Evolving Connectivity for Recurrent Spiking Neural Networks

Oct. 25 , 2023

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 BACKGROUND

Artificial intelligence (AI) is rapidly transforming our world, but its increasing energy demands pose a major challenge. Current AI systems are trained and deployed on power-hungry hardware, which consumes a significant amount of energy and contributes to climate change.

This chapter provides a background on the energy consumption of AI systems, which is the motivation for my research. I will discuss the increasing energy demands of current AI systems and the posed challenges. I will also show some neuron models from the perspective of computational neuroscience, and then present the potential of spiking neural networks (SNNs) and neuromorphic computing to address these challenges.

1.1 Energy-Consuming AI

In recent years, there has been a rapid increase in the development and use of AI. This growth has been particularly evident in major tech companies like Alphabet and Microsoft, which have significantly increased their investment in AI. This was largely driven by the success of OpenAI's ChatGPT, a conversational AI chatbot that gained millions of users in just a few months. In response, Microsoft and Alphabet have launched their own chatbots, Bing Chat and Bard, respectively[1-3].

While this rapid progress is exciting, it has also raised concerns about the environmental impact of AI and data centers. Data centers currently consume about 1% of the world's electricity, excluding cryptocurrency mining. Between 2010 and 2018, global data center electricity consumption increased by only 6%. However, if AI continues to develop at its current pace, data center energy consumption could grow significantly[1,4]. Figure 1.1 shows the estimated energy from the information and computing technology (ICT) industry by 2030 will account for between 7% to 20% of the total global demand[5]. What's more, data centers consume a lot of water to cool their systems. In fact, data centers are one of the top 10 water-consuming commercial industries in the United States.

AI model training is energy-intensive, with large language models (LLMs) like ChatGPT consuming hundreds of megawatt-hours of electricity during training. This is because LLMs are trained on terabytes of data and have hundreds of billions of parameters[6-7]. According to an analysis, it was estimated that to sustain ChatGPT, OpenAI

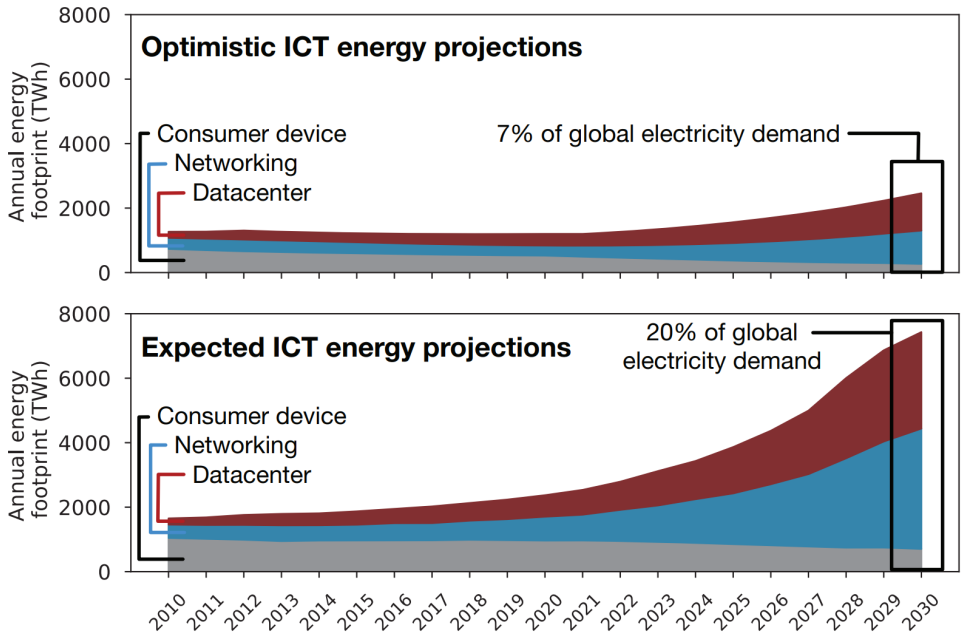

Figure 1.1 Optimistic and expected energy projections of information and computing technology by 2030

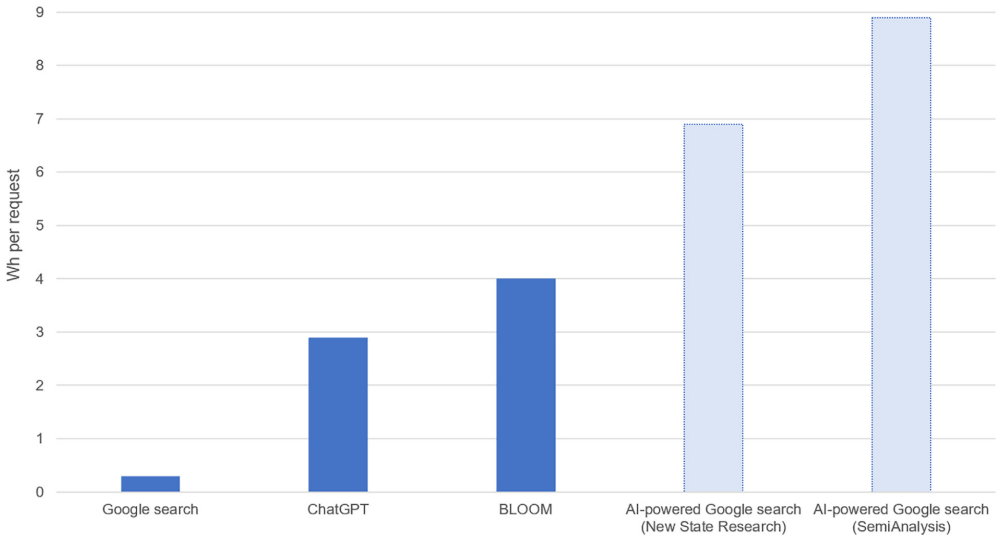

Figure 1.2 Comparison of estimated energy consumption per request of several chatbots and standard Google search

needed 3,617 HGX A100 servers from NVIDIA, accompanied by a total of 28,936 graphics processing units (GPUs). This arrangement indicated a daily energy requirement of 564 MWh[8]. Figure 1.2 presents a comparison between different assessments of the power usage when engaging with a LLM and the power consumption associated with a typical Google search, where the estimations of AI-powered Google search from two institutions are shown[1].

Consequently, the development of low-energy-consuming and environmentally sustainable AI is urgent[5,9]. First, as AI systems become more complex and powerful, they require more energy to train and operate. This is increasing the environmental impact of AI and putting a strain on energy grids and a pressure on both fossil fuels and clean energy[10]. Second, AI is increasingly being used in applications that are critical to our society, such as healthcare, transportation, and manufacturing. If these applications are not energy-efficient, they could lead to widespread disruptions and even blackouts. Third, the development of low-energy-consuming AI is essential for making AI more accessible and affordable. Currently, the high cost of training and operating AI systems limits their use to large companies and governments, or to ordinary customers with a high price or a low-spec or beta version. However, if AI systems can be made more energy-efficient, they will become more accessible and affordable for a wider range of users.

