ONCHIP TRAINING OF SPIKING NEURAL ACCELERATORS USING SPIKE-TRAIN LEVEL DIRECT FEEDBACK ALIGNMENT

A Thesis

by

RENQIAN ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2019

Major Subject: Computer Engineering

ABSTRACT

Spiking Neural Networks (SNNs) are widely researched in recent years and present a promising computing model. Several key properties including biologically plausible information processing and event driven sample learning make SNNs be able for ultra-low power neuromorphic hardware implementation. However, to achieve the same level of performance in training conventional deep artificial neural networks (ANNs), especially for networks with error backpropagation (BP) algorithm, is a significant challenge existing in SNNs training, which is due to inherent complex dynamics and non-differentiable spike activities of spiking neurons. To solve this problem, this thesis proposes the first study on realizing competitive spike-train level backpropagation (BP) like algorithms to enable on-chip BP training of SNNs. This novel alrogithm, called spike-train level direct feedback alignment (ST-DFA), performs better in computation complexity and training latency compared to traditional BP methods. Furthermore, algorithm and hardware co-optimization as well as efficient online neural signal computation are explored for on-chip implementation of ST-DFA. To figure out the performance of this proposed algorithm, the final online version of ST-DFA is tested on the Xilinx ZC706 FPGA board. During testing on real-world speech and image classification applications, it shows excellent performance vs. overhead tradeoffs. SNN neural processors with on-chip ST-DFA training show competitive classification accuracy of 97.23% for the MNIST dataset with 4X input resolution reduction and 87.40% for the challenging 16-speaker TI46 speech corpus, respectively. This experimental result is then compared to the hardware implementation of the state-of-the-art BP algorithm HM2-BP. While trading off classification performance very gracefully, the design of the proposed online ST-DFA training reduces functional resources by 76.7% and backward training latency by 31.6%, which dramatically cut resource and power demand for hardware implementation.

DEDICATION

To my families, my professors, my friends.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Peng Li, and my committee members, Dr. Weiping Shi, and Dr. Shuiwang Ji, for their guidance and support throughout the course of this research.

Thanks to my research mates and guiders Mr. Jeongjun Lee, Miss. Yu Liu, Mr. Wenrui Zhang and Mr. Wang Yu for sharping my mind and guiding me with so much patience.

Thanks also go to my group members, my friends and the department faculty and staff for making my time at Texas A & M University a great experience.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis (or) dissertation committee chaired by Professor Peng Li of the Department of Electrical Computer Engineering. This work is a collaborator work. Parts of the work were done by Jeongjun Lee and Wenrui Zhang in the Electrical Computer Engineering Department. Section 3.2 and section 5.1 is partially done by Mr. Wenrui Zhang. Section 4.2 and section 5.2 is partially done by Mr. Jeongjun Lee. All other work conducted for the thesis (or) dissertation was completed by Renqian Zhang independently.

Funding Sources

This dissertation is based upon work supported by the National Science Foundation (NSF) under Grant No.CCF-1639995 and the Semiconductor Research Corporation (SRC) under Task 2692.001.

NOMENCLATURE

OGAPS	Office of Graduate and Professional Studies at Texas A&M University
B/CS	Bryan and College Station
TAMU	Texas A&M University
SDCC	San Diego Comic-Con
EVIL	Every Villain is Lemons
EPCC	Educator Preparation and Certification Center at Texas A&M University - San Antonio
FFT	Fast Fourier Transform
ARIMA	Autoregressive Integrated Moving Average
SSD	Solid State Drive
HDD	Hard Disk Drive
O&M	Eller Oceanography and Meteorology Building
DOS	Disk Operating System
HDMI	High Definition Multimedia Interface
L^1	Space of absolutely Lebesgue integrable functions; i.e., $\int \lvert f\rvert < \infty$
L^2	Space of square-Lebesgue-integrable functions, i.e., $\int \lvert f\rvert^2 < \infty$
$PC(S)$	Space of piecewise-continuous functions on S
GNU	GNU is Not Unix
GUI	Graphical User Interface

PID	Principal Integral Domain
MIP	Mixed Integer Program
LP	Linear Program

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Spiking Neural Networks

Spiking Neural Networks (SNNs) are brain-inspired models which have gathered significant research interests in recent years [1, 2]. The updates of SNNs are spike-based and event driven. Furthermore, the spike-based communication between spiking neurons is one-or-nothing. A spike generates a trace of synaptic current to the post-synaptic target neurons. When the integrated membrane potential of one target neuron exceeds certain threshold, it will emit another spike for further transmission of information and reset to reaccumulate input current. For a particular neuron, weights of different input synapses can be introduced to represent and adjust strength of vairous information paths. With the training process, these connection weights are finely determined to execute tasks like classification. These behaviors, especially combined with spike-based inputs, support temporal coding schemes and energy efficient VLSI neuromorphic hardware implementations. Well-known implementation of SNNs includes IBM's TrueNorth [3] and Intel's Loihi [4]. With all these recent progresses in SNNs and neuromorphic processor designs, compared with traditional artificial neural networks (ANNs) [5], SNNs still need to be improved for greater performance. Despite of having been shown to become as computationally powerful as ANNs in theory [6], SNNs have not been able to practically achieve the state-of-the-art performance of ANNs for real-world applications. Among the challenges of optimizing classification results of SNNs, the achievable performance and computational complexity take main positions.

