CCF Transactions on High Performance Computing (2020) 2:254–271
https://doi.org/10.1007/s42514-020-00037-6

REGULAR PAPER

FPT-spike: a flexible precise-time-dependent single-spike neuromorphic computing architecture

Tao Liu[1] · Gang Quan[2] · Wujie Wen[1]

Received: 11 March 2020 / Accepted: 14 May 2020 / Published online: 16 June 2020

Abstract
Modern Artificial Neural networks (ANNs) like Convolutional Neural Network (CNN), have found broad applications in real-world cognitive tasks. One challenging faced by these models is their tremendous memory and computing resource requirement. This also greatly hinders their adoptions from resource-constraint platforms, such as drone, mobile phone and IoT devices. Recently the brain-inspired spiking neural network (SNN) has been demonstrated as a promising solution for delivering more impressive computing and power efficiency. For SNNs, a large body of prior work were conducted on the spiking system design with a focus on using the spike firing rate (or rate-coded) for fulfilling the practical cognitive tasks. Such rate-based designs can underestimate the energy efficiency, throughput and system flexibility of SNNs. On the other hand, the potentials of time-based SNN are not fully unleashed in real applications due to lack of efficient coding and practical learning schemes in temporal domain. In this work, we make an early attempt to fill this gap: that said, a flexible precise-time-dependent single-spike neuromorphic computing architecture, namely "FPT-Spike", is developed. "FPT-spike" relies on three hardware-favorable components: precise ultra-sparse spike temporal encoding, efficient supervised temporal learning and fast asymmetric decoding, to realize flexible spatial–temporal information trade-off for neural network size reduction without scarifying data processing capability. Extensive experimental results show that "FPT-Spike" outperforms rate-based SNN and ANN significantly in three aspects: network size, processing speed and power consumption, demonstrating great potentials for its deployment in edge devices.

Keywords Neuromorphic computing · Spiking neural network · Time coding

1 Introduction

As one of the most fascinating developments of Artificial Intelligence (AI), deep learning enabled neural network system, i.e. deep neural network (DNN) or convolutional neural network (CNN), has found broad applications in realistic cognitive tasks such as speech recognition, image processing, machine translation and object detection etc. (LeCun et al. 2015; Szegedy 2016). However, performing high-accurate testings for complex DNNs or CNNs requires massive amounts of computation and memory resources, indicating limited energy efficiency. For instance, the recognition implementation of state-of-the-art CNN–AlexNet (Krizhevsky et al. 2012) involves not only huge volumes of parameters (61 million) generating intensive off-chip memory accesses but also a large number of computing-intensive high precision floating-point operations (1.5 billion) (Farmahini-Farahani et al. 2015). The existing DNN or CNN accelerators implemented in general-purpose platforms, e.g. CPU and GPU, or domain-specific hardware like FPGA, usually focus on performance enhancement while hardware cost and energy consumption are not the primary design considerations (Ciresan et al. 2011; Vanhoucke et al. 2011; Farabet et al. 2011). Such a weakness hinders these solutions from

This work is supported in part by NSF under project CNS-1423137.

✉ Tao Liu
tal519@lehigh.edu

Gang Quan
gang.quan@fiu.edu

Wujie Wen
wuw219@lehigh.edu

[1] Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, USA

[2] Department of Electrical and Computer Engineering, Florida International University, Miami, USA

many emerging applications of mobile autonomous systems like smart device, Internet-of-Things (IoT), wearable device, robotics etc., where very tighten power budget, hardware resource and footprint are enforced (Andri et al. 2016; Han et al. 2016).

Compared to the CNN and DNN designs, spiking-based neuromorphic computing, which is inspired by the working mechanism and efficiency of human brain, has received increased attentions from both academia and industry for achieving tremendous computing efficiency at much lower power of a single chip, e.g. the famous IBM TrueNorth chip that has total 1 million synapses and an operating power of 70 mW (Akopyan et al. 2015). These low-power, light, and small single-chip solutions leverage the efficient event-driven concept to ease the computational load and enable possible cognitive applications in battery-limited smart device, mobile phone, and IoT etc., creating a very unique but promising branch of neuromorphic computing research (Neil and Liu 2014; Corradi and Indiveri 2015; Liu et al. 2017).

The spiking neuromorphic architecture originates from the bio-inspired spiking neural network (SNN) and relies on the electrical spikes to represent the data and perform the computation. The information of SNN is usually conveyed by the occurrence frequency of spikes (rate coding) or their firing time (time coding). Rate coding, which can be efficiently obtained by simply generating a large number of input spikes (i.e. the spike occurrence frequency proportional to the intensity of the input such as the pixel density of the image (Liu et al. 2016; Akopyan et al. 2015), has been widely adopted in many realistic cognition tasks. However, the rate-based SNN has significant limitations in both energy efficiency and processing speed: First, a sufficiently long time window should be always maintained not only for firing enough spikes by the input neurons (i.e. 20 Hz firing rate Akopyan et al. 2015), but also for performing the training and inference tasks by the output neurons (e.g. winner-takes-all rule based on the numbers of fired spikes Maass 2000), resulting in a computation speed degradation; Second, the energy efficiency of spiking-based designs is highly coupled with the number of processed spikes. For example, the energy-per-spike is a key measurement index (i.e. 45 pJ-per-spike for IBM TrueNorth Chip Akopyan et al. 2015). The more spikes are fired, the more energy will be consumed by the spike related activities, i.e. spike firing of the input neurons, synaptic weighting and Integrate-and-Fire (IFC) processing of the output neurons.

Compared to the rate-based SNN, the more biological plausible time-based SNN may offer better system throughput and energy efficiency (Thorpe et al. 2001), since the information can be embedded in the time (temporal) domain of short and sparse spikes instead of the large numbers of spikes. However, the lack of efficient hardware-favorable solutions for time-based information representation and complex spike-timing-dependent (temporal) training of biological synapses greatly limit practical applications of such an emerging architecture (Wang et al. 2015).

On one hand, translating the input stimulus (i.e. image pixels) to the delay of the spikes, namely time-based encoding, is non-trivial because the coding efficiency can be easily degraded by the biased spike delays distributed in the limited coding intervals. As we shall show later, the similarity of different patterns is likely to be mapped to a same spiking delay if not properly encoded, leading to a significant accuracy degradation. Also, the hardware realization of time coding is usually expensive, as the time-based spike kernel needs to be carefully designed to provide accurate time information (e.g. pre-synaptic/post-synaptic time Thorpe et al. 2001) for time-based training.

On the other hand, realizing more biological plausible spiking-time based training, i.e. unsupervised spiking-time-dependent plasticity (STDP), is very complex and costly due to the exponential time dependence of weight change and difficult convergence of learning (Sjöström and Gerstner 2010). In real-world applications, training of the rate-based SNN can be usually performed off-line by directly borrowing the standard back-propagation algorithm from artificial neural network (ANN) (Liu et al. 2016). However, this time-independent learning rule does not fit the time-dependent SNN because of a fundamentally different training mechanism.

In this work, we investigate the possibility of unleashing the potentials of time-based SNN architecture in realistic applications by orchestrating the efficient time-based coding/decoding and learning algorithm. A Flexible Precise-Time-Dependent Single-Spike Neuromorphic Computing Architecture, namely "FPT-Spike", is proposed to facilitate the cognitive tasks like the MNIST digit recognition. Our "FPT-Spike" incorporates three integrated techniques: precise single-spike temporal encoding, efficient supervised temporal learning, and fast asymmetric decoding. Our major contributions are:

1. We developed a precise-temporal encoding approach to efficiently translate the information into the temporal domain of a single spike. The single spike solution dramatically reduces the energy, while offering a flexible spatial–temporal information conversion to satisfy various design trade-offs among model size, speed and accuracy;
2. We proposed a supervised temporal learning algorithm to facilitate synaptic plasticity on this single-spike system. The proposed algorithm significantly improves the single-layer network learning capability and achieves comparable accuracy when compared to the ANN and rate-based SNN under the similar configuration;

3. We developed a novel asymmetric decoding to relieve the unique and serious weight competition issue existing in this single-spike system, and significantly improved the efficacy and efficiency of synaptic weight updating.
4. We developed the time-based error back-propagation learning rule to improve the learnability and discussed a possible pipeline based neural processing scheme to implement the deep learning on the proposed single-spike system and support different embedded devices.

To the best of our knowledge, this is the first comprehensive study on the end-to-end time-based spiking neuromorphic design consists of efficient, holistic techniques such as time-based encoding, learning, and decoding, to explore how the time-based single-spike SNN architecture can be designed to perform the practical recognition tasks. Our simulation and implementation on mobile devices show that our proposed single-spike SNN architecture can achieve similar classification accuracy as traditional rate-based SNNs and ANNs. Moreover, the impressive improvements in power, network size and learning efficiency, and the flexibility in spatial–temporal information conversion for trade-offs, well demonstrate the unique advantages of proposed time-based spiking neuromorphic architecture over existing solutions. The performance of proof-of-concept "*FPT-Spike*" can be further enhanced in various ways, i.e., increasing more layers through the proposed time-based error back-propagation learning rule. We hope that our results enable the community to examine the time-based SNN architecture for real-time cognitive applications in the growing power-constraint platforms, i.e., mobile, IoT devices, etc.

2 Background and motivation

2.1 Spiking based neuromorphic systems

As the next-generation neural network, spiking neural network (SNN) represents its information as the spike train or voltage pulses vector. This biological inspiration indicates a better computation efficiency with a less expensive hardware, and hence triggers its wide adoption in Neuromorphic Computing System (NCS) designs. Neural coding and synaptic plasticity are two crucial components in SNN designs. The encoded spike train will be sent through associated synapses and further tuned by corresponding synaptic conductances to strengthen or weaken the response activity. CMOS VLSI circuit is a realistic option to facilitate the SNN designs in real hardware implementations (Akopyan et al. 2015; Seo et al. 2011). In recent years, the emerging memristor crossbar becomes an attractive solution because the matrix-vector multiplication for synapse weight computation can be naturally conducted with its parallel architecture (Jo et al. 2010; Wang et al. 2015). Figure 1 illustrates the memristor crossbar based SNN processing. The synaptic weight can be naturally mapped to the conductance on memristor nodes, thus conducting the highly paralleled SNN processing.

2.2 Neural coding

The neural coding in SNNs can be generally categorized as rate coding, time coding, rank coding and population coding etc. (Borst and Theunissen 1999). In particularly, the first two codings are the most popular candidates in real applications, since each piece of coded information is only associated with the spikes generated by a single input neuron, offering simplified encoding/decoding procedures and design complexity. On the contrary, the latter two codings use the spike groups from several predefined cooperated neurons to represent a piece of information, meaning more complicated coding/decoding designs and larger model size because of lower coded information density per neuron. Thus, we will focus our discussions on the rate coding and time coding in this work.