There are a number of ways to develop low-energy-consuming AI[5,9]. One approach is to develop new hardware that is specifically designed for AI applications. Another approach is to develop new algorithms that are more energy-efficient.

1.2 Neuron Models

Compared with current AI systems, the human brain is an extraordinarily energy-efficient system, often referred to as the most energy-efficient computer in the world[11-12]. It performs an astounding number of complex tasks while consuming relatively low amounts of energy. In this section, I'd like to introduce two typical biological neuron models–the Hodgkin-Huxley (HH) Model and the Leaky Integrate-and-Fire (LIF) Model[13-15]–and the relationship between them and the AI neuron models[16-17].

1.2.1 Hodgkin-Huxley Model

HH model for the squid giant axon, developed in 1950s, is the most influential model in realistic neural modeling[13,18]. It provides a detailed and biophysically accurate de-

scription of the mechanisms underlying the action potential.

At its core, the HH model unveils the intricate dance of ions within neurons. It elucidates how the flow of sodium and potassium ions across the neuronal membrane gives rise to the characteristic spikes in electrical activity observed in neurons. This model not only deepened our comprehension of neural dynamics but also opened the door to the study of complex neuronal behaviors and the development of advanced mathematical and computational models.

The HH model allows us to explore the biophysical properties of neurons, offering insights into how ion channels, voltage gradients, and ionic currents orchestrate the neuron's electrical activity. Its contribution to realistic neural modeling cannot be overstated, as it laid the foundation for subsequent advancements in computational neuroscience.

While the HH model is highly detailed and biophysically accurate, it can be computationally intensive. To simplify and reduce computational complexity, researchers have developed some reduced HH models[19-21]. These models retain key features of the original HH model but reduce the number of parameters and equations, making simulations more manageable. Reduced models are valuable for large-scale simulations and studies of neural networks.

1.2.2 Leaky Integrate-and-Fire Model

LIF model is a simplifed, yet highly effective, neuron model, which is is equivalent to a simple RC circuit[14-15]. While LIF models lack the biophysical detail of the HH model, they capture essential aspects of neural behavior. In the LIF model, when the membrane potential reaches a certain threshold, the neuron "fires," generating an action potential and resetting the membrane potential to a resting state.

The LIF model's simplicity and efficiency make it a valuable tool for large-scale simulations and studies of neural networks. However, to capture more complex aspects of neuronal behavior, extensions to the LIF model have been developed. One such extension is the Generalized Integrate-and-Fire (GIF) model, which enhances the LIF model by incorporating additional features such as adaptation and threshold dynamics[22-24]. The GIF model provides a more accurate representation of some neuron behaviors.

1.2.3 Link from HH to LIF to AI Neuron Models

Figure 1.3 shows the progression from the HH model to the LIF model to the AI neuron model. The HH model is the most realistic of the three models, but it is also the most

computationally expensive. The LIF model is a simplified version of the HH model that is more computationally efficient. The AI neuron model is a further simplification of the LIF model that is designed to be as simple and computationally efficient as possible, able to use backpropagation to update its weights. Note that AI neuron models have some important differences. For example, AI neuron models typically use specialized activation functions, such as the rectified linear unit (ReLU) or sigmoid functions. Another difference between AI neuron models and biological neurons is that AI neuron models produce continuous output, while biological neurons produce spikes.

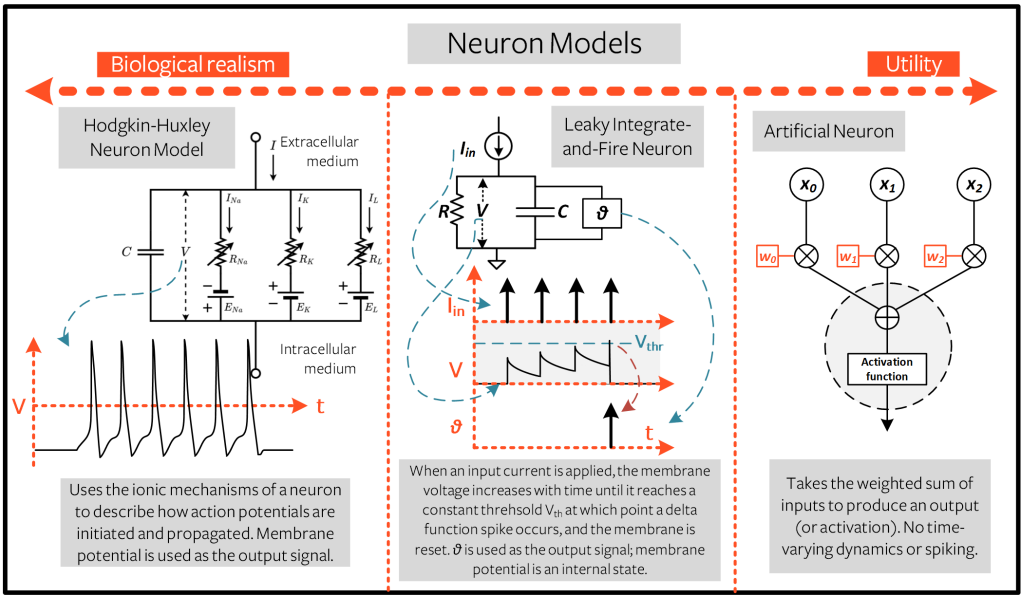

Figure 1.3 Neuron models: biological and artificial

However, these differences come at a cost. AI neuron models, by producing continuous outputs, tend to be more energy-consuming than their biological counterparts. The reasons are multifold:

- **Continuous Activity**: AI neuron models constantly emit activity, while biological neurons spike only when necessary, conserving energy. The sparse and event-driven nature of biological neurons minimizes energy expenditure.
- **Complex Activation Functions**: The use of specialized activation functions in AI neuron models can lead to more intensive computational requirements compared to the relatively straightforward ion channel dynamics of biological neurons.
- **Backpropagation**: The training process in AI neuron models, relying on backpropagation, is computationally demanding and energy-intensive compared to the distributed and experience-based learning mechanisms in biological systems.
- **Hardware Requirements**: The energy efficiency of biological brains stems not

only from neural models but also from the specialized hardware, such as neuromorphic processors, designed to emulate the brain's event-driven nature.

So, while AI neuron models offer computational advantages and are essential for many machine learning applications, they do deviate significantly from the energy-efficient, spike-based operation of biological neurons. Balancing computational efficiency and biological realism is a central challenge in the field of neural networks and neuromorphic architecture design.

1.3 Spiking Neural Networks

Maass[25] explores the emergence of SNN models, representing the third generation of neural network paradigms. These models are inspired by the spiking behavior of biological neurons, emphasizing precise timing, synchronization, and event-driven processing. The paper discusses their biological underpinnings and how they encode information through the timing of spikes. It also covers applications, learning mechanisms, and the challenges associated with these models. Spiking neural networks have significant implications for fields like neuromorphic computing and brain-computer interfaces, offering a more biologically faithful approach to artificial neural networks.