1.2 Error Backpropagation

Error backpropagation (BP) and its variants such as stochastic gradient decent [7] is a strong technique used in traditional ANNs to improve performance. According to the definition of BP, the weights of connections are initialed randomly. Then with certain period of training, the connection weights of a network can be adjusted to minimize the loss function (for example mean square error (MSE)) between the expected and actual outputs of network through gradient descent. When the loss function satisfies requirement settings, the connection weights are determined and the training process of this network finishes. Inspired by this technique, various methods have tried to implement BP on SNNs to attain the same level of classification result [8, 6, 9, 10]. The major challenges in BP training of SNNs stem from the fact that spike signals are not differentiable. This property, along with temporal dynamics, prevents straightforward derivative computation of BP. SpikeProp [8] is known as the first attempt to implement BP algorithm on SNNs. However, the network structure and application range of this method are quite limited. Only single-spike training is accepted and the learning functions are also quite simple, like XOR. [6] proposes a BP algorithm which differentiates neuron's membrane potential instead of discrete output spikes. This method adds low-pass filtered spiking signals onto the membrane potential, ignoring sudden variation of membrane potential during back propagating to generate differentiable activation functions which is critical to error backpropagation. This idea is analogous to the non-linear activation function in traditional neural networks. Though the work shows competitive learning results, it lacks explicit consideration of temporal correlations of neural activities. [9] improves [6] by capturing temporal effects with backpropapogation through time (BPTT) [11]. However, the error gradient is still computed by differentiating the membrane potential, leading to inconsistency w.r.t the rate-coded loss function.

The inconsistency problems of applying BP on SNNs are solved by a more recently published work [10]. In this paper, Jin proposes a hybrid macro/micro level backpropagation (HM2-BP) algorithm to train deep SNNs (which contain multiple layers). HM2-BP finds a new approach to capture temporal behavior and directly computes the gradient of the rate-coded loss functions to prevent inconsistency, so that it outperforms all other existing BP algorithms based on the leaky integrated-and-fire neuron model. This method derives the gradient by decomposing each derivative into two components, backpropagation over firing rates (macro-level) and over spike trains (micro-level). At microscopic level for each pre/post-synaptic neural connection, this method precisely computes the spike-train level post-synaptic potential (S-PSP) to gather temporal contribution on certain spike times [10]. Then after all inputs are passed through the network, it aggregates S-PSPs of all connections of one neuron to define a rate-based loss and back-propagates the error of this loss. By processing in this way, HM2-BP can evaluate the direct impact of weight changes on the rate-coded loss function and further adjust the number of spikes generated by each neuron.

1.3 Spike-Train Level Direct Feedback Alignment

Though HM2-BP can result in a state-of-the art performance in classification, it still has several limitations. One is the property that the error signal is transferred layer by layer through weights symmetric to the feed-forward weights in backpropagation, which is not biologically plausible. Also the high complexity of this method and complex layer-by-layer backward computation constrain it to achieve lower training latency and be implemented on a wider range of platforms, especially platforms with limited computation resources. For instance, while HM2-BP improves the scalability of BPTT [9] by operating on the spike-train level, i.e. application of BP does not discretize time, it still involves complex computations and its latency in the backward phase is proportional to network depth.

To solve the first problem, a recent discovered direct feedback alignment (DFA) can be introduced [12]. The concept of feedback alignment is that the error back-propagating weights may not need to be symmetric to the feed-forward weights to gain a good training performance. Instead, a randomly generated weight matrix can be used and can stay unchanged since the networks can learn how to make feedback useful during training. With DFA applied, the error is more biologically-plausibly fed back to each hidden layer through fixed random feedback connections directly from the output layer, reducing a bulk of the BP complexity. This DFA technique is first utilized in deep neural networks. But SNNs can also benefit from this property. Furthermore, DFA can be performed for all hidden layers concurrently, reducing the backward phase latency, especially for deeper SNNs. To solve the second limitation of BP mentioned in the previous section, this work finds a sidestepping backpropagation approach to reduce the computation cost and then combines this approach with DFA algorithm to achieve a hardware implementation. The derived approach is called spike-train level direct feedback alignment (ST-DFA).

This work aims to answer the following questions: 1) Can biologically plausible mechanisms developed to sidestep complex BP algorithms while delivering competitive performance? 2) Can such mechanisms be leveraged for efficient on-chip training of multi-layer SNNs?