In rate coding, the receptor or encoding neuron interprets the intensity of stimulus as the number of the spikes occurring in a fixed interval of time, i.e. the average firing

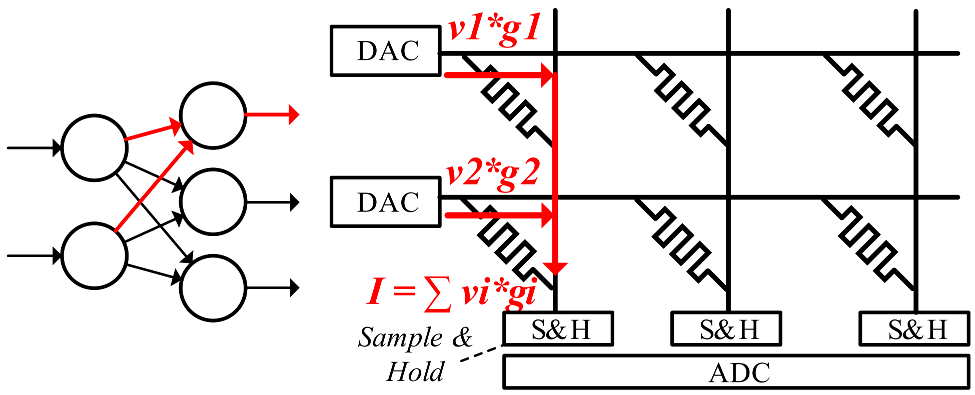


Fig. 1 An illustration of memristor crossbar based SNN processing. The SNN input and synaptic weight can be naturally mapped to the input voltage and conductance *g* on memristor nodes, thus conducting the highly paralleled SNN processing. Analog and digital mixed signal circuits are usually adopted to support the SNN processing

frequency of a Poisson-like spike train is proportional to each image pixel density (Diehl and Cook 2015). Thus, the overall spike count generated in a dedicated task is determined by the number of input stimulus and the intensity of each stimulus. The accuracy, speed and power of an SNN can be fundamentally limited by such an encoding scheme, i.e. always maintaining enough spikes in an adequate time window. However, time coding, which is inspired by the human visual system (Thorpe et al. 2001), conveys the information by the presence and occurring time of individual spikes. For example, each stimulus can be translated into a single spike but with different firing time.

2.3 Related works

Most existing researches about spiking neuromorphic designs mainly concentrate on the SNN model mapping and hardware implementation, including CMOS VLSI circuit (Akopyan et al. 2015; Seo et al. 2011; Cao et al. 2015; Esser et al. 2016), reconfigurable FPGA (Neil and Liu 2014), and emerging memristor crossbar (Chu et al. 2015; Liu et al. 2016). Recently, the spiking-based multilayer perception (MLP) and deep convolutional neuromprphic designs are also successfully developed to handle more complicated cognitive tasks (Cao et al. 2015; Esser et al. 2016). However, these works mainly focus on the rate- or time-based SNN model mapping and hardware implementations, rather than the SNN architecture optimization, i.e. coding, decoding and learning approaches etc. For example, the relevant researches are narrowly conducted on the rate-based SNNs due to the similarity between numerical information representation of ANNs and rate-based coding, and the direct deployment of learning rules from ANNs (Liu et al. 2016), while the practical architecture of time-based SNNs that can unleash its unique potentials in realistic applications haven't been well studied.

The concept of temporal coding, which relies on the arrival time or delay of a spike train for information representation, has been widely explored and proved in the development of time-based SNN (Kempter et al. 1996; Butts et al. 2007). These theoretical studies, however, mainly emphasize on the biological explanations of time-based SNN models based on simple cognitive benchmarks (i.e. two inputs XOR gate), which are far from the complicated real-world problems such as image recognition. Recently, Zhao et al. (2016) proposed an encoding circuit to handle the temporal coding, however, this type of work still concentrates on component-level hardware implementations with simple case studies, and hence is lack of a holistic architecture-level solution set capable of handling realistic tasks. In Yu et al. (2013), a complete time-based SNN design is proposed. However, their solution suffers from limited accuracy fundamentally constrained by existing coding and temporal learning rule, and is not optimized towards hardware-based neuromorphic system designs.

Since the popular learning approaches such as backpropagation (Rumelhart et al. 1988) widely used in ANN or rate-based SNN are unable to handle precise-time-dependent information due to a fundamentally different neural processing, many proposals dedicated to the time-based learning have been developed Sjöström and Gerstner (2010), Gütig and Sompolinsky (2006) and Ponulak (2005). However, these learning algorithms are neither hardware-favorable nor applicable for realistic tasks due to the expensive convergence and theoretical limitation. For example, in the unsupervised Spike-timing dependent plasticity (STDP) learning rule, the neural network structure and synaptic computation will be exponentially increased due to the expensive convergence and clustering. The proposed "Tempotron" and "Remote Supervised Method (ReSuMe)" can use the teaching spike to adjust desired spiking time for temporal learning, however, are not applicable to handle complicated patterns.

2.4 Motivation: time-based single-spike neural coding

Figure 2 demonstrates an example of conceptual comparison between rate coding and time coding in SNNs. As shown in Fig. 2, T_e and T_i (R_e and R_i) denote two types of input neurons: the time-coded (rate-coded) excitatory and inhibitory neurons, respectively. The excitatory neuron can exhibit an active response to the stimulus while the inhibitory neuron intends to keep silent. T_1 and T_2 (R_1 and R_2) denote two time-coded (rate-coded) output neurons for the classification. The rate-based SNN generates far more number of spikes than that of time-based SNN in both types of input neurons. After the input spikes are processed by the two different SNNs, a single spike firing at a specific time interval can perform an inference task in the output layer of the time-based SNN. However, a considerable number of spikes are needed for fulfilling a rate-based classification in the rate-based SNN, indicating a much higher power consumption. Moreover, the rate-based SNN may exhibit a slower processing speed than that of time-based SNN, since the output neuron of the former SNN needs to count the spiking numbers (i.e. through Integrate-and-Fire Burkitt 2006) in the whole predefined time window, while that of the latter one may quickly suspend its computations once a spike is detected.

Based on our observation on the time-based single-spike neural coding, we propose the "*FPT-Spike*" architecture which is substantially different from previous works. We explore how the time-based single-spike SNN architecture can be designed to perform the realistic tasks through a holistic efficient techniques spanning time-based coding, learning to decoding. In particular, a low cost and efficient

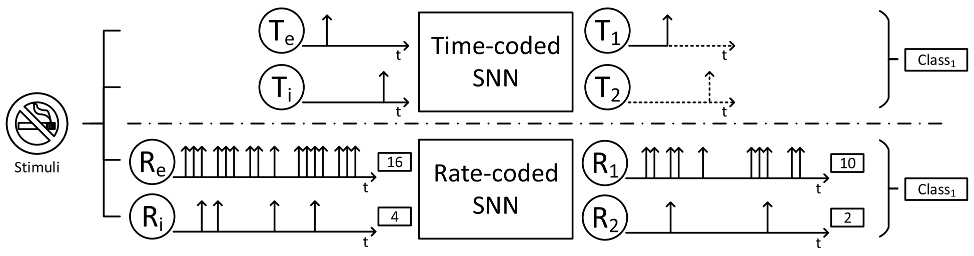


Fig. 2 The conceptual view of rate-coding and time-coding in SNNs. T_e and T_i (R_e and R_i) denote two types of input neurons: the time-coded (rate-coded) excitatory and inhibitory neurons, respectively. The excitatory neuron can exhibit an active response to the stimulus while the inhibitory neuron intends to keep silent. T_1 and T_2 (R_1 and R_2) denote two time-coded (rate-coded) output neurons for the classification, based on the spike timing (or spike rate). A considerable number of spikes are needed in the rate-based SNN, indicating a much higher power consumption and a slower processing speed than that of time-based SNN

temporal learning named "PT-Learning" is augmented from the "Tempotron" learning by considering a synthesized contribution of the cost function and the hardware-favorable time-dependent kernel for weight updating. By integrating with proposed "Precise Temporal Encoding" and "Asymmetric Decoding", "*FPT-Spike*" can improve the accuracy, power, learning efficiency, and the model size reduction through the spatial–temporal information conversion significantly.

3 Design details

To unleash the potentials of the time-based SNN in performance, power and speed, in this section, we present the design details of our proposed "*FPT-Spike*"—a flexible precise-time-dependent spiking neuromorphic architecture. We first introduce the system architecture of the "*FPT-Spike*", which is centered around the single-spike neural information processing with a capability of flexible spatial–temporal information conversion for different design trade-offs. Three key techniques: flexible precise temporal encoding–"FPT-Encoding", efficient supervised temporal learning with partial weights updating–"PT-Learning", and fast asymmetric decoding–"A-Decoding" will be also discussed in this section.

3.1 System architecture

As an SNN realization, our "*FPT-Spike*" design is inspired from biological spiking neuron models. As Fig. 3 shows, our system mainly consists of following three components:

1. Flexible precise temporal encoding: Translating the sampled stimulus into a precise-time-dependent spike train followed by a low cost spike kernel modulation, as well as providing a flexible design trade-off through adjustable coding resolution of the "FPT-Encoding" technique;
2. Synaptic processing: Performing the computations, i.e. matrix-vector multiplication, IFC etc., based on modulated spike kernels and associated synaptic weights, as well as conducting synaptic plasticity by partially updating the weights through "PT-Learning";
3. Asymmetric decoding: Decoding the output spike through "A-Decoding" with a proposed asymmetric rule to results classification for learning and inference.

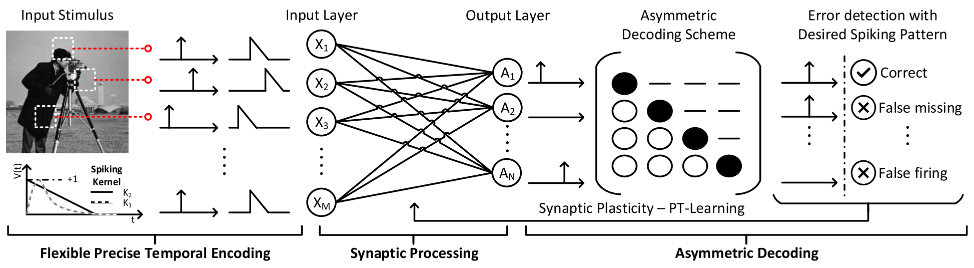


Fig. 3 The overview of "*FPT-Spike*" system architecture, including three components: Flexible precise temporal encoding, Synaptic processing and Asymmetric decoding. First, the stimulus will be captured by the temporal perceptors to generate a single spike in temporal domain and modulated to a voltage vector through a spiking kernel function. For example, we use a simple linear-decayed kernel in our design to replace the original double exponential kernel function; Second, voltage pulses will be sent to the synaptic network for a weighting process and to generate output spike trains; Then output spike trains will be transmitted to the asymmetric decoder for classification and error detection. In learning procedure, the detected errors will be sent-back for synaptic plasticity

Figure 3 also depicts a comprehensive data processing flow of proposed "*FPT-Spike*". First, the stimulus will be captured by the temporal perceptors to generate a sparse spike train (i.e. single spike) with a flexible temporal coding resolution. Each spike train will be further modulated in time or temporal domain by a linear-decayed spiking kernel. Note here the information for each input neuron are carried by the time-dependent voltage pulse. Second, those voltage pulses will be sent to the synaptic network for a weighting process, i.e. the memristor crossbar with IFC design can be employed for parallel processing. The output neurons will exhibit time-varying weighting responses due to the time-dependent input information. After that, the output neuron will fire a spike if the weighted post-synaptic voltage is higher than a threshold voltage. Then spike trains from the output layer will be transmitted to the asymmetric decoder. Finally, the target pattern will be classified by analyzing the synchronized output spikes with a predefined asymmetric rule. During the learning procedure, desired spike patterns are coded by following the similar asymmetric rule during decoding. The detected errors will be sent-back for synaptic plasticity through the proposed "PT-Learning"—a supervised temporal learning algorithm.