SNNs offer a promising avenue for enhancing the future of artificial intelligence, with several notable advantages that position them as efficient and capable models[26-28]:

- **Energy Efficiency**: SNNs are inherently energy-efficient due to their event-driven processing nature. This aligns well with the need for more sustainable and energy-conscious AI systems, making them suitable for edge devices and IoT applications.
- **Biological Plausibility**: These networks are biologically inspired, capturing the spiking behavior of real neurons. This biological fidelity can enhance our understanding of neural computation and potentially lead to more biologically plausible AI systems.
- **Temporal Processing**: SNNs excel at processing information over time. Their spike-timing-dependent plasticity (STDP) learning rule allows them to learn and respond to temporal patterns, making them valuable for tasks that require temporal understanding, such as speech recognition and motion detection.
- **Sparse Data Representation**: SNNs naturally produce sparse spiking patterns, which can be advantageous for information encoding, reducing redundancy in data, and improving memory efficiency.

- **Neuromorphic Hardware**: Neuromorphic hardware, inspired by SNNs, has the potential to revolutionize computing by providing highly efficient, brain-inspired processing capabilities.

1.4 Neuromorphic Computing

Neuromorphic computing represents an interdisciplinary approach to computing that draws inspiration from the biological brain's operation[29]. From the algorithmic perspective, it leverages neuron models such as the HH model, LIF models, and SNNs. These algorithms closely mimic biological neural processing, offering a level of biological plausibility crucial for understanding neural computation and potentially achieving human-like cognition. Algorithms based on spiking neurons embrace event-driven processing. This means they only respond to input events, or spikes, which allows for highly efficient and real-time decision-making.

It's worth noting that the effectiveness of neuromorphic algorithms is greatly enhanced by specialized hardware platforms[30]. These hardware platforms, such as brain-inspired chips that are computing in or near memory, are tailored to optimize the execution of neuromorphic algorithms[31]. This combination of efficient algorithms and hardware results in exceptional energy efficiency. The hardware's parallel processing capabilities align with the brain's neural parallelism, allowing for efficient execution of algorithms like SNNs, which frequently involve vast numbers of simple neurons. Some neuromorphic hardware designs incorporate on-chip learning capabilities, which is essential for applications requiring continuous adaptation, such as robotics and autonomous systems. The event-driven nature of neuromorphic hardware also facilitates real-time processing, making it an excellent fit for applications like robotics, autonomous vehicles, and sensor networks.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews the research on SNN learning, neuron evolutions, and neuromorphic architecture. SNN learning is the process of training SNNs to perform specific tasks, which is a challenging task due to the complexity of SNNs. Neuron evolutions is the study of how neurons evolve over time, which is an important area for developing more efficient SNNs. Neuromorphic architecture is the design of computer hardware that specifically mimics the structure and function of the brain, which is essential for the development of stronger SNNs.

2.1 SNN Learning

Artificial neural networks (ANNs) use gradient descent, including variants like SGD and ADAM[32-33], along with normalization and distributed training for large-scale models in practical AI. In contrast, SNNs lack universally recognized core learning algorithms, leading to diverse methods due to biological plausibility, task performance, neuron models, and coding schemes[34].

There are three main types of methods to train an SNN.

First, SNNs can be trained by unsupervised learning through bio-inspired mechanisms like Hebbian learning and STDP. Unlike layer-by-layer backpropagation, this method modifies weights using locally available information, making them suitable for distributed computing and online learning. Such SNN algorithms explore the role of synaptic plasticity in building intelligent systems. At the network level, they demonstrate that time-correlated synaptic dynamics enable learning. For example, Guyonneau et al.[35] show that STDP can lead to fast recognition in post-synaptic neurons.

Second, Motivated by the achievements of deep learning, researchers have explored the use of error backpropagation with strategically crafted surrogate gradients to tackle the non-differentiability challenge[36-37]. This approach has shown promise in achieving results comparable to deep learning. Nonetheless, integrating surrogate gradients into RSNNs raises two notable issues. From an algorithmic perspective, surrogate gradients introduce inherent inaccuracies in the descent direction and sensitivity to function scale choices[38-39]. On the implementation side, gradient-based training poses compatibility challenges with leading neuromorphic devices because it necessitates access to the entire

network state at each time step[31].

Third, the transformation of ANNs into SNNs can be used for SNN training. This approach harnesses the efficiency of neuromorphic computing and builds upon established deep learning techniques. The fundamental idea involves approximating ANNs' continuous activation values with SNNs' average firing rates. This is achieved through targeted weight adjustments in pre-trained ANNs, making use of backpropagation for training. The resulting converted SNNs closely align with ANNs in terms of accuracy, making them adaptable for large datasets and evolving network structures. Notably, this approach has been successfully applied to renowned architectures like VGG and ResNet, progressively closing the conversion gap on ImageNet[40-44]. However, constraints from ANNs affect SNN performance. Low-latency SNNs are a challenge due to longer inference times than direct SNN training. Converted SNNs suit static data but not streaming data. Conversion prioritizes performance, limiting exploration of SNN dynamics for brain-inspired intelligence[34].

2.2 Neuron Evolutions

Deep neuroevolution employs evolutionary algorithms to train deep neural networks. In this domain, Natural Evolution Strategies (NES) have been at the forefront of gradient estimation[45]. A widely used version, Evolution Strategies (ES), is applied in training deep neural networks for tasks involving sequential decision-making[46]. ES optimizes a single set of continuous network weights through the introduction of Gaussian perturbations, and it updates these weights using the NES gradient estimator.

Capitalizing on the achievements of ES, several evolutionary algorithms have been introduced for training deep neural networks. For example, Such et al.[47] demonstrated the effective training of deep neural networks using basic genetic algorithms that introduce weight mutations. Additionally, Conti et al.[48] integrated exploration and novelty-seeking into ES to overcome local minima.

Besdies, Hebbian learning[49] and neuron evolution can be related. Hebbian learning can be incorporated into the broader framework of neuron evolution[20,50]. In a neuroevolutionary context, where neural network architectures and synaptic weights are evolved over generations, Hebbian learning can be one of the learning rules used to fine-tune connections between neurons. This is especially relevant when evolving neural networks that are inspired by biological systems. Neuron evolution typically focuses on the architecture

and topology of neural networks, including decisions about the number of neurons, layers, and connectivity patterns. After evolution, Hebbian learning can be applied during the training phase to adapt the network's synaptic weights based on input data. This combination of evolutionary optimization and learning can lead to more efficient networks. In computational neuroscience and biologically inspired neural network models, Hebbian learning is often used to capture the biological plausibility of learning mechanisms. When designing neural networks that aim to mimic the behavior of real neurons and synapses, incorporating Hebbian learning can align the model more closely with biological principles.