The main contributions of this work are:

- Demonstrate the *first* direct feedback alignment algorithm for training multi-layer SNNs by extending the DFA concept developed for conventional ANNs;

- The spiking DFA algorithm is embodied at the spike-train level to further improve scalability by avoiding involved error feedback over time;

- Perform algorithm-hardware co-optimization and demonstrate the *first* hardware realization of DFA for SNNs with significantly reduced hardware overhead, energy dissipation, and latency while achieving competitive performances for image/speech recognition tasks.

The proposed ST-DFA is optimized and implemented on the Xilinx ZC706 FPGA board. Experimental result shows excellent cost-effectiveness for on-chip SNN training and decent classification performance compared to state-of-the-art algorithms. Hardware SNNs with ST-DFA deliver competitive accuracy of 97.23% for the MNIST [13] with 4X input resolution reduction and 87.40% for the challenging 16-speaker TI46 [14] speech corpus, respectively. Compared to the hardware implementation of the state-of-the-art BP algorithm HM2-BP, the design of the proposed ST-DFA reduces functional resources by 76.7% and backward training latency by 31.6% while gracefully trading off classification performance.

The rest of this thesis is organized as follows:

Section 2 shows some background of Direct Feedback Alignment (DFA) and Spike-train level Post Synaptic Potential (S-PSP). Section 3 mainly introduces the proposed ST-DFA method, the derivation and some simplification of it. Section 4 is about high level architecture and detailed structure of critical modules of ST-DFA hardware implementation and optimization. Section 5 illustrates the experimental results on classification performance, power consumption and time cost. Section 6 is the conclusion of this thesis and outlook for future work.

I have close co-operations with my group mate Mr. Jeongjun Lee to finish this work. Also Mr. Wenrui Zhang and Ms. Yu Liu give me valuable guidance in algorithm section and writing skills. In this work my contribution includes the feed-forward part design of ST-DFA implementation and the high level finite state machine (FSM) part for control logic of the design. Also, I tested the power efficiency and classification performance on

MNIST dataset [13], finished some debugging and performed further optimization based on these results.

2. BACKGROUND

2.1 Direct Feedback Alignment

Just as mentioned in the last section, backpropagation (BP) has been widely utilized in training process of neural networks. The basic process of BP can be concluded as computing a global error at the output layer and then propagate this error signal layer by layer through all previous layers until reaching the input layer. During the "propagating" process, each layer finds its error by multiplying incoming error from the previous layer with a weight matrix that is completely symmetric to the one for the feed-forward connections. Actually a recent work called Feedback Alignment (FA) [15] has demonstrated that this fact is not biologically plausible. In fact there is no causal relationship between a good performance and keeping the weight used for layer by layer error transferring being symmetric to the weights used for forward propagation. Since the neural network can learn how to make feedback useful when training performs, the feedback weight matrix can be generated randomly and stay unchanged during training process. [16] applies FA for training SNNs.

Based on the result of this work, [12] introduced Directed Feedback Alignment (DFA), which abandons the backpropagation way of error used by traditional BP algorithms. In DFA, random and fixed feedback connections are generated directly between each layer and the output layer, rather than between current layer and the previous layer in BP algorithm. This property makes DFA more biologically friendly since in this method errors can be treated completely locally and layers more closely to spikes input layer side do not need to wait long backpropagation time to receive error inputs. The elimination of symmetric weight also improves the biologically plausibility of DFA. [12] shows that for conventional multi-layer ANNs, for example DNN, DFA will not result in any significant performance drop, compared to state-of-the-art BP method.

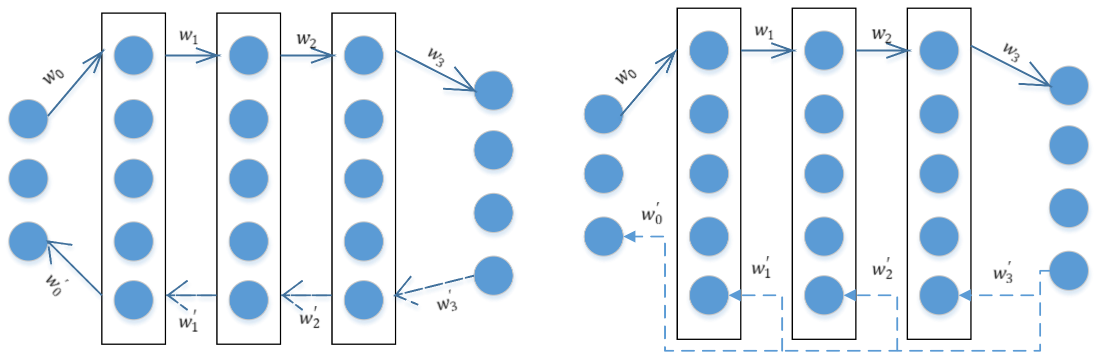

Figure 2.1: (a) Backpropagation (BP) vs. (b) direct feedback alignment (DFA). Solid arrows indicate feedforward paths and dashed arrows indicate feedback paths. The feedback matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ need not be symmetric to $\boldsymbol{W}_2$ or $\boldsymbol{W}_3$.