3.2 FPT-encoding and spatial–temporal information conversion

As discussed in Sect. 2, in rate coding, a large number of spikes within a proper time window will be needed for to precisely indicate the amplitude of an input signal, i.e. the pixel density of visual stimulus. Such a one-one mapping between each stimulus and spike train of each input neuron can lead to a significant energy overhead. Meanwhile, the time or temporal information of those spike trains are not fully leveraged by each neuron. Hence, we investigate several existing temporal coding techniques and further propose the "FPT-Encoding" to efficiently translate the stimulus into the temporal domain of the generated spikes. To maximize the power efficiency with minimized number of spikes, the input information will be represented as an extreme sparse train–single spike and its occurring delay in all aforementioned coding approaches.

3.2.1 Efficient temporal encoding

We explored a typical temporal coding technique widely used in the practical image cognitive tasks (i.e. MNIST dataset)—the "1-1 coding", that is, each single pixel is directly mapped to an input neuron with its spiking delay inversely proportional to that pixel density, as demonstrated in Fig. 4a. Here the spike delay is $D_i = T \times round\left(1 - \frac{Pix_i}{max(Pix_i)}\right)$, where T is the selected length of encoding time, Pix_i is the density of the pixel i, $i = 1, 2, \dots, N$. N denotes total number of pixels or input neurons to represent an input image. We also define that there is no spike if $Pix_i = 0$. As we shall present later, our results show that the "1-1 coding" achieves very unacceptable training accuracy (($\sim$ 20%) even under a large model size, that is, 784 input neurons for a 28×28 image. The reason is that the majority of generated spikes are centralized around just a few time slots (low coding efficiency) and unable to differentiate different patterns in the temporal domain (see the distribution of spiking delays of "1-1 coding" in Fig. 4b).

A more effective coding may be achieved if the spike occurs at each time interval with almost equal probability, thus to greatly enrich the temporal information needed in the training and testing. Inspired by this observation, we further propose two "FPT-Encoding" schemes–"Random n-1 coding" and "Conv-like coding" by perceiving the localized information from multiple interested pixels, i.e. spiking delay is equal to the number of "0s" among several selected binarized pixels. As Fig. 4a shows, "Random n-1 coding" can map n random pixels into an input neuron, in which a temporal delay is derived from multiple random selected binary pixels; "Conv-like coding" is inspired from human visual cortex and Convultional Neural Network (CNN), where a convolution kernel (i.e. a unit square matrix) will be applied on the full image (binarized pixels) to capture the spatial information and then translated into a single spike delay in temporal domain. The spiking delays of "Conv-like coding" are almost evenly distributed across the whole time domain (See Fig. 4b), which indicates the best utilization of temporal representation and best accuracy among the three candidates, as we will show in Sect. 4.

3.2.2 Spatial–temporal information conversion for model size reduction

In "Conv-like coding" or "Random n-1 coding", an input image can be represented by a pair of M input neurons (spatial domain) and certain amount of time information per neuron determined by the number of interested pixels (i.e. temporal coding resolution R, in temporal domain). Adjusting R can directly impact the information entropy of temporal domain, e.g. encoding time frame T, thus to change M accordingly. Hence, besides the power efficiency over rate coding, another unique advantage of the proposed temporal coding is to offer a flexible spatial–temporal information conversion, which enables a possible neural network model size reduction. Figure 4c demonstrates the concept of spatial–temporal trade-off to realize the model size reduction based on "Conv-like coding". In this example, we assume the "baseline design" as follows: $M = 6$ input neurons, 12 synaptic weights (spatial domain) with a given coding time window (temporal domain) can

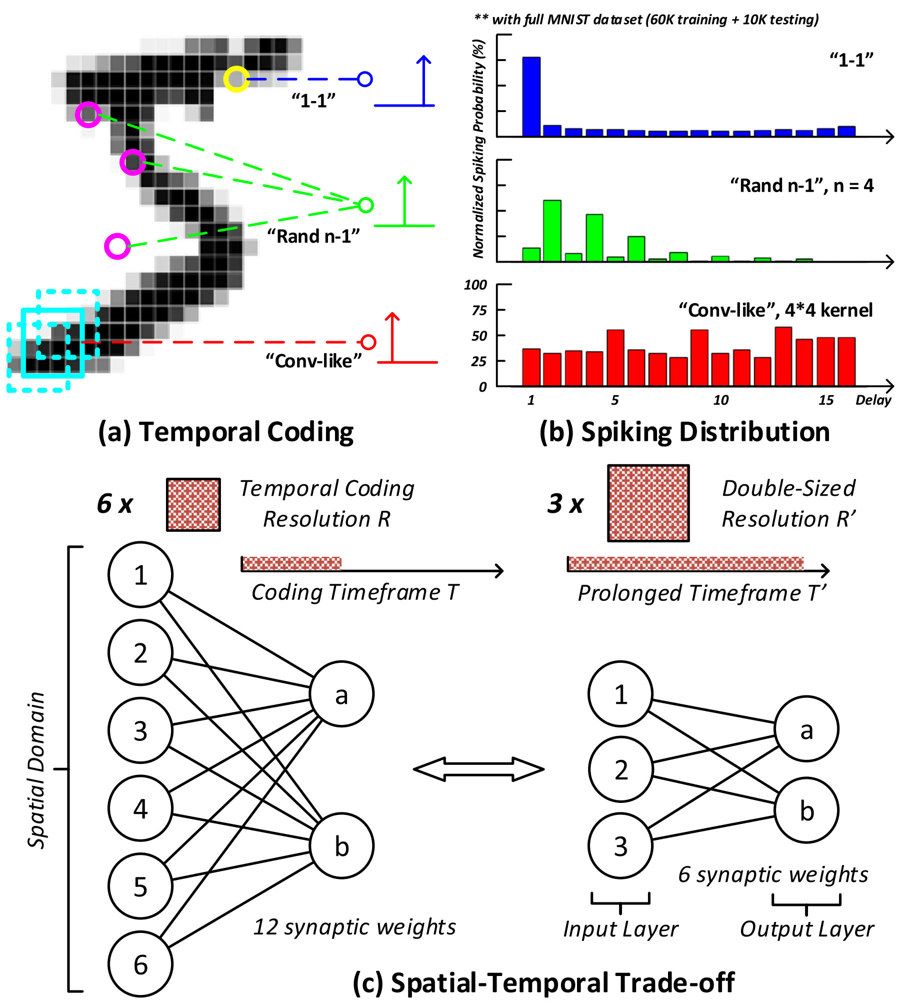


Fig. 4 The design exploration of "FPT-Encoding" and "Spatial–Temporal Information Conversion". **a** "1-1 coding" is based on the direct mapping from single pixel to an input neuron with spiking time inversely proportional to pixel density while "Random n-1 coding" and "Conv-like coding" are based on binary-entropy of multiple pixels and have the capacity to offer the flexible spatial–temporal trade-off for model reduction; **b** Histogram of spiking delays for three different coding schemes on the full MNIST dataset. The spiking occurring at each delay interval with a similar probability may enrich the encoding information of input neurons, leading to a more effective temporal encoding and stronger learning capability. Compared with the other candidates, "Conv-like coding" is showing the well-distributed spiking across the whole time domain and is adopted in our design; **c** A flexible spatial–temporal trade-off is developed for neural network model size reduction by adjusting the temporal coding resolution

represent the entire information of the image. An alternative "size-reduced design" can offer the similar information with only $M = 3$ input neurons, 6 synaptic weights (50% less spatial information) by enlarging the kernel size or encoding time (more temporal information). Therefore, "FPT-Spike" can offer various design options (i.e. neural network size and processing speed) to satisfy different application requirements. However, different "FPT-Spike" designs can vary in accuracy, speed and number of synaptic connections etc., as we will discuss later in Sect. 4.

3.2.3 Linearized spiking kernel

Once the delay for the single spike is determined, as shown in Fig. 3, a spiking kernel K will be applied to shape the associated spikes for input neurons. The kernel plays an important role in the following synaptic weighting for the output voltage $V_n(t)$, as shown in Eq. (1):

$$V_n(t) = \sum_{m}^{M} w_{mn} \sum_{t_s}^{T} K(t - t_s) \tag{1}$$

where weight $V_n(t)$ represents the voltage of output neuron n, w_{mn} denotes the synaptic efficacy between input neuron X_m and output neuron A_n. t_s is the decoded spiking delay of X_m.

To provide sufficient and accurate temporal information for the classification, the exponential decayed post-synaptic potential in the biological spike response neural model (Gerstner 2001) can be expressed as:

$$K_1(t - t_s) = \mu(exp[-(t - t_s)/\tau_1] - exp[-(t - t_s)/\tau_2]) \tag{2}$$

where τ (τ_1 and τ_2) denotes decay time constant, and μ is the normalizing constant.

However, such an exponential decaying function requires expensive computation and hardware resource. In "*FPT-Spike*", we employ a more hardware-favorable kernel function K_2—a linear decaying function (see K_1 and K_2

comparison in Fig. 3), to simplify the costly dual-exponential function K_1:

$$K_2(t - t_s) = 1 - \tau(t - t_s) \tag{3}$$

As we shall show later, such a linear approximation cause very marginal classification accuracy degradation.

3.3 Asymmetric decoding

In '*FPT-Spike*", a novel Asymmetric decoding scheme, namely "A-Decoding", is proposed for the classification. As the error signal for the proposed supervised temporal learning will be also generated through asymmetric decoding, we will discuss the "A-Decoding" technique first.

In rate-based SNN, the target pattern can be directly determined by the output neuron with highest spiking numbers, which is similar to the "one-hot" coding based calssification in traditional neural network. Since all the output neurons will be participating in the learning process of each target pattern, the costly weight updating will be performed in all synapses at each iteration of learning. The subsequent neural competition (weight conflict) among different patterns can be rectified by enough information provided by the large number of input spikes. Hence a good classification accuracy may be achieved for all different patterns. However, the similar case cannot occur in our proposed "FPT-Spike", since its weight updating solely relies on the very limited number of spare spikes (e.g. a single spike) in temporal domain. Our investigations show that the synaptic weight conflict becomes more serious among different patterns due to the similar weak weight updating, indicating a significant accuracy degradation. Hence, we propose the "A-Decoding" to alleviate the neural competition for accuracy improvement.