2.3 Neuromorphic Architecture

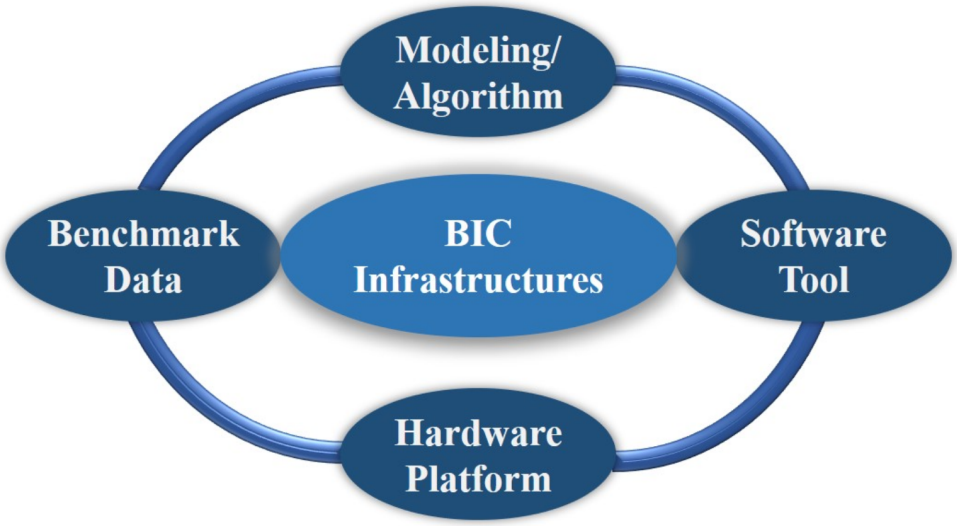

Figure 2.1 Four components of of brain inspired computing (BIC) infrastructures

Figure 2.1 shows brain inspired computing's four important components[34]. From the perspective of theoretical study, the algorithm and hardware should be most focused on in these four components.

In the realm of algorithm, the existing landscape encompasses a spectrum of approaches. This includes the intricate modeling of single neurons at the cellular level, the construction of SNN models at the network level, and the refinement of training mechanisms for these neural systems. A pressing challenge lies in innovating more biologically grounded methodologies capable of addressing tasks that pose inherent difficulties for current AI models. The urgent quest is to explore avenues that draw inspiration from highly bio-plausible neurons, exemplified by multi-compartment neuron models or more

sophisticated neuron representations, or more biologically interpretable and engineering effective learning methods[51-53]. However, this journey must navigate the nuanced intersection: the fidelity to biological principles, practical effectiveness, computational efficiency, and the feasibility of training such systems.

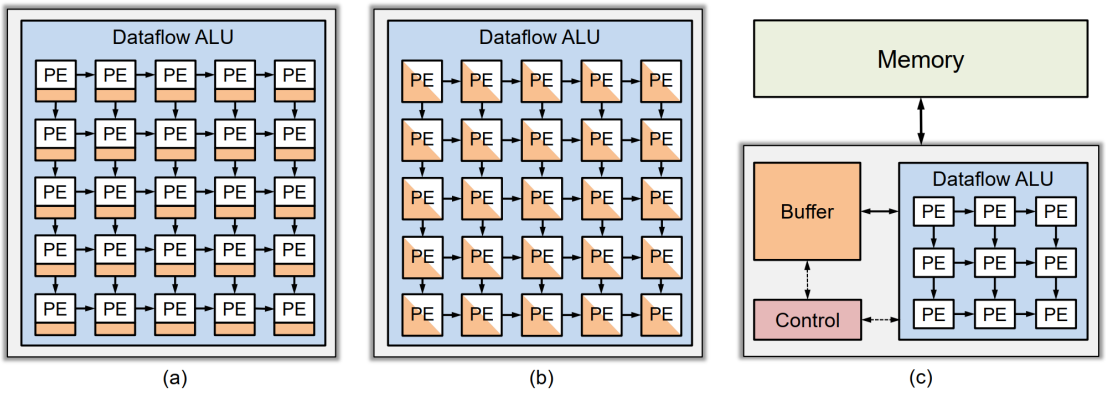

Figure 2.2 Brain inspired computing hardware architecture comparison: (a) distributed many-core design with computation close to memory (near-memory); (b) distributed manycore design with computation within memory (in-memory); (c) version specialized for ANN acceleration.

In terms of hardware platforms, brain inspired computing chips can be aptly designated as neuromorphic chips. In contrast to traditional deep learning accelerators, these chips are designed from the ground up to emulate brain-inspired SNNs. The brain's cortex seamlessly integrates computation and memory, diverging from the separate structures of the von Neumann architecture. Drawing inspiration from this neural arrangement, most BIC chips employ decentralized many-core architectures. In this setup, each core possesses localized computational and memory resources that are tightly interwoven. BIC chips thus exhibit substantial computational parallelism and exhibit a high degree of memory locality, diminishing the necessity for frequent off-chip memory access[54-56]. Moreover, such hardware should match well with the new algorithms that enables SNN learning efficiently. Figure 2.2 shows two types of brain-like computing hardware and the traditional hardware for ANNs[34].

CHAPTER 3 RESEARCH PROPOSAL

In this chapter, I delve into the core of my research proposals, exploring learning, acceleration and application approach to advancing the field of SNNs. I've structured the research into four key aspects, each meticulously designed to address critical challenges. I will begin with "SNN Evolving Connectivity," where we aim to introduce innovative ways of shaping network structures. "Learning Acceleration" takes center stage next, as I strive to enhance the speed and efficiency of SNN learning mechanisms. Transitioning to "Performance Optimization," we will unveil strategies to boost the capabilities and reliability of SNNs. Lastly, under "Application to More Tasks and Networks," I extend the findings to a broader spectrum of tasks and network configurations, in order to showcase the adaptability and potential of SNNs. The comprehensive approach promises to unlock new horizons for SNNs, further bridging the gap between biology and artificial intelligence. This chapter is based on our lab's research[57].

3.1 SNN Evolving Connectivity

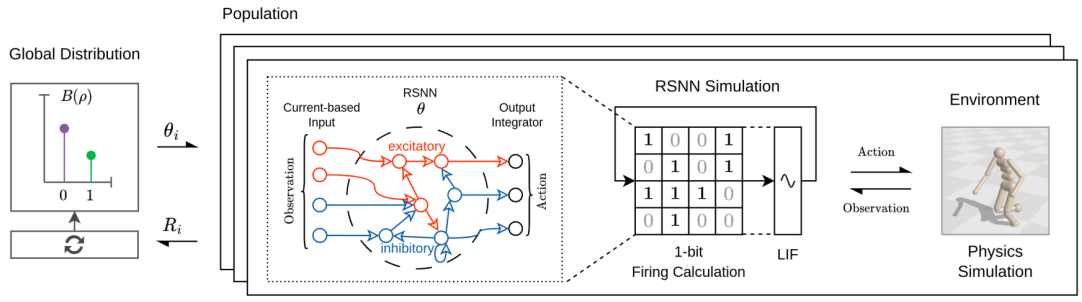

Figure 3.1 Schematic diagram of Architecture of Evolving Connectivity

Recurrent spiking neural networks (RSNNs) have the potential to significantly advance artificial general intelligence by taking cues from the human nervous system and demonstrating promise in capturing complex dynamics. However, the prevalent methods for training RSNNs, which rely on surrogate gradients, suffer from inherent inaccuracies and are not well-suited for neuromorphic hardware. To overcome these challenges, our lab has proposed the evolving connectivity (EC) framework. This framework presents an inference-only approach to train RSNNs. It reimagines the process of adjusting weights as a quest to explore parameterized connection probability distributions and utilizes NES

to optimize these distributions. The EC framework eliminates the need for gradients and exhibits hardware-friendly traits, such as sparse Boolean connections and remarkable scalability. Figure 3.1 shows the architecture of EC.