In this thesis, the idea of DFA is borrowed and extended from traditional ANNs into SNNs. To the best of writer's knowledge, this is the first work applying DFA to SNNs. Furthermore the proposed DFA method, as known as ST-DFA, operates at spike-train level, so that the calculation and implementation of the DFA can be much more efficient and cost less than tradition backpropagation rule. A more detailed demonstration will be shown in next sections.

2.1.1 Spike-train Level Post-synaptic Potential

Before describing the proposed ST-DFA in Section 3, the concept of Spike-train Level Post-synaptic Potential (S-PSP) behind the spike-train level computation of ST-DFA should be introduced first since it is a critical internal variable in calculation.

The widely adopted leaky integrate-and-fire (LIF) model for spiking neurons is given by [17]:

$$\tau_m \frac{u_i(t)}{dt} = -u_i(t) + R\,\alpha_i(t), \tag{2.1}$$

with

$$\tau_s \frac{\alpha_i(t)}{dt} = -\alpha_i(t) + \sum_j w_{ij} \sum_{t_j^{(f)}} D\left(t - t_j^{(f)}\right), \tag{2.2}$$

Where $u_i(t)$ is the membrane potential of the neuron i, $\alpha_i(t)$ is the first order synaptic model where τ_s is the time constant used in this model to control potential accumulation, and τ_m is the time constant of membrane potential with value $\tau_m = RC$. R and C are the effective leaky resistance and effective membrane capacitance. The weight of the synapse from the pre-synaptic neuron j to current neuron i is represented by w_{ij}. $t_j^{(f)}$ denotes a particular firing time of the neuron j. $D(t)$ is the Dirac delta function. R is set to 1 since it can be absorbed into synaptic weights.

Integrating (2.1) and (2.2) gives the spike response model (SRM) [10]:

$$u_i(t) = \sum_j w_{ij} \sum_{t_j^{(f)}} \epsilon\left(t - \hat{t}_i^{(f)}, t - t_j^{(f)}\right), \tag{2.3}$$

where $\hat{t}_i^{(f)}$ denotes the last firing time of the neuron i. $\epsilon(s,t)$ specifies the normalized time course of the *post-synaptic potential* evoked by a single firing spike of the pre-synaptic neuron:

$$\epsilon(s,t) = \frac{1}{C}\int_0^s \exp\left(-\frac{t'}{\tau_m}\right)\, \alpha_i\left(t - t'\right)\, \mathrm{d}t'. \tag{2.4}$$

Integrating (2.4) gives:

$$\epsilon(s,t) = \frac{e^{(-\max(t-s,0)/\tau_s)}}{1 - \frac{\tau_s}{\tau_m}} \left[e^{\left(-\frac{\min(s,t)}{\tau_m}\right)} - e^{\left(-\frac{\min(s,t)}{\tau_s}\right)}\right] H(s)H(t), \tag{2.5}$$

where $H(t)$ is the Heaviside step function.

The (normalized) **spike-train level post-synaptic potential** (**S-PSP**) $e_{i|j}$ is defined by summing all the (normalized) post-synaptic potential of the neuron i, while the sample time is chosen as right before all the neuron i's firing times evoked by the spike train of the pre-synaptic neuron j. This is given by

$$e_{i|j} = \sum_{t_i^{(f)}} \sum_{t_j^{(f)}} \epsilon(t_i^{(f)} - \hat{t}_i^{(f)}, t_i^{(f)} - t_j^{(f)}). \tag{2.6}$$

S-PSP accumulates the spike train of the pre-synaptic neuron j and calculate the effect of this spike train on the membrane potential of the post-synaptic neuron i. This provides the basis to build a connection between firing counts to spike events.

Summing the weighted S-PSPs from all pre-synaptic neurons of the neuron i gives the **total post-synaptic potential** (**T-PSP**) a_i, which is directly correlated to the neuron i's firing count o_i through the firing threshold voltage ν:

$$a_i = \sum_j w_{ij}\, e_{i|j}. \qquad o_i = g(a_i) \approx \frac{a_i}{\nu} \tag{2.7}$$

3. PROPOSED SPIKE-TRAIN LEVEL DIRECT FEEDBACK ALIGNMENT (ST-DFA)

3.1 Proposed ST-DFA Algorithm

In conventional (non-spiking) ANNs like DNNs, the error for one training sample is the squared error and can be defined at the output layer by:

$$E = \frac{1}{2}||o - y||_2^2, \tag{3.1}$$

where y and o are vectors specifying the desired output (label) and the actual output, respectively. The output o_i of each neuron i is determined by the activation function ϕ_i: $o_i = \phi_i(\sum_j w_{ij}x_j)$, where x_j is the input value from the pre-synaptic neuron j and w_{ij} is the synaptic weight between neuron j and neuron i.