Figure 5 illustrates the key idea of proposed "A-Decoding", including pattern readout and error detection. Pattern $\{P_i\}$ can be decoded based on the firing status of output neuron $\{N_i\}$. In our asymmetric decoding, the output neuron can work on three different statuses: "firing", "not firing" and "independent", as shown in Fig. 5. Note "independent" means that the associated neurons will not participate in the learning process of a certain pattern, and it will only occur in learning mode.

In testing mode, the output neuron will be only in following two status: $\{1 - \mathit{firing}/0 - \mathit{notfiring}\}$. The target pattern is scanned according to the order of the first firing neuron. Assume a binary code $\tilde{N}_1\tilde{N}_2\tilde{N}_3 \cdots \tilde{N}_i$ is generated by output neurons $\{N_i\}$, a Huffman-style decoding procedure can be performed (See Fig. 5 left part). For example, if the first firing neuron is N_3, the corresponding code will be $\tilde{0}\tilde{0}\tilde{1}$. Thus, the target pattern is P_3. In "*FPT-Spike*", the early detection

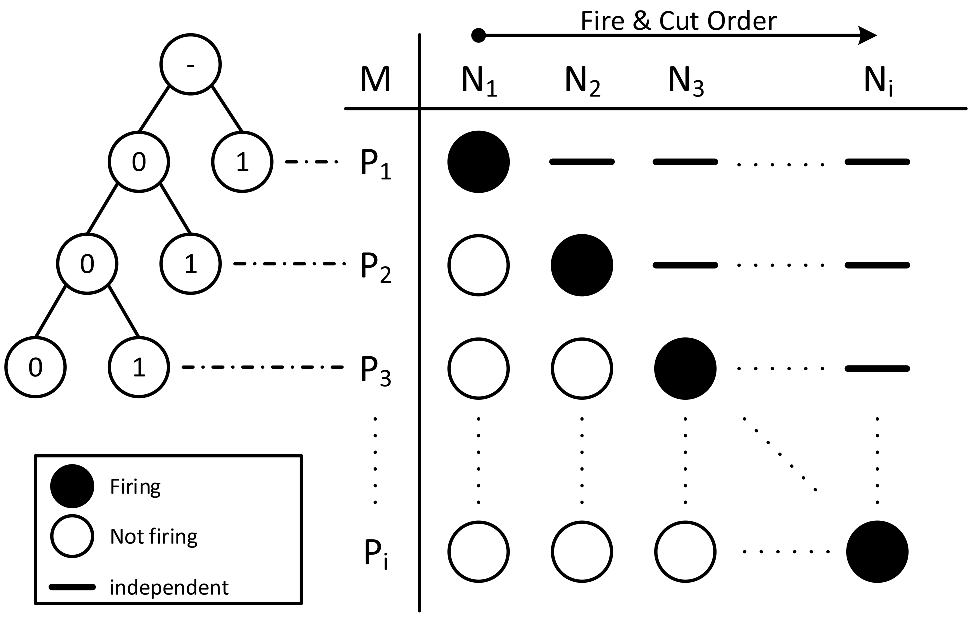


Fig. 5 An overview of proposed "A-Decoding". Output neuron can work on three different statuses: "firing", "not firing" and "independent". In testing mode, the target pattern is scanned according to the order of the first firing neuron through a Huffman-style decoding (left part). System may perform an early-detection by following the "Fire&Cut Order". In learning mode, a desired spike pattern can be reversely generated according to the Huffman-style decoding of pattern $\{P_i\}$ (right part) and "independent" status will not be involved during the learning, leading to a fast learning speed. Once a participated neuron triggers an unexpected fire or a missing fire, an error will be detected and its synaptic weights will be modified according to "PT-learning". Such an asymmetrical correlation between $\{N_i\}$ and $\{P_i\}$ can ease the neural competition among the output neurons and boost the accuracy

of testing, namely “Fire&Cut”, can be realized based on the temporal “winner-take-all” rule: Once the IFC of neuron N_i triggers a spike, all the remained IFCs for other neurons will be shut down by following the “Fire&Cut Order”, which may save the additional power consumed by the IFCs. In learning mode, a desired spike pattern is reversely generated according to the Huffman-style decoding of pattern $\{P_i\}$ (See Fig. 5 right part). Once a participated neuron N_i triggers an unexpected firing or a missing firing, an error will be detected and only the synaptic weights of N_i will be modified according to our proposed “PT-learning”. Note only “partial” output neurons (NOT in“independent” status), will be involved during the learning of pattern $\{P_i\}$, namely “Partial Learning”. Such a mechanism significantly accelerates the learning procedure and saves power consumed by the unnecessary neural processing. Meanwhile, $\{N_i\}$ is “asymmetrically” correlated with $\{P_i\}$ and thus can ease the neural competition. For example, neuron N_i only engages in the synaptic plasticity of pattern P_i and will be ignored during the learning of all other patterns. As we shall show later, by taking advantages of “Fire&Cut”, “Partial Learning” and “Ease Competition”, our proposed “A-Decoding” can significantly enhance the weighting efficiency and learning accuracy.

Algorithm 1: Post-Synaptic Processing

```
// Pseudocode of Asymmetric Decoding and PT-Learning
1  Detecting:
2  foreach output neuron $N_i$ in [$N_1$ .. $N_I$] do
3      if testing mode then
4          if firing then
5              return $P_i$// “Fire&Cut”
6      else
           // learning mode
7          if $N_i$ is independent to $P_i$ then
8              return// “Partial Learning” and “Ease Competition”
9          else if actual firing pattern $\neq$ desired pattern then
10             call Learning($V_{max}$, $T_{max}$)
11 Learning:
   // change synaptic weights of $N_i$
12 Err $\leftarrow V_{th} - V_{max}$
13 $T_{fal} \leftarrow T_{max}$
14 foreach input neuron $X_c$ in [$X_1$ .. $X_M$] do
15     if $K_2(T_{fal} - T_c) \leqslant 0$ then
16         continue// “Partial Updating”
17     else
           // pre-spiking at $T_c$ contributed to post-spiking
18         $\Delta w \leftarrow \lambda Err K_2(T_{fal} - T_c)$
19         $w_{ci} \leftarrow \Delta w + w_{ci}$
```

3.4 PT-learning for light-weighted online training

In “*FPT-Spike*”, only a single precisely delayed spike is needed to represent the coded information. However, the popular learning approaches such as Perceptron (Rumelhart et al. 1985) and Back-propagation (Rumelhart et al. 1988) in ANN or rate-coded SNN are unable to handle precise-time-dependent information due to the fundamentally different neural processing. Meanwhile, the existing time-based learning algorithms are usually too costly to be applied in real hardware implementation. For example, the unsupervised STDP may handle complex realistic tasks, however, the neural network structure and synaptic computation will be exponentially increased due to the expensive convergence and clustering (Legenstein et al. 2005). The supervised algorithm, such as Tempotron (Gütig and Sompolinsky 2006) and Remote Supervised Method (ReSuMe) (Ponulak 2005), are either not hardware-favorable or incapable to handle the real-world applications. Thus, we propose the “PT-Learning”– an applicable supervised learning algorithm with accelerated “Partial weights updating” dedicated for the proposed single-spike Temporal architecture .

“PT-Learning” coordinates with the aforementioned “A-Decoding” to capture the errors needed for synaptic weights updating. An error detected by the “A-Decoding” will be processed by “PT-Learning” to generate corresponding weight changes and send back for synapse updating. As shown in Figs. 3 and 6a, two types of errors may occur in the output neuron: “false missing” (Stimulus A) and “false fire” (Stimulus B). Here “false missing” means that the neuron does not fire a spike expected by the training pattern, while “false fire” is defined as an undesired spike firing.

To develop a low cost and efficient temporal learning algorithm, “*FPT-Spike*” is further augmented from the Tempotron learning by considering a synthesized contribution of the error and the time-dependent kernel for weight updating. As shown in Algorithm 1, once an error is detected, the error spiking time (T_{fal}) and the cost function (Err) will be extracted from T_{max} and $V_{th} - V_{max}$. Here V_{max} and T_{max} are the maximum membrane potential and its (potential) spiking time, respectively. As shown in Fig. 6a, a negative (positive) Err means a false- fire (missing). Hence, the gradient of Err with respect to each weight w_c at pre-synaptic spiking time T_c can be calculated as:

$$-\frac{\mathrm{d}Err}{\mathrm{d}w_c} = Err \sum_{T_c \leq T_{max}} K_2(T_{max} - T_c) + \frac{\partial V(T_{max})}{\partial T_{max}} \frac{\mathrm{d}T_{max}}{\mathrm{d}w_c} \tag{4}$$

Here K_2 is the linear decayed spike kernel defined in Eq. (3). The summation include all pre-synaptic spikes arriving before T_{max} which are weighted through the synaptic efficacy w_c. The second term indicates the derivative between the “spiking voltage” and “error time”; however, it will not contribute to the synaptic update since $\frac{\partial V(T_{max})}{\partial T_{max}} = 0$ due to the definition of T_{max}. Therefore, by further considering Err into the change of w_c, Δw_c can be expressed as:

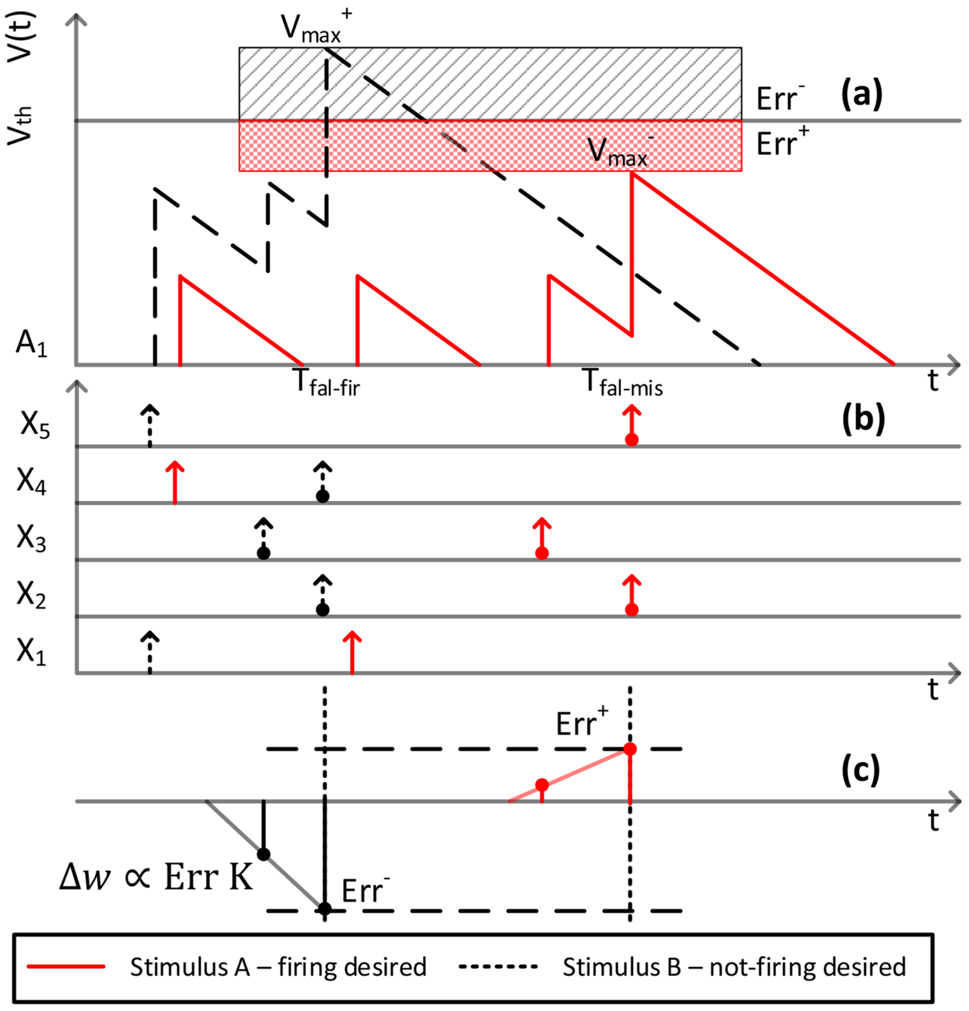


Fig. 6 Illustration of the basic idea of PT-Learning. **a**, **b** The input spike trains of $\{X_M\}$ will be modulated and integrated to generate output voltage pulses on each output neuron $\{A_N\}$ and output spike will be triggered once the integrated voltage crosses the threshold. Two types of errors may occur according to the undesired output spiking status: "false missing" and "false fire". Once an error is detected, the error spiking time (T_{fal}) and the cost function (Err) will be extracted to calculate weight adjustment; **b**, **c** Spiking kernel K_2 is used reversely to calculate the voltage contributions from the input neuron X_c at time T_c ($=T_{fal}$). The weight increment Δw_c can be by further calculated with Err and K_2 for "partial weights updating" only on the synapses connected with contributed neurons

$$\Delta w_c = \lambda Err \sum_{T_c \le T_{fal}} K_2(T_{fal} - T_c) \tag{5}$$

where λ denotes the learning rate and spike kernel K_2 can be used again to calculate the contributions from the input neuron X_c at time T_c. As depicted in Fig. 6c, for each output neuron A_N, Δw_c, which is determined by the error and the time-dependent linear kernel (see Eq. 5), will be only applied to "partial" synapses connected with the contributed neurons X_c (i.e. neuron X_2, X_3 and X_5 in the case of "false-missing" and neuron X_2, X_3 and X_4 in the case of "false-fire" from Fig. 6b).

As discussed in "A-Decoding", only partial output neurons will be involved during the learning of a certain pattern, meaning that only partial synaptic weights will be updated. The dual-level acceleration, contributed by both "A-Decoding" and "PT-Learning", can improve the learning efficiency significantly. As we shall show later, the synaptic computation can be reduced more than 200% when compared with the standard learning approach without accelerations. Moreover, "PT-Learning" together with "A-Decoding" can boost the accuracy for realistic recognition task significantly.

3.5 Time-based error back-propagation for powerful offline training

Our developed "PT-Learning" is a highly efficient learning rule to enable the *online training* for proposed time-based single-spike neuromorphic architecture. We use the voltage difference and the threshold as an error function and a non-differentiable activation function, respectively, which is incompatible with the traditional back-propagation learning rule. This online training design can only be applied to the simple neural network structure, i.e., a single layer. We further develop the time-based error back-propagation learning rule, namely "TBP-Learning", which can be used as an *offline training* approach to improving the learnability of proposed "*FPT-Spike*" architecture.

In our exploration, we find that the time-based neural processing can be approximated as:

$$d_j(w_{ij}, d_i) = \frac{1}{n} \sum_{i=1}^{n} w_{ij} d_i \tag{6}$$

where w_{ij} is the synaptic weight between neuron i and j, d_i is encoded spike delay of neuron i in time-based SNN, and n is number of post-synaptic neurons. In particular, the spike delay d_j of output neuron j can be directly tuned by weights w_{ij} through the approximate average function. Based on such an approximation of time-based neural processing, we can define a temporal error function on output neuron j:

$$E_j = \frac{1}{2}\left(d_{t(j)} - d_{a(j)}\right)^2 \tag{7}$$

where $d_{t(j)}$ is its target spike delay and $d_{a(j)}$ is the actual spike delay. Therefore, the output delay of neuron j is given as:

$$d_j^l = \varphi(net_j^l) = \varphi\left(\frac{1}{n}\sum_{i=1}^{n} w_{ij}^l d_i^{l-1}\right) \tag{8}$$

where d_i^{l-1} is the pre-synaptic delay of the i-th neuron and φ is the implicit activation function based on the approximate average function. Therefore, the partial derivative of temporal error with respect to weight w_{ij}^l can be expressed as:

$$\frac{\partial E_j}{\partial w_{ij}^l} = \frac{\partial E_j}{\partial d_j^l}\frac{\partial d_j^l}{\partial net_j^l}\frac{\partial net_j^l}{\partial w_{ij}^l} \tag{9}$$

which is similar to the traditional error back-propagation. In particular, the derivative of last term can be calculated as:

$$\frac{\partial net_j^l}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}^l}\left(\frac{1}{n}\sum_{i=1}^{n} w_{ij}^l d_i^{l-1}\right) = \frac{d_i^{l-1}}{n} \tag{10}$$

Based on the approximate average function, the derivative of mid-term can be also simplified as:

$$\frac{\partial d_j^l}{\partial net_j^l} = \frac{\partial}{\partial net_j^l}\varphi\left(\mathrm{net}_j^l\right) = 1 \tag{11}$$

and the derivative of first term for neuron j at output layer l is:

$$\frac{\partial E_j}{\partial d_j^l} = \frac{\partial E_j}{\partial d_{a(j)}} = \frac{\partial}{\partial d_{a(j)}}\frac{1}{2}(d_{t(j)} - d_{a(j)})^2 = d_{a(j)} - d_{t(j)} \tag{12}$$

thus we have:

$$\frac{\partial E_j}{\partial w_{ij}^l} = \frac{d_i^{l-1}(d_{a(j)} - d_{t(j)})}{n} \tag{13}$$

in particular, for the neuron j at hidden layer(s), we have:

$$\frac{\partial E_j}{\partial d_j^l} = \sum_{k=1}^{n}\left(\frac{\partial E_j}{\partial net_k^{l+1}}\frac{\partial net_k^{l+1}}{\partial d_j^l}\right) = \sum_{k=1}^{n}\left(\frac{\partial E_j}{\partial d_j^l}\frac{\partial d_j^l}{\partial net_k^{l+1}} w_{jk}^{l+1}\right) \tag{14}$$

where k is the post-synaptic neuron of j, by defining:

$$\delta_j^l = \frac{\partial E_j}{\partial d_j^l}\frac{\partial d_j^l}{\partial net_j^l} = \begin{cases} d_{a(j)} - d_{t(j)} & , l \textit{ is output layer} \\ \sum_k \delta_k^{l+1} w_{jk}^{l+1} & , l \textit{ is hidden layer} \end{cases} \tag{15}$$

therefore, we can obtain the weight updating at learning rate η as:

$$\Delta w_{ij}^l = -\eta\frac{\partial E_j}{\partial w_{ij}^l} = -\eta\delta_j^l\frac{d_i^{l-1}}{n} \tag{16}$$

3.6 Pipeline-based neural processing scheme

Based on the proposed online "PT-Learning" and offline "TBP-Learning", the "*FPT-Spike*" architecture can be implemented through several different approaches: 1) For the light-weighted cognition tasks on a simple SNN, we follow the end-to-end procedure as shown in Fig. 3 to implement the "*FPT-Spike*" with online "PT-Learning" rule. This implementation requires the analog and digital mixed signal circuits to encode spike delay and to modulate the spiking kernel. 2) For the complex cognition tasks on a large-scaled deep SNN, we first to train the weights of SNN through the offline "TBP-Learning". Then, we apply the trained weights and conduct the feed-forward time-based neural processing for inference.

Besides, we also devised an alternative implementation – the pipeline-based neural processing scheme for digital spikes. In our proposed "TBP-Learning", the neural network weights can be directly tuned to adjust the spike delays. Such a design also enables the alternative digital spike (e.g., binary spikes "0-non-fire/1-fire") based implementation without evolving the spiking kernel modulation on analog and digital mixed signal circuits, therefore to reduce the implementation cost significantly. However, unlike the two implementations above, the digit spike is event-driven incompatible, i.e., a neuron will not be updated only when it receives or emits a spike asynchronously. Therefore, we propose the pipeline-based neural processing scheme to improve processing efficiency.

As shown in Fig. 7, the pipeline-based neural processing scheme consists of two processing phases, i.e., "SR—Spike Receiving" and "SG—Spike Generation". Such pipeline-based neural processing will follow the similar layer-wised processing order on SNNs. In the first computing cycle CC1, input data will be encoded into a time-based spike train (single-spike with a certain delay) during the SR phase on the input layer. After that, the spike train will be processed layer-wisely through the SG+SR pipeline until the output spike train is generated during

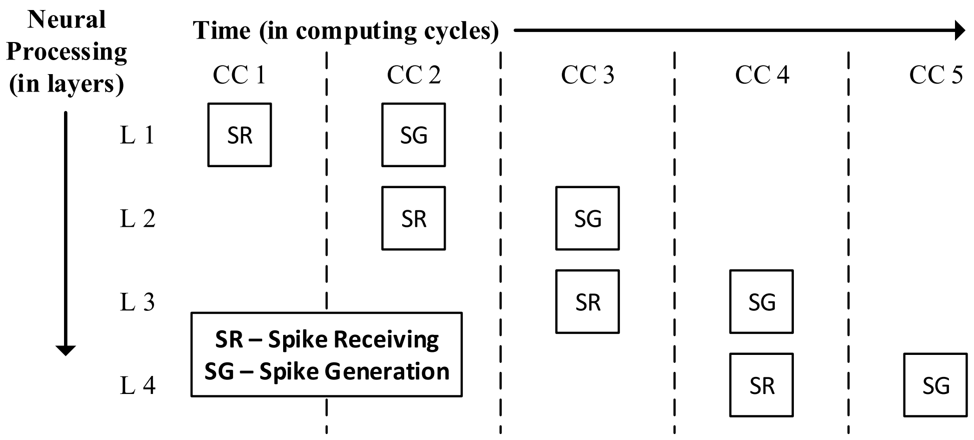


Fig. 7 An illustration of the pipeline-based neural processing scheme. The 2-step layer-wised neural processing consists of "SR-Spike Receiving" and "SG-Spike Generation" phases. For the first layer L1 (input layer) in the first computing cycle CC1, input data will be encoded into a time-based spike train during the SR phase. Start from the second computing cycle CC2, spike train will be processed in layer-wise through SG+SR pipeline until the final output spike train is generated during the SG phase on the output layer

the SG phase on the output layer. The output spike train will be further decoded into the cognitive results. Compared with the event-driven spiking kernel-based design, the pipeline-based neural processing scheme will consume one extra computing cycle in inference, while saving the implementation cost.