3.1.1 Network Dynamics

Traditional neural network training primarily focuses on weight assignment to achieve the desired functionality. In contrast, the weight-agnostic neural network concept[58] challenges this notion, emphasizing the significance of a network's structure, akin to conventional weighted networks. Moreover, investigations into the lottery ticket hypothesis have indicated that within an overly parameterized network, an effective subnetwork exists, regardless of the initial weight values[59-61].

Furthermore, theoretical analyses have established that a sufficiently large deep neural network, endowed with random weight values, inherently harbors a subnetwork with the potential to approximate any given function[62-63].

The novel approach builds upon the foundation of connection-encoded networks, introducing a framework that leverages connection probabilities as a means to parameterize the network.

Just like in the human brain, our network comprises two types of neurons: excitatory and inhibitory neurons. Each neuron is represented as a LIF neuron. A neuron generates a spike when its membrane potential (u) surpasses a certain threshold, at which point the membrane potential is reset to 0. In our model, we describe the dynamics of the membrane potential (u) and synaptic current (c) as follows:

$$\tau_m \frac{\mathrm{d}\mathbf{u}^{(g)}}{\mathrm{d}t} = -\mathbf{u}^{(g)} + R\mathbf{c}^{(g)} \tag{3.1}$$

where $g = \{Exc, Inh\}$ is the excitatory and inhibitory group, respectively, and R is the resistance.

The current dynamics is as follows.

$$\frac{\mathrm{d}\mathbf{c}^{(g)}}{\mathrm{d}t} = -\frac{\mathbf{c}^{(g)}}{\tau_{syn}} + \sum_{g_j} I_{g_j} \sum_j \mathbf{W}_{ij}^{(g_i g_j)} \delta(t - t_j^{s(g_j)}) + \mathbf{I}_{ext} \tag{3.2}$$

where $t_j^{s(g_j)}$ denotes the spike time of neuron j in neuron group g_j, δ denotes Dirac delta function. I_g defines the connection strength of excitatory and inhibitory synapses respectively, where $I_{Exc} > 0$ and $I_{Inh} < 0$. Weight $\mathbf{W}^{(g_i g_j)}$ was defined as a matrix with non-negative elements. The external current $\mathbf{I}_{ext}$ is generated linearly by the observation $\mathbf{x}$.

Discretize the LIF differential equations (3.1) and (3.2) with Δt:

$$\mathbf{c}^{(t,g)} = d_c \mathbf{c}^{(t-1,g)} + \sum_{g_j} I_{g_j} \mathbf{W}^{(g_i g_j)} \mathbf{s}^{(t-1,g_j)} + \mathbf{I}_{ext}^{(t,g)} \tag{3.3}$$

$$\mathbf{v}^{(t,g)} = d_v \mathbf{u}^{(t-1,g)} + R\mathbf{c}^{(t,g)} \tag{3.4}$$

$$\mathbf{s}^{(t,g)} = \mathbf{v}^{(t,g)} > \mathbf{1} \tag{3.5}$$

$$\mathbf{u}^{(t,g)} = \mathbf{v}^{(t,g)}(\mathbf{1} - \mathbf{s}^{(t,g)}) \tag{3.6}$$

where $d_c = e^{-\frac{\Delta t}{\tau_{syn}}}$ and $d_v = e^{-\frac{\Delta t}{\tau_m}}$ are two constant parameters.

The output vector $\mathbf{o}$ is extracted using linear projection of neuron firing rates in a short time period τ, i.e.,

$$\mathbf{o}^{(t)} = \sum_{\tau} k(\tau) \sum_{g} \mathbf{W}_{out}^{(g)} \mathbf{s}^{(t-\tau,g)} \tag{3.7}$$

where $\mathbf{W}_{out}^{(g)}$ denotes the output weight, and k is a function averaging the time period τ.

3.1.2 Connectivity Evolution

Based on the above network dynamics, then we can define the popultaion and optimization for EC.

Population: generated by sampling from a Bernoulli distribution.

$$\mathbf{W}_{ij} = \mathbf{w}_{ij} \cdot \boldsymbol{\theta}_{ij}, \text{where } \boldsymbol{\theta}_{ij} \sim B(\rho) \tag{3.8}$$

where ρ is the connection probability matrix. If setting the weights to unit size, then we have:

$$\mathbf{W}_{ij} = \boldsymbol{\theta}_{ij}, \text{where } \boldsymbol{\theta}_{ij} \sim B(\boldsymbol{\rho}_{ij}). \tag{3.9}$$

Optimization: maximizing the expected performance metric function $R(\cdot)$ across individual network samples. $R(\cdot)$ is from the environment.

$$\boldsymbol{\rho}^* = \text{argmax}_{\boldsymbol{\rho}} J(\boldsymbol{\rho}) = \text{argmax}_{\boldsymbol{\rho}} \mathrm{E}_{\boldsymbol{\theta} \sim B(\boldsymbol{\rho})}[R(\boldsymbol{\theta})] \tag{3.10}$$

Estimate the gradient of $J(\boldsymbol{\rho})$ to update the connectivity:

$$\nabla_{\boldsymbol{\rho}} J(\boldsymbol{\rho}) = \mathrm{E}_{\boldsymbol{\theta} \sim B(\boldsymbol{\rho})}[\nabla_{\boldsymbol{\rho}} \log P(\boldsymbol{\theta}|\boldsymbol{\rho}) R(\boldsymbol{\theta})] \tag{3.11}$$

$$= \mathrm{E}_{\boldsymbol{\theta} \sim B(\boldsymbol{\rho})}[\frac{\boldsymbol{\theta} - \boldsymbol{\rho}}{\boldsymbol{\rho}(1 - \boldsymbol{\rho})} R(\boldsymbol{\theta})] \tag{3.12}$$

$$\approx \frac{1}{N} \sum_{k=1}^{N} \frac{\boldsymbol{\theta}_k - \boldsymbol{\rho}}{\boldsymbol{\rho}(1 - \boldsymbol{\rho})} R_k \tag{3.13}$$

The step of learning can be tried with different expressions in the simulation to find the best one.