The well-known BP algorithm for ANNs [18], which is ubiquitously used in deep learning, is:

$$\begin{aligned} \Delta w_{ij} &= \eta \frac{\partial E}{\partial w_{ij}^k} = \eta \delta_i^k \phi_j^{k-1} \\ \delta_i^k &= \begin{cases} o_i - y_i & \text{for output layer,} \\ \phi_i'^{k+1} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{li}^{k+1} & \text{for hidden layers,} \end{cases} \end{aligned} \tag{3.2}$$

where η is the learning rate, δ_i^k the error for the i_{th} neuron of the k_{th} layer, r^k the number of neurons in the k_{th} layer.

Recently several works [6, 9, 10] have demonstrated that to produce highly competitive performance, training SNNs using BP with respect to a rate-coded loss function can be treated as a good method. These rate-coded loss functions are also adopted for the proposed ST-DFA. As mentioned in last section, different from BP, the proposed ST-DFA algorithm for SNNs computes each error δ by direct feedback from the output layer on the spike-train level rather than by error from connected previous layer, giving to the following update rule:

$$
\begin{aligned}
\Delta w_{ij} &= \eta \frac{\partial E}{\partial w_{ij}} = \eta \delta_i^k e_{i|j}^k, \\
\delta_i^k &= \begin{cases} \frac{o_i^o - y_i^o}{\nu} & \text{for output layer,} \\ \sum_{l=1}^{r^o} \delta_l^o b_{li}^k & \text{for hidden layers,} \end{cases}
\end{aligned}
\tag{3.3}
$$

where η is the learning rate, δ_i^k the error of the neuron i in the k_{th} hidden layer, $e_{i|j}^k$ the S-PSP from the neuron i to neuron j, o_i^o the actual firing count of neuron i in the output layer, y_i^o the desired firing count for the neuron i, ν the firing threshold, r^o the number of neurons in the output layer, δ_l^o the error of the neuron l in the output layer, and b_{li}^k the value of the fixed random feedback.

The last equation of (3.3) is based on the concept of DFA. As shown in figure, with ST-DFA, each hidden layer builds a direct connection with the output layer where the measurement is a different matrix which is called the random feedback matrix B. The weights (values) in these matrices are randomly generated and then stay fixed during training. The error vector δ^k of the hidden layer k is directly obtained from the error vector of the output layer δ^o and the random feedback matrix B^k as: $\delta^k = B^k \times \delta^o$. The detailed derivation of ST-DFA is introduced next.

3.2 Derivation of ST-DFA

Similar to (3.1) and using (2.7), we define the rate-coded loss function as:

$$E = \frac{1}{2}||o - y||_2^2 = \frac{1}{2}||\frac{a}{\nu} - y||_2^2, \tag{3.4}$$