4 Evaluations

To evaluate the accuracy, processing efficiency and power consumption of our proposed "*FPT-Spike*" neuromorphic architecture, extensive experiments are conducted in the platforms like MATLAB and heavily modified open-source simulator–Brian (Goodman and Brette 2009).

4.1 Simulation setup

In our evaluation, a full MNIST database is adopted as the benchmark, which includes 60K training images and 10K testing images (LeCun et al. 1998). A set of "*FPT-Spike*" designs–"FPT-Spike(R)" with temporal resolution "R", are implemented to demonstrate the advantage of flexible "spatial–temporal information trade-offs". Note the adjustable temporal resolution "R" denotes the number of interested pixels per input neuron, and it is a square number when we adopt "FPT-encoding"–"conv-like coding" in all "FPT-Spike(R)" designs. We also assume the encoding time frame (T) is $T = \tau \times R(ms)$, where $\tau = 1(ms)$ is the fixed minimum time interval to fire the spike. The maximum temporal information T can be adjusted by tuning the parameter R. The number of input neurons (spatial domain) can be expressed as $M = \lceil \frac{P-\sqrt{R}+1}{S} \rceil^2$, where P and S represent the width of an input image and the stride with which we slide the filter– the convolution kernel of the"conv-like coding", respectively. $P = 28$ and $S = 2$ are selected in our evaluations of MNIST dataset. Similar as the convolution computation in CNN, zero padding will be applied for each input image if necessary. Two representative baselines under similar network configurations, including the rate-coded SNN–"Diephl-15" Diehl and Cook (2015) and the ANN–"Lecun-98" LeCun et al. (1998), are also implemented for the energy and performance comparisons with proposed "*FPT-Spike*". Table 1 presents the structural

Table 1 Structural parameters of selected candidates

Candidate	Number of input neurons	Number of output neurons	Number of synaptic weights	neural processing time-frame T
FPT-Spike(4)	196	10	1960	4 ms
FPT-Spike(16)	169	10	1690	16 ms
FPT-Spike(25)	144	10	1440	25 ms
FPT-Spike(100)	100	10	1000	100 ms
Diehl-15	784	100	78400	500 ms
Lecun-98	784	10	7840	–

parameters of selected candidates. All the evaluations are conducted based on following settings: Training data is equally fed into neural networks with three training iterations, i.e. 180K samples, followed by one testing iteration. The initial weights ($w \in (-1, +1)$) are randomly generated before training.

4.2 Accuracy and flexible spatial–temporal information trade-offs

Figure 8 shows the accuracy comparison among different "FPT-Spike (R)", the ANN–"Lecun-98" and rate-based SNN–"Diehl-15". "FPT-Spike(25)" can achieve very comparable accuracy at much lower cost ($\sim$ 86%, 1440 synaptic weights) when compared with "Diehl-15" ($\sim$ 83%, 78400 synaptic weights) and "Lecun-98" ($\sim$ 88%, 7840 synaptic weights). Meanwhile, "FPT-Spike(16)" and "FPT-Spike(25)" also show a very close accuracy ($\sim$ 87% and $\sim$ 86%), which is much better than "FPT-Spike(4)" and "FPT-Spike(100)" ($\sim$ 63% and $\sim$ 70%).

We also evaluated the individual training accuracy improvement contributed by various proposed techniques, such as "linearized spiking kernel", "conv-like coding", "A-Decoding" and "PT-Learning", receptively. Here, we choose the "FPT-Spike(16)" as the baseline design that employs all aforementioned techniques. "Exponential Kernel", "random 16-1 coding", "non A-Decoding" and "Tempotron" denote the designs that substitute only one out of the four techniques, that is, linearized spiking kernel v.s. exponential spiking kernel), "conv-like coding" v.s. "random 16-1 coding", and "A-Decoding" v.s. standard decoding, respectively. As shown in Fig. 10, "FPT-Spike(16)" shows a very marginal accuracy degradation (0.2%) because of the "linearized spiking kernel" (K_2 in Eq. (3)) when compared with the original costly "Exponential Kernel" design (86.9%, K_1 in Eq. (2)). Furthermore, "FPT-Spike(16)" boosts the accuracy by $\sim$ 13%, $\sim$ 19% and $\sim$ 38% when compared with the designs of "random 16-1 coding" ($\sim$ 74%), "non A-Decoding" ($\sim$ 68%), and the theoretical "Tempotron" learning rule ($\sim$ 49%), respectively, which clearly demonstrates the effectiveness of the proposed"conv-like coding", "A-Decoding" and "PT-Learning".

Table 1 illustrates the advantage of spatial–temporal information conversion offered by "*FPT-Spike*". Increasing the number of interested pixels per input neuron (R) means an enlarged encoding time frame T to enrich the temporal

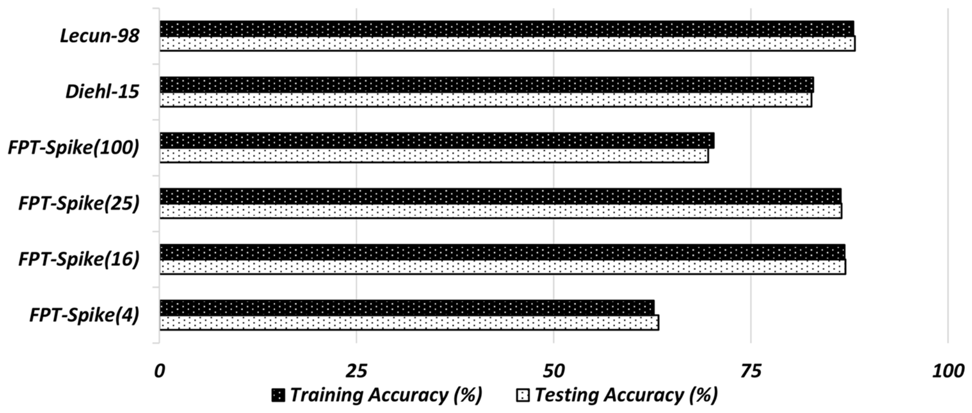


Fig. 8 Training and testing accuracy of MNIST database on selected candidates

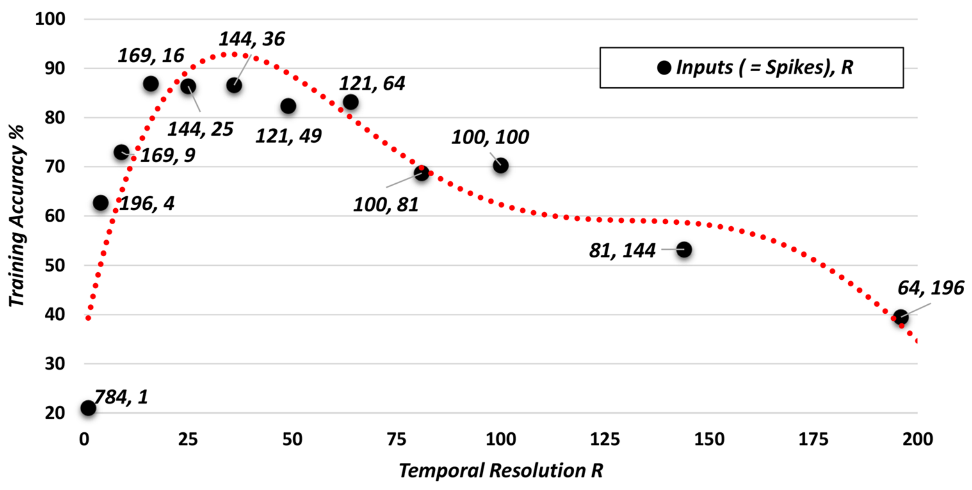


Fig. 9 Training accuracy with various spatial–temporal trade-offs

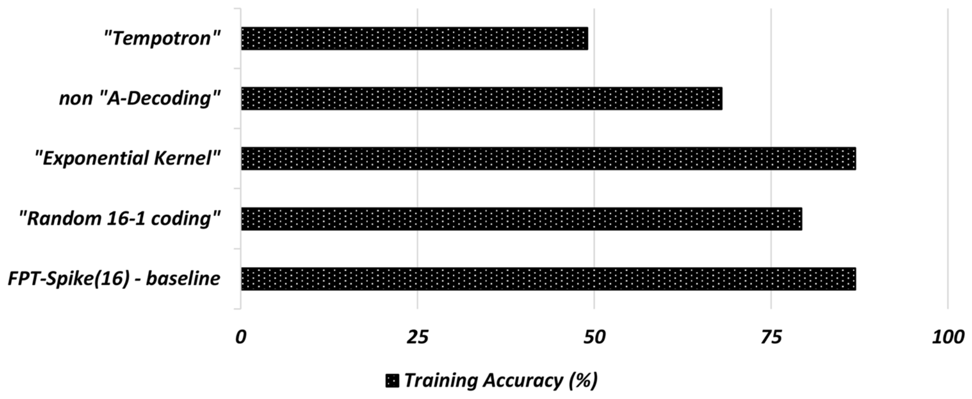


Fig. 10 Training accuracy on different design variants

information, and hence reduces the needed spatial information or number of input neurons (i.e. from $M = 196$ neurons, $T = 4$ms of “FPT-Spike(4)” to $M = 144$ neurons, $T = 25$ms of “FPT-Spike(25)”). However, as Fig. 9 shows, the training accuracy varies as R changes, which indicates that a proper combination of spatial–temporal information (M,T) is essential to guarantee a high classification accuracy. Either fast designs, e.g. (784,1), (196,4), with too limited temporal information but larger spatial information or slow designs like (81,144) or (64,196) with too much temporal information but less spatial information, can lead to very prominent accuracy degradation. Yet there still exists multiple combinations of (M,T) to achieve the best accuracy, i.e. (169,16) and (144,36) for $\sim$ 86% accuracy. Table 1 also shows that the time-based designs, i.e. “FPT-Spike(25)”, can reduce the number of synapse weights notably ($\sim$ 5× and $\sim$ 50×) as compared with “Lecun-98” and “Diephl-15” at a similar accuracy because of additional temporal information. All these results clearly demonstrate the possibility of design trade-offs between network size and speed even without sacrificing the accuracy.