3.2 Learning Acceleration

3.2.1 Binary Operation Acceleration

Our population are in binary pattern, which correspond to the connectivity between neurons. To accelerate this operation while training the RSNN model, an effective method is to group binary matrices and vectors into a smaller-dimension matrix or vector that is in int32 format. That is, we can group long big binary numbers into smaller bitsets. The multiplication between a binary matrix and a binary vector is an "and" operation plus a "sum" operation, which is equivalent to a "bitcount" operation for an int32 number. A "bitcount" operation is very fast, which counts the number of 1 for an int32 number in binary memory.

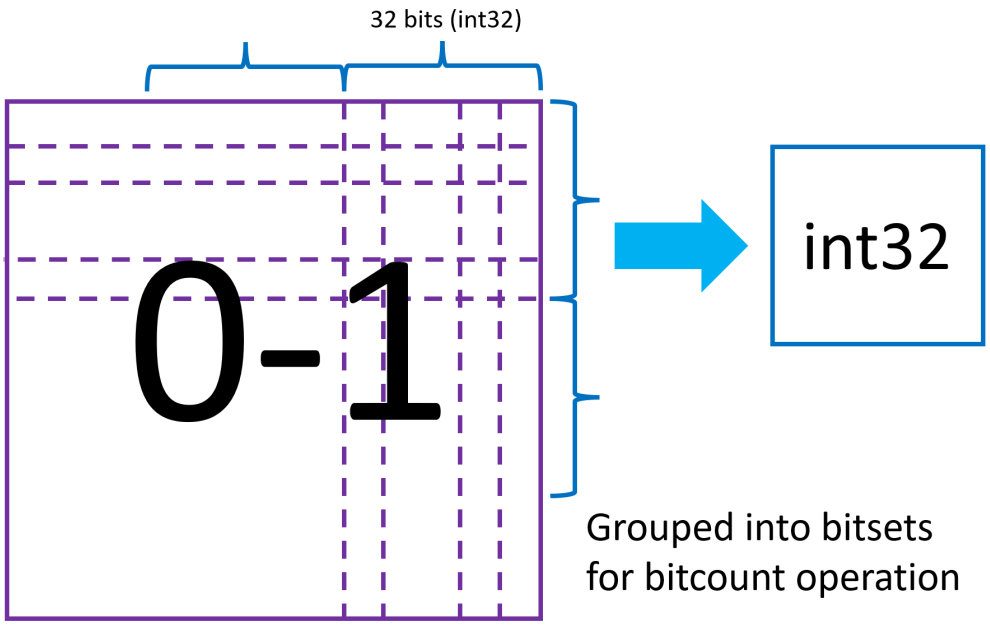

Figure 3.2 Schematic diagram of transforming a binary matrix into a int32 matrix

Figure 3.2 shows the process of grouping a binary matrix into a int32 matrix, which are bitsets of the binary format. The transform of vectors are similar. For example, a 0-1 matrix A of size 70×70 can be converted into an int32 matrix B of size 3×3, and a 0-1 vector of length 70 can be transformed into an int32 vector of length 3.

3.2.2 Excitatory and Inhibitory Neuron Weights Acceleration

For weights of excitatoy neurons $\mathbf{W}^+$ and inhibitory neurons $\mathbf{W}^-$ ($\mathbf{W}^- = \text{logicNot}(\mathbf{W}^-)$), to avoid the minus operation of them, which may produce -1 in the matrix and disable the "bitcount" operation, we can calculate them as follows:

$$(\mathbf{W}^+ – \mathbf{W}^-)x = (2\mathbf{W}^+ – \mathbf{1})x = 2\mathbf{W}^+ x – \mathbf{1}x \tag{3.14}$$

Then $\mathbf{W}^+ x$ is suitable for the "bitcount" operation.

3.3 Performance Optimization

The key attributes of the EC framework in terms of implementation, primarily due to its utilization of connection probability distributions.

First, a major challenge with neuromorphic devices is the lack of an efficient hardware-friendly learning algorithm. Many neuromorphic chips are incapable of directly performing error backpropagation, a critical component of surrogate gradient methods. The EC framework, designed for inference-only operations, provides an alternative approach for training on these inference chips.

Second, the inference-only nature of the EC framework ensures that evaluations are not dependent on prior data, allowing sampled parameters to be distributed across independent workers. Their reported performance measures can then be collected, making the EC framework highly scalable. Furthermore, the random seed for sampling the population can be transmitted between nodes rather than the connections, reducing communication overhead and enabling more efficient scalar-only communication.

Third, the EC framework employs 1-bit sparse connections throughout both training and deployment, replacing the conventional floating-point weight matrix. These 1-bit connections facilitate the use of more cost-effective integer arithmetic instead of resource-intensive floating-point operations. This approach not only accelerates computations on devices such as GPUs but also holds the potential to drive the development of innovative 1-bit connection neuromorphic computing hardware.

Our lab has tested the framework on three robotic locomotion tasks–Humanoid (17-DoF), Walker2d (6-DoF), and Hopper (3-DoF)[64-65], which are ordinary reinforcement learning tasks Figure 3.3.

Our baselines are deep RNNs, RSNN trained with Surrogate Gradients (SG) and ES,

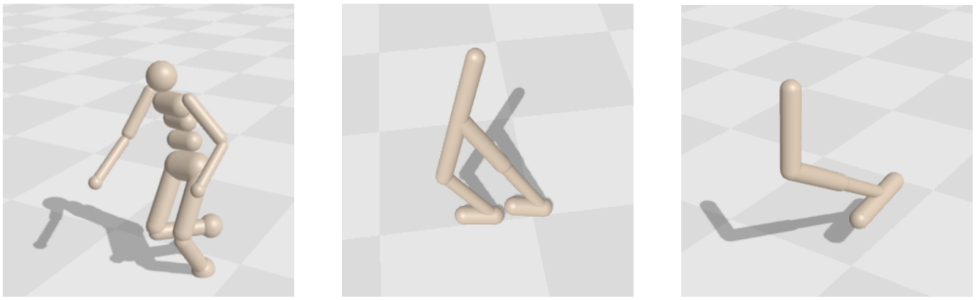

Figure 3.3 Locomotion tasks–Humanoid (17-DoF), Walker2d (6-DoF), and Hopper (3-DoF)

and ES trained GRU and LSTM.

The comparison results are shown in Figure 3.4 and Figure 3.5.