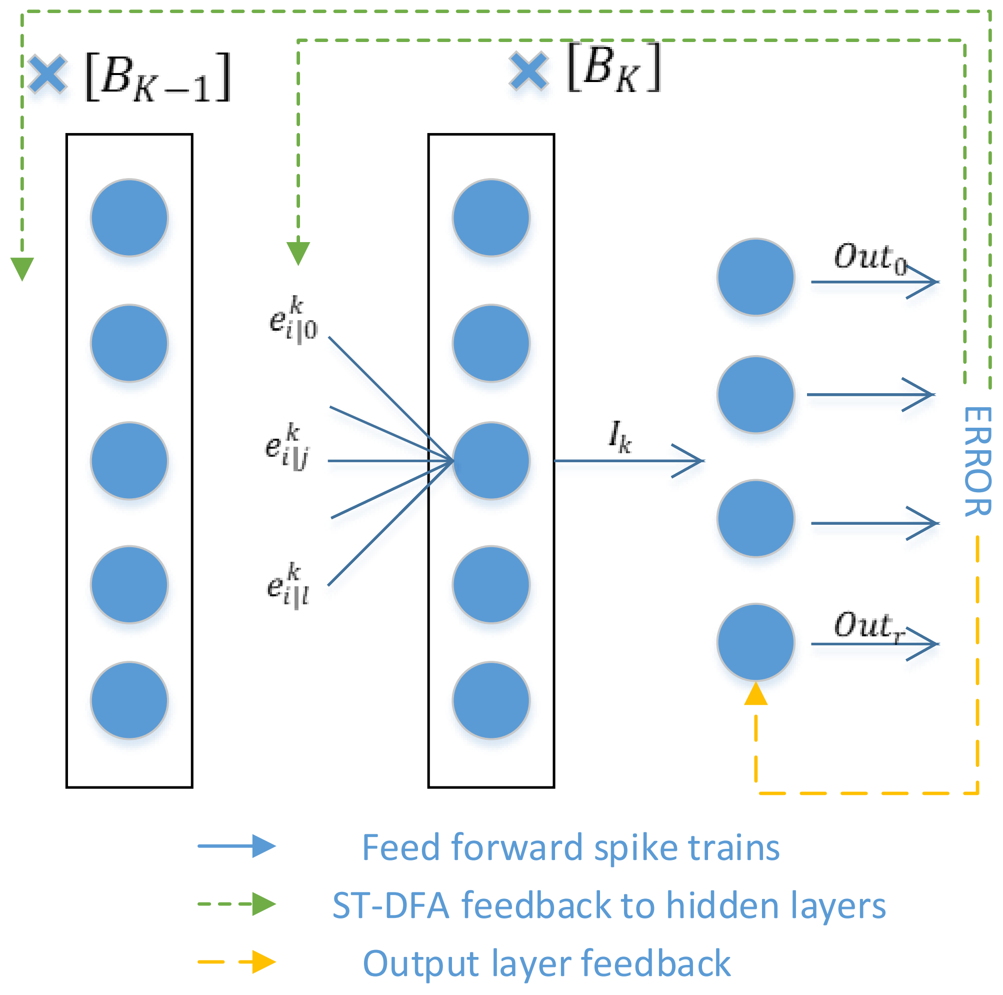

Figure 3.1: The proposed spike-train level DFA (ST-DFA).

where y, o and a are vectors specifying the desired firing counts (label), the actual firing counts, and the T-PSP of the output neurons, respectively. Differentiating the loss function with respect to each trainable weight w_{ij} leads to:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_i^k}\frac{\partial a_i^k}{\partial w_{ij}} = \delta_i^k \frac{\partial a_i^k}{\partial w_{ij}}, \tag{3.5}$$

where a_i^k is the T-PSP of the neuron i in the k_{th} layer.

At the microscopic level, in each pre/post-synaptic spike train pair, S-PSP is precisely computed to account for the temporal contribution of the given pre-synaptic spike train to the firings of the post-synaptic neuron based on exact spike times. At the macroscopic level, the rate-based errors in the output layer is backpropagated by aggregating the effects of spike trains on each neuron's firing count via the use of S-PSPs. This is a practical way of linking spiking events to firing rates. To assist backpropagation, a decoupled model of the S-PSP for disentangling the effects of firing rates and spike-train timings is proposed to allow differentiation of the S-PSP w.r.t. pre and post-synaptic firing rates at the micro-level. As a result, HM2-BP approach is able to train synaptic weights at the spike-train level. In contrast to other methods, this hybrid approach can directly compute the gradient of the rate-coded loss function with respect to tunable parameters.

With explaination above, it is instrumental to note that each S-PSP $e_{i|j}$ depends on both rate and temporal information of the pre/post spike train pair, i.e. $e_{i|j}$ depends on the pre/post-synaptic firing counts o_i and o_j and pre/post-synaptic firing times $\mathbf{t}_j^{(f)}$ and $\mathbf{t}_i^{(f)}$:

$$e_{i|j} = f(o_j, o_i, \mathbf{t}_j^{(f)}, \mathbf{t}_i^{(f)}). \tag{3.6}$$

For the i_{th} output neuron, δ_i^o can be obtained from (3.5) and (2.7):

$$\delta_i^o = \frac{\partial E}{\partial a_i^o} = (o_i - y_i)\,\frac{\partial o_i}{\partial a_i} = \frac{o_i - y_i}{\nu}. \tag{3.7}$$

For each i_{th} neuron in the hidden layer k, δ_i^k is derived from the chain rule based on (2.7):

$$\begin{aligned}\delta_i^k = \frac{\partial E}{\partial a_i^k} &= \sum_{l=1}^{r^{k+1}} \frac{\partial E}{\partial a_l^{k+1}} \frac{\partial a_l^{k+1}}{\partial a_i^k} = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} \frac{\partial a_l^{k+1}}{\partial a_i^k} \\ &= \sum_{l=1}^{r^{k+1}} \delta_l^{k+1}\, w_{li}^{k+1} \frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}.\end{aligned} \tag{3.8}$$

The first key development in ST-DFA is that the way in which the error δ_i^k is calculated in each hidden layer changes from $\sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{li}^{k+1} \frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}$ to $\sum_{l=1}^{r^o} \delta_l^o d_{li}^k \frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}$, where d_{li}^k is the direct feedback alignment from the output neuron l to the hidden layer neuron i. d_{li}^k is a randomized and fixed value. In this process, the w_{li}^{k+1} is replaced from $(k+1)_{th}$ layer to k_{th} layer in (3.8) by d_{li}^k, leading to:

$$\delta_i^k = \delta_l^o d_{li}^k \frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}. \tag{3.9}$$

As such, the error δ^k of each hidden neuron is directly determined by the output layer error vector δ^o rather than by the error vector δ^{k+1} of the connected next layer.