4.3 Processing efficiency

The occurrence frequency of synaptic events is calculated to evaluate the system processing efficiency, including both weighting and weights updating. The weighting process in feed-forward pass consists of multiply-accumulate operations necessary in both training and testing stages of each input image, while the weights updating only occurs during the training or feed-back pass.

Figure 11 compares the number of weighting operations among three designs in the feed-forward pass. Here“non A-Decoding” design represents the “*FPT-Spike*” only without employing the “A-Decoding” technique. The first training iteration is selected as the representative feed-forward pass, given the same number of weighting operations for both training and testing stages of “Diehl-15” and “non A-Decoding”. However, the amount of weight operations of “FPT-Spike(16)” is different between training and testing due to the “Fire&Cut” mechanism in“A-Decoding”. Hence, the weighting of the first testing iteration is also included in “FPT-Spike(16)” (see Fig. 11).

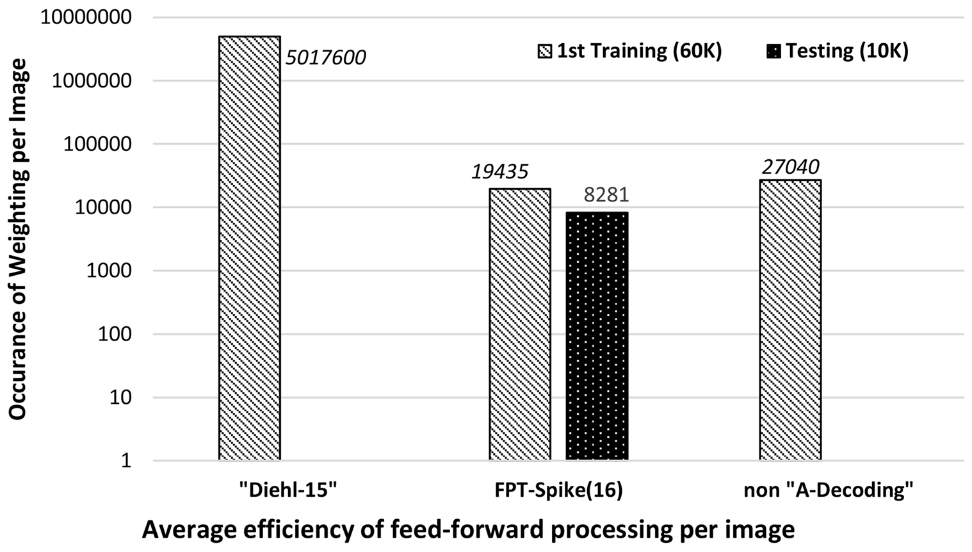


Fig. 11 Feed-forward efficiency per input image

As shown in Fig. 11, even the “non A-Decoding”, i.e. “FPT-Spike(16)” without the “A-Decoding” technique, gains ∼ 185× weighting operation reduction as compared with rate-coded SNN–“Diehl-15”. The reason is that rate-coded SNN requires a long time window to process the spikes at enlarged neuron model size, i.e. 350(ms) for “Diehl-15” Diehl and Cook (2015) v.s. 16(ms) of “FPT-Spike(16)”, causing tremendous weighting processes on each time slot. Compared with “non A-Decoding”, weighting operations of “FPT-Spike(16)” can be further reduced by ∼ 28% and ∼ 69% in first training iteration and testing iteration, respectively. As expected, the “early-detection” working mechanism in “A-Decoding” removes many unnecessary weighting operations on both “initialized” weights and “well-trained” weights.

We also characterize the occurrence frequency of weights updating during the first training iteration to evaluate the processing efficiency in the feed-back pass. Each detected error, including ‘false missing” and “false fire”, can cause a substantial number of weights updating. Thus, we average the total number of weights updating by each image and each error. In particular, weights updating “per image” can represent an overall weighting efficiency for three different designs, i.e. “Diehl-15”, “PT-Spike”, and “Worst Case”. Here “Worst Case” denotes the “FPT-Spike(16)” without employing “A-Decoding” and “PT-Learning”. As Fig. 12 shows, even “Worst Case” achieves ∼ 4.6× and 40× reductions on weights updating per image and per error, respectively, compared with “Diehl-15”. Such impressive improvement is introduced by the significant compressed model size. Moreover, compared with the “worst case”, “PT-Learning” and “A-Decoding” contribute ∼ 2× and ∼ 4× weights updating reduction per error and per image for “FPT-Spike(16)”, respectively, demonstrating the effectiveness of “dual-level acceleration” from decoding and learning.

4.4 Power consumption

To evaluate the power efficiency contributed by the proposed single-spike neuromorphic architecture, we adopted a similar methodology from Akopyan et al. (2015), Cao et al. (2015). In particular, in our measurement we assume that a single spike activity consumes α Joules of energy, despite whether the spike is pre-synaptic or post-synaptic. The total power consumption can be estimated based on this unit power consumption per single spike together with total spike numbers. In our evaluation, the total spiking energy is also calibrated in one testing iteration (10K images). Figure 13 compares the power consumption averaged by each input neuron and each image, respectively, among three different designs: “Minitaur”, “Diehl-15”,“FPT-Spike(16)”. For example, our proposed“FPT-Spike(16)” solution will consume $151.12 \times \alpha$ Joules in average on the generated spike train for encoding a single input image. The new candidate “Minitaur” Neil and Liu (2014) is introduced for a fair comparison since it is a more hardware-oriented rate-coded SNN. As Fig. 13 shows, “FPT-Spike(16)” saves ∼ 8× and ∼ 64× power for each input neuron and each input image over “Diehl-15’, respectively, indicating the efficiency of our proposed single-spike coding technique. Compared with the hardware-oriented rate-coded SNN design “Minitaur”, “FPT-Spike(16)” can still achieve ∼ 1.4× (∼ 6.6×) power reduction on each input neuron (input image).

4.5 TBP-learning

Table 2 shows the evaluation results of “TBP-Learning”. We trained a two-layer time-based SNN (169-500-10) with proposed offline time-based error back-propagation as our “TBP-Learning” candidate with the MNIST dataset. In our evaluation, we compare its testing accuracy with other

Fig. 12 Feed-back efficiency

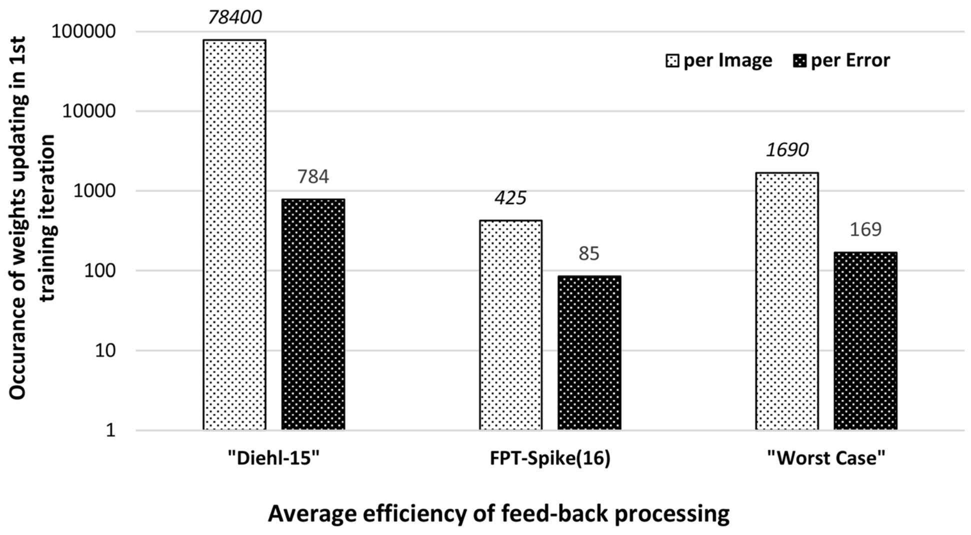


Fig. 13 Spiking power consumption (α Joules /spike)

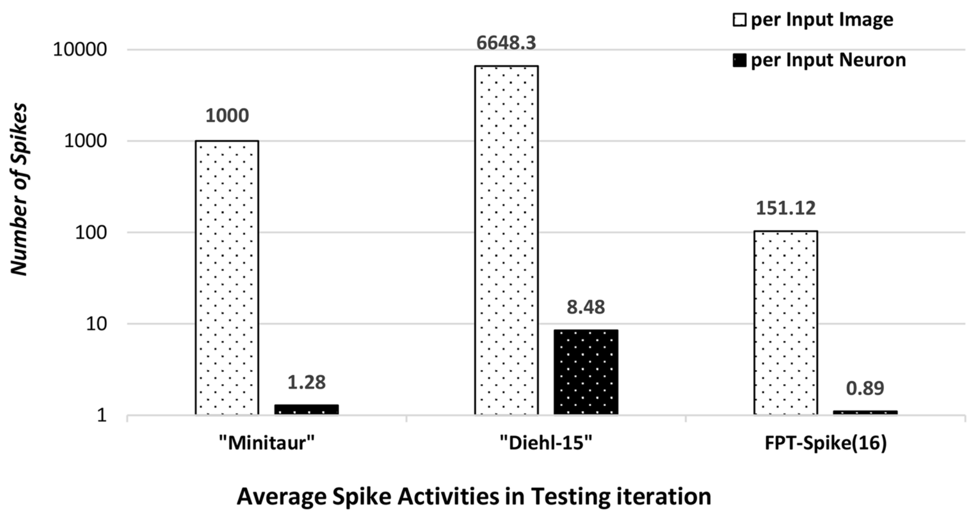


Table 2 Evaluation on TBP-Learning

Candidate	Types	Testing accuracy	Network structure
TBP-Learning	tSNN	99.1%	169-500-10
PT-Learning	tSNN	89.6%	169-10
Diehl	rSNN	95%	784-6400
Minitaur	rSNN	92%	784-500-500-10
Lenet-5	CNN	99.05%	1024-C1-S2-C3-S4-C5-F6-10

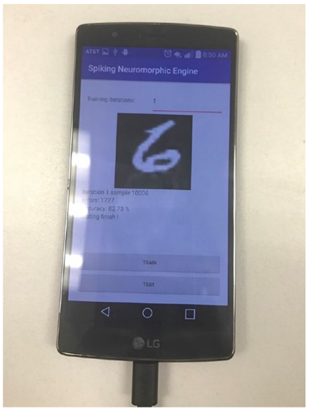


Fig. 14 An Android implementation of “FPT-Spike” with pipeline-based neural processing

candidates include the five-layer CNN “’Lenet-5”. As listed in Table 2, “TBP-Learning” can boost the testing accuracy from 89.6% (single-layer “PT-Learning”) to 99.1% through the time-based error back-propagation training on the two-layer time-based SNN, achieving the best results among all candidates and even comparable with the five-layer CNN “Lenet-5” (99.05%). In particular, compared with the hardware-oriented design “Minitaur” (92%), “TBP-Learning” can achieve better accuracy with a more simplified network structure, indicating a promising solution for resource-limited embedded platforms.