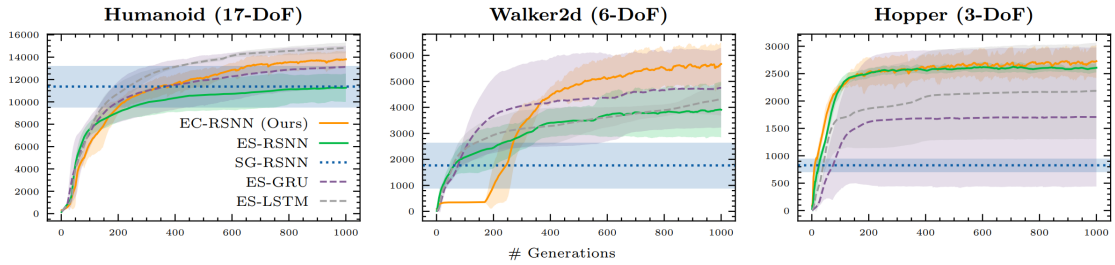

Figure 3.4 Comparison of performance of different models

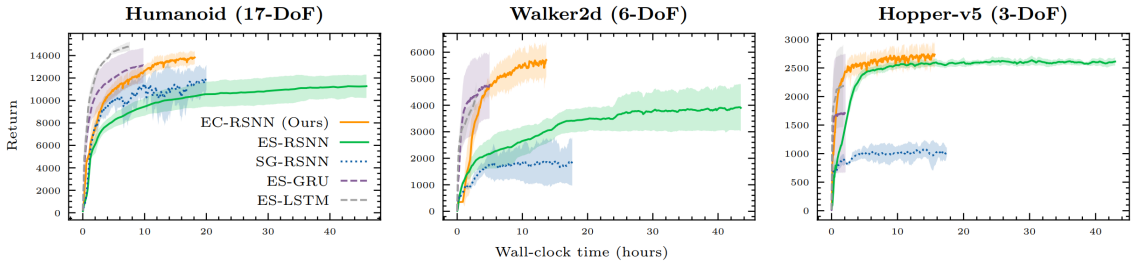

Figure 3.5 Comparison of speed of different models

Basically, for performance, we can see that EC-RSNN outperforms ES-RSNN and is comparable with the ANN models; for speed, EC-RSNN outperforms ES-RSNN but is slower than LSTM and GRU because RSNN is highly complex and GPUS have been optimized for LSTM and GRU. Also, in these results, int32 acceleration is not used yet.

3.4 Discussion and Application to More Tasks and Networks

Developing efficient on-chip learning algorithms for neuromorphic hardware has been a significant challenge compared to traditional deep learning methods. Our EC framework offers a unique solution by eliminating the need for gradients and has proven effective in

locomotion tasks, addressing the on-chip learning challenge. The framework is versatile, supporting large-scale learning in cloud environments and energy-efficient applications at the edge, making it a promising tool for advancing neuromorphic applications. Furthermore, it introduces a novel design principle of transitioning from floating-point to 1-bit connections, potentially reducing manufacturing and energy costs.

The EC framework introduces a novel approach to neuroscience research. It utilizes real-world-like tasks with recurrent spiking neural networks, setting the stage for in-depth exploration of decision-making processes and motion control. This approach also generates valuable neuron-to-neuron connection probability data, offering insights into brain-wide connectomes that are otherwise experimentally challenging to obtain. Moreover, our framework can incorporate neuroanatomical and neurophysiological data, facilitating the construction of data-driven neurosimulation models based on factors like spatial distance, receptive field, and neuron types.

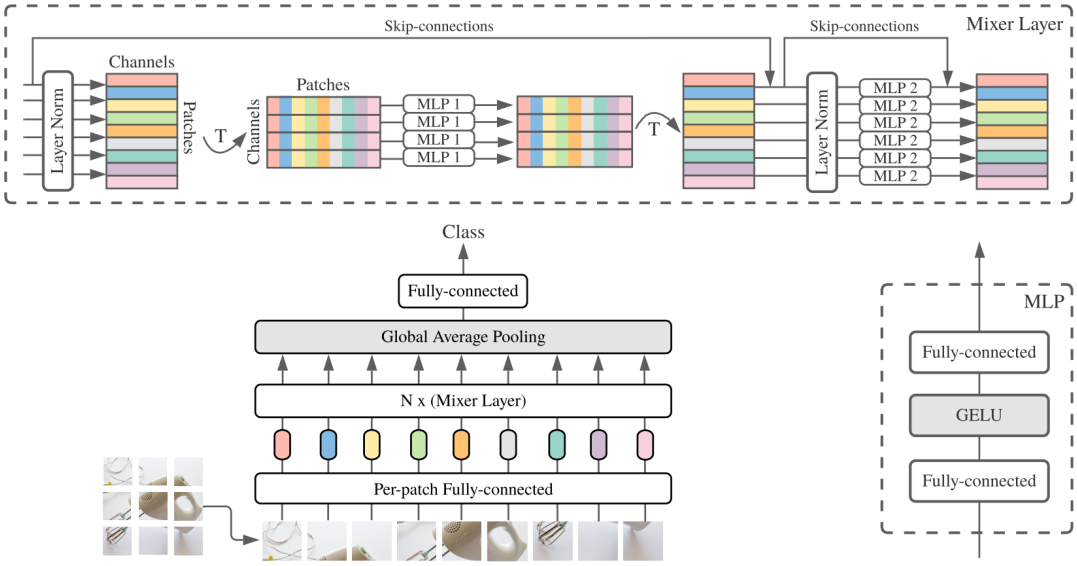

Figure 3.6 Schematic diagram of MLP-mixer

The next plan is to optimize the EC framework and extend it to more tasks and network configurations. A new task can be image recognition: we can use the MLP-mixer–which is a vision transformer achieved by MLPs so that it is easier for implementation in RSNNs[66] (see Figure 3.6). A new network configuration can be RetNet–which is a successor to transformer for LLMs with lower complexity compared to traditional transformers[67].

More possibilities are being explored!

REFERENCES

[1] de Vries A. The growing energy footprint of artificial intelligence[J]. Joule, 2023.

[2] AI G. Bard[EB/OL]. 2023. https://bard.google.com/chat.

[3] OpenAI. Chatgpt[EB/OL]. 2023. https://chat.openai.com/chat.

[4] Patterson D, Gonzalez J, Hölzle U, et al. The carbon footprint of machine learning training will plateau, then shrink[J]. Computer, 2022, 55(7): 18-28.

[5] Gupta U, Kim Y G, Lee S, et al. Chasing carbon: The elusive environmental footprint of computing[C]//2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021: 854-867.

[6] Verdecchia R, Sallou J, Cruz L. A systematic review of green ai[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2023: e1507.

[7] Luccioni A S, Viguier S, Ligozat A L. Estimating the carbon footprint of bloom, a 176b parameter language model[A]. 2022.

[8] SemiAnalysis. The inference cost of search disruption - large language model cost analysis [EB/OL]. 2023. https://www.semianalysis.com/p/the-inference-cost-of-search-disruption.

[9] Wu C J, Raghavendra R, Gupta U, et al. Sustainable ai: Environmental implications, challenges and opportunities[J]. Proceedings of Machine Learning and Systems, 2022, 4: 795-813.

[10] Elkin C, Witherspoon S. Machine learning can boost the value of wind energy[EB/OL]. 2019. https://deepmind.com/blog/article/machine-learning-can-boost-value-wind-energy.

[11] Attwell D, Laughlin S B. An energy budget for signaling in the grey matter of the brain[J]. Journal of Cerebral Blood Flow & Metabolism, 2001, 21(10): 1133-1145.