Moreover is the following key observation. In (3.9), since d_{li}^k is randomly generated, $\frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}$ can be absorbed into d_{li}^k to further simplify ST-DFA. Denote the new DFA parameter absorbing $\frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}$ by $b_{li}^k = d_{li}^k \frac{\partial e_{l|i}^{k+1}}{\partial a_i^k}$, the simplified error computation becomes:

$$\delta_i^k = \begin{cases} \frac{o_i^o - y_i^o}{\nu} & \text{for output layer,} \\ \sum_{l=1}^{r^o} \delta_l^o b_{li}^k & \text{for hidden layers,} \end{cases} \tag{3.10}$$

where b_{li}^k is one entry of the random feedback matrix B in Fig. 3.1.

Thus, ST-DFA reduces the computational complexity by not only avoiding layer-by-layer propagation but also the additional simplification via the using of b_{li}^k.

3.3 Simplification for Hardware Friendliness

The last term on the right-hand side of (3.5) differentiates the total post-synaptic potential (T-PSP) a_i^k. Considering (2.7), it can be written as:

$$\begin{aligned}\frac{\partial a_i^k}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left(\sum_{j=1}^{r^{k-1}} w_{ij}\, e_{i|j}^k \right) = e_{i|j}^k + \sum_{l=1}^{r^{k-1}} w_{il} \frac{\partial e_{i|l}^k}{\partial o_i^k} \frac{\partial o_i^k}{\partial w_{ij}} \\ &= e_{i|j}^k + \frac{e_{i|j}^k}{\nu} \sum_{l=1}^{r^{k-1}} w_{il} \frac{\partial e_{i|l}^k}{\partial o_i^k}.\end{aligned} \tag{3.11}$$

The exact evaluation of the above expression requires multiple additions, multiplications, and divisions, introducing high hardware overhead and additional latency for hardware implementation.

The first term $e_{i|j}^k$ on the right-hand side of (3.11) can be interpreted as the direct influence exerted on the T-PSP a_i^k by changing the synaptic weight w_{ij} as seen from (2.7). The second term $\frac{e_{i|j}^k}{\nu} \sum_{l=1}^{r^{k-1}} w_{il} \frac{\partial e_{i|l}^k}{\partial o_i^k}$ comes from the fact that changing the weight w_{ij} leads to variation in the post-synaptic spike train. Thus, the S-PSP $e_{i|l}^k$ to the neuron i also varies as it depends on the firing times of the post-synaptic neuron. Nevertheless, it has been observed that the first term dominates the second term. By dropping the second term, the final hardware-friendly ST-DFA algorithm can be reached of (3.3). With the deduction process shown before, it is convincing that this algorithm can maintain good performance.

In comparison, the spike-train level BP algorithm HM2-BP is [10]:

$$\begin{aligned} \Delta w_{ij} &= \eta \delta_i^k e_{i|j}^k \left(1 + \frac{1}{\nu} \sum_{l=1}^{r^{k-1}} w_{il} \frac{\partial e_{i|l}^k}{\partial o_i^k}\right), \\ \delta_i^k &= \begin{cases} \frac{o_i^k - y_i^k}{\nu} & \text{for output layer,} \\ \frac{1}{\nu} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{li} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k} & \text{for hidden layers.} \end{cases} \end{aligned} \tag{3.12}$$

While HM2-BP delivers the state-of-the-art performance, it would be very costly to implement on hardware if ever feasible.

In all, compared to HM2-BP in (3.12), ST-DFA in (3.3) is much more hardware friendly. With ST-DFA, direct error feedback to each hidden layer is accomplished without layer-by-layer back propagation while HM2-BP requires high-resolution multiplications with the transpose of the forward weights and other expensive operations layer by layer. Next section will introduce the way to efficiently realize the ST-DFA algorithm on digital hardware.

4. SNN ACCELERATORS WITH ST-DFA ON-CHIP TRAINING

4.1 Architecture

Using the proposed ST-DFA on-chip training algorithm, the architecture of the proposed multi-layer spiking neural processors is shown in Fig. 4.1. For illustration purpose, only two hidden layers are introduced. Architecturally, the processor is comprised of an input spike buffer feeding multiple hidden layers composed of hidden neuron elements (HEs). The last hidden layer connects to the output layer which consists of a set of output neuron elements (OEs). The function of same type of neuron elements is identical. Therefore, a modular design approach can be used to implement each spiking neuron with the form of HE or OE. As such, given an arbitrary network depth and width, a proper number of HEs and OEs can be instantiated to this particular multi-layer SNNs.