4.6 Android implementation with pipeline-based neural processing

In addition to the simulation results, we also implemented the “FPT-Spike” on the Android device applied with the proposed pipeline-based neural processing scheme and tested the battery consumption and time consumption on 300K MNIST samples. Fig. 14 shows the configuration and testing results on our Android implementation. Compared with the android version “Lenet” CNN, our “FPT-Spike” significantly reduces the battery consumption from $\sim$ 13% to $\sim$ 2%. Besides, the time consumption for “FPT-Spike” to process a single image is less than $1ms$, outperforms the “Lenet” CNN while maintaining a similar testing accuracy. These results show that the proposed “FPT-Spike” architecture and pipeline-based neural processing scheme is a flexible solution that can be implemented on different embedded devices (e.g., without the analog and digital mixed signal circuits) with significantly improved performance.

5 Conclusion

As the rate-based spiking neural network (SNN) subjects to power and speed challenges due to processing large number of spikes, in this work, we systematically studied the possibility of utilizing the more power-efficient time-based SNN in real-world cognitive tasks. Three integrated techniques– precise single-spike temporal encoding, efficient supervised temporal learning and fast asymmetric decoding, were proposed to construct the Flexible Precise-Time-Dependent Spiking Neuromorphic Architecture, namely, "FPT-Spike". The single-spike temporal encoding offers an energy-efficient information representation solution with the potentials of design trade-offs by leveraging the flexible spatial–temporal information conversion. The supervised learning and asymmetric decoding can work cooperatively to deliver a more effective and efficient synaptic weight updating and classification. Our evaluations on the MNIST database well demonstrate the advantages of "FPT-Spike" over the rate-based SNN in terms of network size, speed and power, with a comparable accuracy. We hope our study can inspire and motivate more in-depth research on the time-based SNN for realistic applications in energy-constraint platforms.

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.J., et al.: Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(10), 1537–1557 (2015)

Andri, R., Cavigelli, L., Rossi, D., Benini, L.: Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. In: VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on, pp. 236–241. IEEE (2016)

Borst, A., Theunissen, F.E.: Information theory and neural coding. Nat. Neurosci. **2**(11), 947–957 (1999)

Burkitt, A.N.: A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. Biol. Cybern. **95**(1), 1–19 (2006)

Butts, D.A., Weng, C., Jin, J., Yeh, C.I., Lesica, N.A., Alonso, J.M., Stanley, G.B.: Temporal precision in the neural code and the timescales of natural vision. Nature **449**(7158), 92–95 (2007)

Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. Int. J. Comput. Vision **113**(1), 54–66 (2015)

Chu, M., Kim, B., Park, S., Hwang, H., Jeon, M., Lee, B.H., Lee, B.G.: Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron. IEEE Trans. Ind. Electron. **62**(4), 2410–2419 (2015)

Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on, pp. 1135–1139. IEEE (2011)

Corradi, F., Indiveri, G.: A neuromorphic event-based neural recording system for smart brain-machine-interfaces. IEEE Trans. Biomed. Circuits Syst. **9**(5), 699–709 (2015)

Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience **9**, (2015)

Esser, S.K., Merolla, P.A., Arthur, J.V., Cassidy, A.S., Appuswamy, R., Andreopoulos, A., Berg, D.J., McKinstry, J.L., Melano, T., Barch, D.R., et al.: Convolutional networks for fast, energy-efficient neuromorphic computing. Proc. Natl. Acad. Sci p. 201604850 (2016)

Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., Talay, S.: Large-scale fpga-based convolutional networks. Scaling up Machine Learning: Parallel and Distributed Approaches pp. 399–419 (2011)

Farmahini-Farahani, A., Ahn, J.H., Morrow, K., Kim, N.S.: Nda: Near-dram acceleration architecture leveraging commodity dram devices and standard memory modules. In: High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on, pp. 283–295. IEEE (2015)

Gerstner, W.: A framework for spiking neuron models: the spike response model. Handb. Biol. Phys. **4**, 469–516 (2001)

Goodman, D.F., Brette, R.: The brian simulator. Front. Neurosci, **3**, 26 (2009)

Gütig, R., Sompolinsky, H.: The tempotron: a neuron that learns spike timing-based decisions. Nat. Neurosci. **9**(3), 420–428 (2006)

Han, S., Shen, H., Philipose, M., Agarwal, S., Wolman, A., Krishnamurthy, A.: Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, pp. 123–136. ACM (2016)

Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: Nanoscale memristor device as synapse in neuromorphic systems. Nano Lett. **10**(4), 1297–1301 (2010)

Kempter, R., Gerstner, W., Van Hemmen, J.L., Wagner, H.: Temporal coding in the sub-millisecond range: Model of barn owl auditory pathway. In: Advances in neural information processing systems, pp. 124–130 (1996)

Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)

LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits (1998)

Legenstein, R., Naeger, C., Maass, W.: What can a neuron learn with spike-timing-dependent plasticity? Neural Comput. **17**(11), 2337–2382 (2005)

Liu, C., Yang, Q., Yan, B., Yang, J., Du, X., Zhu, W., Jiang, H., Wu, Q., Barnell, M., Li, H.: A memristor crossbar based computing engine optimized for high speed and accuracy. In: VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on, pp. 110–115. IEEE (2016)

Liu, T., Liu, Z., Lin, F., Jin, Y., Quan, G., Wen, W.: Mt-spike: A multilayer time-based spiking neuromorphic architecture with temporal error backpropagation. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 450–457. IEEE (2017)

Maass, W.: On the computational power of winner-take-all. Neural Comput. **12**(11), 2519–2535 (2000)

Neil, D., Liu, S.C.: Minitaur, an event-driven fpga-based spiking network accelerator. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(12), 2621–2628 (2014)

Ponulak, F.: Resume-new supervised learning method for spiking neural networks. Institute of Control and Information Engineering, Poznan University of Technology.(Available online at: http://d1.cie.put.poznan.pl/fp/research.html) (2005)

Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep, DTIC Document (1985)

Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling **5**(3), 1 (1988)

Seo, J.s., Brezzo, B., Liu, Y., Parker, B.D., Esser, S.K., Montoye, R.K., Rajendran, B., Tierno, J.A., Chang, L., Modha, D.S., et al.: A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In: Custom Integrated Circuits Conference (CICC), 2011 IEEE, pp. 1–4. IEEE (2011)

Sjöström, J., Gerstner, W.: Spike-timing dependent plasticity. Spike-timing dependent plasticity p. 35 (2010)

Szegedy, C.: An overview of deep learning. AITP **2016**, (2016)

Thorpe, S., Delorme, A., Van Rullen, R.: Spike-based strategies for rapid processing. Neural Netw. **14**(6), 715–725 (2001)

Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on cpus. In: Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop, vol. 1, p. 4. Citeseer (2011)

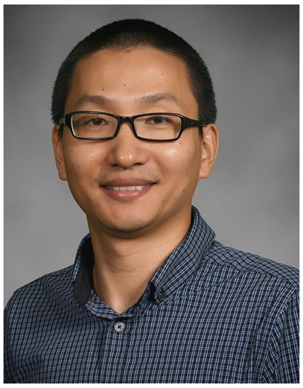
Wang, Y., Tang, T., Xia, L., Li, B., Gu, P., Yang, H., Li, H., Xie, Y.: Energy efficient rram spiking neural network for real time classification. In: Proceedings of the 25th GLVLSI, pp. 189–194. ACM (2015)

Yu, Q., Tang, H., Tan, K.C., Li, H.: Precise-spike-driven synaptic plasticity: learning hetero-association of spatiotemporal spike patterns. PLoS One **8**(11), e78318 (2013)

Zhao, C., Wysocki, B.T., Thiem, C.D., McDonald, N.R., Li, J., Liu, L., Yi, Y.: Energy efficient spiking temporal encoder design for neuromorphic computing systems. IEEE Trans. Multi-Scale Comput. Syst. **2**(4), 265–276 (2016)

Dr. Tao Liu received his Ph.D. in Electrical and Computer Engineering and M.S. in Computer Engineering from Florida International University in 2020 and 2013, respectively, and his B.S. in Computer Science from Southeast University, Nanjing, China in 2007. He is currently a research associate in the Department of Electrical and Computer Engineering at Lehigh University. His research interests include neuromorphic computing, embedded machine learning system design, and machine learning security. His works have been published on conferences and journals across the fields of EDA, architecture, computer systems, machine learning, and artificial intelligence, etc., including DAC, ICCAD, CVPR, HOST, MICCAI, TCAD, JTEC, etc. Dr. Liu received the best paper nominations from DAC 2020, ICCAD 2018, ASP-DAC 2018. He was also the recipient of the 54th DAC A. Richard Newton Young Student Fellow Award, the 36th ICCAD ACM Student Research Competition Travel Award.

Dr. Gang Quan received his Ph.D. from the Department of Computer Science & Engineering, University of Notre Dame, USA, his M.S. from the Chinese Academy of Sciences, Beijing, China, and his B.S. from the Department of Electronic Engineering, Tsinghua University, Beijing, China. He is currently a full professor in the Electrical and Computer Engineering Department, Florida International University. His research interests and expertise include real-time systems, embedded system design, power-/thermal-aware computing, advanced computer architecture and cloud computing. Dr. Quan is the recipient of a National Science Foundation Faculty Career Award. He also won the Best Paper Award from the 38th Design Automation Conference. His paper was also selected as one of the Most Influential Papers of 10 years Design, Automation, and Test in Europe Conference (DATE) in 2007. Dr. Quan is a senior member of IEEE.



Wujie Wen is currently an assistant professor in the department of Electrical and Computer Engineering at Lehigh University. He received his Ph.D. from University of Pittsburgh in 2015. He earned his B.S. and M.S. degrees in electronic engineering from Beijing Jiaotong University and Tsinghua University, Beijing, China, in 2006 and 2010, respectively. He was an assistant professor in the ECE department of Florida International University, Miami, FL, during 2015–2019. Before joining the academy, he also worked with AMD and Broadcom for various engineer and intern positions. His research interests include deep learning hardware acceleration/security/application, neuromorphic computing, electronic design automation (EDA), and circuit-architecture design for emerging memory technologies etc. His works have been published widely across venues in EDA, machine learning/AI etc., including HPCA, DAC, ICCAD, DATE, ICPP, HOST, CVPR, ECCV, AAAI etc. Dr. Wen is the associate editor of Neurocomputing and serves/served as the General Chair of ISVLSI 2019 (Miami), Technical Program Chair of ISVLSI 2018 (Hong Kong), as well as program committee for many conferences such as DAC, ICCAD, DATE, ASP-DAC etc. He received best paper nominations from DAC2020, ICCAD2018, ASP-DAC2018, DATE2016 and DAC2014. He was also the recipient of the 49th DAC A. Richard Newton Graduate Scholarship—the most prestigious Ph.D. scholarship (one awardee per year) in EDA society. His research projects are sponsored by the US National Science Foundation, Air Force Research Lab and the Florida Center for Cybersecurity etc.