, A.H., Wayne, G., Kording, K.P.: Toward an integration of deep learning and neuroscience. Front. Comput. Neurosci. **10**, 1–94 (2016). https://doi.org/10.3389/fncom.2016.00094

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., et al.: SpykeTorch: efficient simulation of convolutional spiking neural networks with at most one spike per neuron. Front. Neurosci. **13**, 1–625 (2019). https://doi.org/10.3389/fnins.2019.00625

Müller, E., Schmitt, S., Mauch, C., et al.: The operating system of the Neuromorphic BrainScaleS-1 system. arXiv (2020)

Mureşan, R.C., Ignat, I.: The "Neocortex" neural simulator a modern design. In: Proceedings of International Conference on Intelligent Engineering Systems (2004)

Neckar, A., Fok, S., Benjamin, B.V., et al.: Braindrop: a mixed-signal neuromorphic architecture with a dynamical systems-based programming model. Proc. IEEE **107**(1), 144–164 (2019). https://doi.org/10.1109/JPROC.2018.2881432

Paszke, A., Gross, S., Massa, F., et al.: PyTorch: an imperative style, high-performance deep learning library. Adv. Neural. Inf. Process. Syst. **32**, 8024–8035 (2019)

Pehle, C., Pedersen, J.E.: Norse—a deep learning library for spiking neural networks. Zenodo (2021). https://doi.org/10.5281/zenodo.4422025

Pei, J., Deng, L., Song, S., et al.: Towards artificial general intelligence with hybrid Tianjic chip architecture. Nature **572**, 106–111 (2019). https://doi.org/10.1038/s41586-019-1424-8

Preissl, R., Wong, T.M., Datta, P. et al.: Compass: a scalable simulator for an architecture for cognitive computing. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (2012). https://doi.org/10.1109/SC.2012.34

Prezioso, M., Merrikh-Bayat, F., Hoskins, B., et al.: Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature **521**, 61–64 (2015). https://doi.org/10.1038/nature14441

Qu, P., Zhang, Y., Fei, X., et al.: High performance simulation of spiking neural network on GPGPUs. IEEE Trans. Parallel Distrib. Syst. **31**(11), 2510–2523 (2020). https://doi.org/10.1109/TPDS.2020.2994123

Rhodes, O., Bogdan, P.A., Brenninkmeijer, C., et al.: sPyNNaker: a Software Package for running PyNN Simulations on SpiNNaker. Front. Neurosci. **12**, 1–816 (2018). https://doi.org/10.3389/fnins.2018.00816

Richards, B.A., Lillicrap, T.P., Beaudoin, P., et al.: A deep learning framework for neuroscience. Nat. Neurosci. **22**, 1761–1770 (2019). https://doi.org/10.1038/s41593-019-0520-2

Richmond, P., Cope, A., Gurney, K., et al.: From model specification to simulation of biologically constrained networks of spiking neurons. Neuroinformatics **12**, 307–323 (2014). https://doi.org/10.1007/s12021-013-9208-z

Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**, 607–617 (2019). https://doi.org/10.1038/s41586-019-1677-2

Schemmel, J., Grübl, A., Hartmann, S. et al.: Live demonstration: a scaled-down version of the BrainScaleS wafer-scale neuromorphic system. In: 2012 IEEE International Symposium on Circuits and Systems (2012). https://doi.org/10.1109/ISCAS.2012.6272131

Snider, G., Amerson, R., Carter, D., et al.: From synapses to circuitry: using memristive memory to explore the electronic brain. Computer **44**(2), 21–28 (2011). https://doi.org/10.1109/MC.2011.48

Stimberg, M., Goodman, D.F.M., Benichoux, V., Brette, R.: Equation-oriented specification of neural models for simulations. Front. Neuroinform. **8**, 1–6 (2014). https://doi.org/10.3389/fninf.2014.00006

Stimberg, M., Goodman, D.F.M., Nowotny, T.: Brian2GeNN: accelerating spiking neural network simulations with graphics hardware. Sci. Rep. **10**, 410 (2020). https://doi.org/10.1038/s41598-019-54957-7

Waldrop, M.: The chips are down for Moore's law. Nature **530**, 144–147 (2016). https://doi.org/10.1038/530144a

Wilson, M., Bhalla, U., Uhley, J., Bower, J: GENESIS: a system for simulating neural networks. In: Advances in neural information processing systems, p. 1 (1988).

Yao, P., Wu, H., Gao, B., et al.: Fully hardware-implemented memristor convolutional neural network. Nature **577**, 641–646 (2020). https://doi.org/10.1038/s41586-020-1942-4

Yavuz, E., Turner, J., Nowotny, T.: GeNN: a code generation framework for accelerated brain simulations. Sci. Rep. **6**, 18854 (2016). https://doi.org/10.1038/srep18854

Yu, S., Wu, Y., Jeyasingh, R., et al.: An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation. IEEE Trans. Electron Devices **58**, 2729–2737 (2011). https://doi.org/10.1109/TED.2011.2147791

Zhang, T., Xu, B.: Research advances and perspectives on spiking neural networks. Chinese J. Comput. **9**, 1767–1785 (2021). https://doi.org/10.11897/SP.J.1016.2021.01767

Zhang, Y., Qu, P., Ji, Y., et al.: A system hierarchy for brain-inspired computing. Nature **586**, 378–384 (2020). https://doi.org/10.1038/s41586-020-2782-y

Zhang, Y., Qu, P., Zheng, W.: Towards "general purpose" brain-inspired computing system. Tsinghua Sci. Technol. **26**(5), 664–673 (2021). https://doi.org/10.26599/TST.2021.9010010

Peng Qu was born in 1991. He received his B.S. and Ph.D. degrees in computer science from Tsinghua University in 2013 and 2018, respectively. He is currently a postdoctoral researcher in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer architecture and neuromorphic computing.

Le Yang was born in 1997. He received his B.S. degree in computer science from Tsinghua University in 2020. He is currently working toward the Master degree in the Department of Computer Science and Technology at Tsinghua University. His research interests include neuromorphic computing and spiking neural networks.

Weimin Zheng was born in 1946. He received the Master degree in computer science from Tsinghua University. Currently, he is an academician of the Chinese Academy of Engineering and a professor at the Department of Computer Science and Technology at Tsinghua University. His research interests include high performance computing, network storage, and parallel compiler.

Youhui Zhang was born in 1974. He received his B.S. and Ph.D. degrees in computer science from Tsinghua University in 1998 and 2002. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer architecture and neuromorphic computing. He is a member of CCF, ACM, and IEEE.