[12] Lennie P. The cost of cortical computation[J]. Current biology, 2003, 13(6): 493-497.

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

[14] Rotter S, Diesmann M. Exact digital simulation of time-invariant linear systems with applications to neuronal modeling[J]. Biological cybernetics, 1999, 81(5-6): 381-402.

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

[16] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

[17] Hassabis D, Kumaran D, Summerfield C, et al. Neuroscience-inspired artificial intelligence[J]. Neuron, 2017, 95(2): 245-258.

[18] Hodgkin A L, Huxley A F, Katz B. Measurement of current-voltage relations in the membrane of the giant axon of loligo[J]. The Journal of physiology, 1952, 116(4): 424.

[19] Izhikevich E M. Dynamical systems in neuroscience[M]. MIT press, 2007.

[20] Gerstner W, Naud R. How good are neuron models?[J]. Science, 2009, 326(5951): 379-380.

[21] Gerstner W, Kistler W M, Naud R, et al. Neuronal dynamics: From single neurons to networks and models of cognition[M]. Cambridge University Press, 2014.

[22] Fourcaud-Trocmé N, Hansel D, Van Vreeswijk C, et al. How spike generation mechanisms determine the neuronal response to fluctuating inputs[J]. Journal of neuroscience, 2003, 23 (37): 11628-11640.

[23] Richardson M J, Gerstner W. Synaptic shot noise and conductance fluctuations affect the membrane voltage with equal significance[J]. Neural computation, 2005, 17(4): 923-947.

[24] Brette R, Gerstner W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity[J]. Journal of neurophysiology, 2005, 94(5): 3637-3642.

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

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

[27] Maass W, Natschläger T, Markram H. Real-time computing without stable states: A new framework for neural computation based on perturbations[J]. Neural computation, 2002, 14(11): 2531-2560.

[28] Furber S. Large-scale neuromorphic computing systems[J]. Journal of neural engineering, 2016, 13(5): 051001.

[29] Tang J, Yuan F, Shen X, et al. Bridging biological and artificial neural networks with emerging neuromorphic devices: fundamentals, progress, and challenges[J]. Advanced Materials, 2019, 31(49): 1902761.

[30] Indiveri G, Liu S C. Memory and information processing in neuromorphic systems[J]. Proceedings of the IEEE, 2015, 103(8): 1379-1397.

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

[32] Bottou L. Stochastic gradient descent tricks[M]//Neural networks: Tricks of the trade. Springer, 2012: 421-436.

[33] Kingma D P, Ba J. Adam: A method for stochastic optimization[A]. 2014.

[34] Li G, Deng L, Tang H, et al. Brain inspired computing: A systematic survey and future trends [M]. TechRxiv, 2023.

[35] Guyonneau R, VanRullen R, Thorpe S J. Neurons tune to the earliest spikes through stdp[J]. Neural Computation, 2005, 17(4): 859-879.

[36] Wu Y, Deng L, Li G, et al. Spatio-temporal backpropagation for training high-performance spiking neural networks[J]. Frontiers in neuroscience, 2018, 12: 331.

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

[38] Zenke F, Vogels T P. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks[J]. Neural computation, 2021, 33(4): 899-925.

[39] Li Y, Guo Y, Zhang S, et al. Differentiable spike: Rethinking gradient-descent for training spiking neural networks[J]. Advances in Neural Information Processing Systems, 2021, 34: 23426-23439.

[40] Stockl C, Maass W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes[J]. Nature Machine Intelligence, 2021, 3(3): 230-238.

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

[42] Sengupta A, Ye Y, Wang R, et al. Going deeper in spiking neural networks: Vgg and residual architectures[J]. Frontiers in Neuroscience, 2019, 13.

[43] Hu Y, Tang H, Pan G. Spiking deep residual network[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[44] Han B, Roy K. Deep spiking neural network: Energy efficiency through time based coding[C]// European Conference on Computer Vision. 2020: 388-404.

[45] Wierstra D, Schaul T, Glasmachers T, et al. Natural evolution strategies[J]. The Journal of Machine Learning Research, 2014, 15(1): 949-980.

[46] Salimans T, Ho J, Chen X, et al. Evolution strategies as a scalable alternative to reinforcement learning[A]. 2017.

[47] Such F P, Madhavan V, Conti E, et al. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning[A]. 2017.

[48] Conti E, Madhavan V, Petroski Such F, et al. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents[J]. Advances in neural information processing systems, 2018, 31.

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

[50] Rizzuto D, Madsen J, Bromfield E, et al. Reset of human neocortical oscillations during a working memory task[J]. Proceedings of the national academy of Sciences, 2003, 100(13): 7931-7936.

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

[52] Sacramento J, Ponte Costa R, Bengio Y, et al. Dendritic cortical microcircuits approximate the backpropagation algorithm[C]//Advances in Neural Information Processing Systems: volume 31. 2018.

[53] Guerguiev J, Lillicrap T P, Richards B A. Towards deep learning with segregated dendrites[J]. ELife, 2017, 6: e22901.

[54] Painkras E, Plana L A, Garside J, et al. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation[J]. IEEE Journal of Solid-State Circuits, 2013, 48(8): 1943-1953.

[55] 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.

[56] Pehle C, Billaudelle S, Cramer B, et al. The brainscales-2 accelerated neuromorphic system with hybrid plasticity[J]. Frontiers in Neuroscience, 2022, 16.

[57] Wang G, Sun Y, Cheng S, et al. Evolving connectivity for recurrent spiking neural networks [A]. 2023.

[58] Weight agnostic neural networks[C]//Advances in Neural Information Processing Systems: volume 32. 2019.

[59] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks [A]. 2018.

[60] Zhou H, Lan J, Liu R, et al. Deconstructing lottery tickets: Zeros, signs, and the supermask[J]. CoRR, abs/1905.01067, 2019.

[61] Ramanujan V, Wortsman M, Kembhavi A, et al. What's hidden in a randomly weighted neural network?[C]//2020.

[62] Fischer J, Gadhikar A, Burkholz R. Lottery tickets with nonzero biases[Z]. 2022.

[63] Malach E, Yehudai G, Shalev-Schwartz S, et al. Proving the lottery ticket hypothesis: Pruning is all you need[C]//Proceedings of the 37th International Conference on Machine Learning: volume 119. 2020: 6682-6691.

[64] Brockman G, Cheung V, Pettersson L, et al. Openai gym[A]. 2016.

[65] Freeman C D, Frey E, Raichuk A, et al. Brax–a differentiable physics engine for large scale rigid body simulation[A]. 2021.

[66] Tolstikhin I O, Houlsby N, Kolesnikov A, et al. Mlp-mixer: An all-mlp architecture for vision [J]. Advances in neural information processing systems, 2021, 34: 24261-24272.

[67] Sun Y, Dong L, Huang S, et al. Retentive network: A successor to transformer for large language models[A]. 2023.