Both inference and training are supported. Training over an input example splits into two phases: forward passing and backward passing. To compute the S-PSPs, which is an critical internal variable used in ST-DFA training, an online manner is used in the forward passing phase of training. The remaining computations of the forward passing are identical to those performed in inference. To support ST-DFA training, the error generator utilizes an array of subtractors to compute the difference between the actual OE output spike counts with expected ones (label). At each hidden layer, this output-layer error vector is multiplied with the associated ST-DFA random feedback matrix inside each layer to allow weight updates performed by each neuron.

4.1.1 On-chip Training

For each training sample, a global controller (FSM) controls the behaviors of the forward and backward parts of training. At a particular layer, all neurons will process input

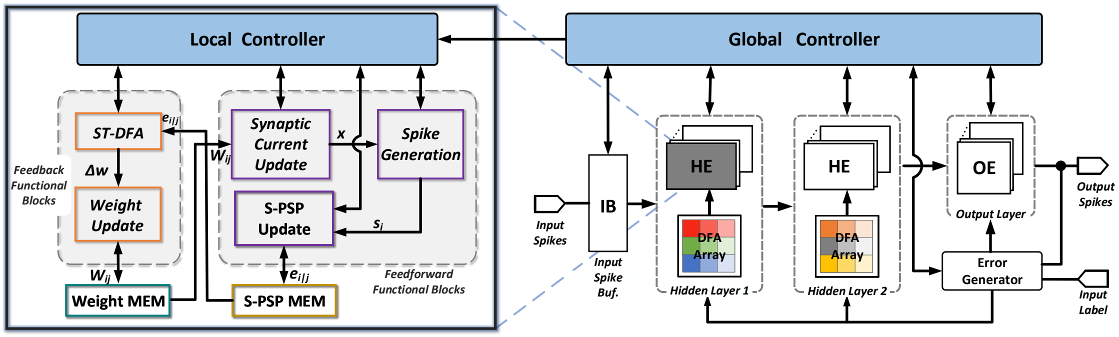

Figure 4.1: Proposed architecture of multi-layer SNNs with onchip ST-DFA training. HE represents a digital hidden neuron element; and OE represents a digital output neuron element.

information in parallel so that the inherent parallelism of the hardware SNN processor architecture can be fully explored. In the forward passing phase, starting from the hidden layer connected to inputs till the output layer, only one layer is activated by the global controller at certain biological time step. After output spikes are generated at current time step, the global controller will inform the connected post layer and push the training forward to the next time step. The processor will repeat this process as long as there are training samples not learned by the networks. After all samples processed, the global controller will shut down the function of feed forward passing phase to reduce power consumption and start the backward passing phase. In backward part, the first step is to calculate the output error δ_l^o in (3.3) based on a pre-set expected output spike counts (label) and accumulated output spikes vector stored in an array of registers. After that, all hidden layers start to perform ST-DFA for weight updating at the same time. The weight updating time for one neuron is related to the number of input synapses and the input connection density of neurons in different layers is not the same. Thus, the weight update process of all hidden layers may not end at one time. Again this is the global controller's job to collect end information of different weight updating process and find out the latest among all layers. After detecting that all hidden layers finishing ST-DFA weight updates, the networks will move onto next training data.

4.1.2 Neuron Unit Design

Each HE or OE contains several functional blocks which can be categorized into feed-forward functional blocks and feedback functional blocks as shown in Fig. 4.1. Except the ST-DFA learning module, OEs are identical to HEs. This is because that the error δ_i^k defined for output neurons is computed by the Error Generator module, as mentioned in last paragraph. In each neuron unit, two memory modules are used to store the synaptic weights and all its spike-train level post-synaptic potentials (S-PSPs), respectively. The weight memory is implemented with block RAM (BRAM) and a 2-D array of flip flops (FFs) is utilized to build the S-PSP memory on the FPGA. A neuron-level local controller (FSM) controls the detailed inference/training steps, which shown in Fig. 4.2. When receiving feed forward enable signal from global controller, the local FSM starts the process of updating input current to each synapse from previous layer. After this stage the local FSM changes into membrane potential calculating state to update potential of current neuron. With membrane potential updated, the neuron's firing activity in current biological time step is decided. The local controller also communicates with the global controller for synchronizing processes between different layers and inference/training stages.

In the forward passing phase of training, first, the synaptic current x through each synapse is calculated. With the updated synaptic current, some key variables in calculating the spike-train level post-synaptic potential (S-PSP) is updated for the same synapse, which will be introduced in detail in next sub-section and Fig. 4.1. The synaptic current update and the S-PSP update modules shown in Fig. 4.1 are shared by all input synapses. Hence, all input synapses of current neuron are processed in series. After all synaptic responses updated, the spike generation module calculates the updated neuron's membrane

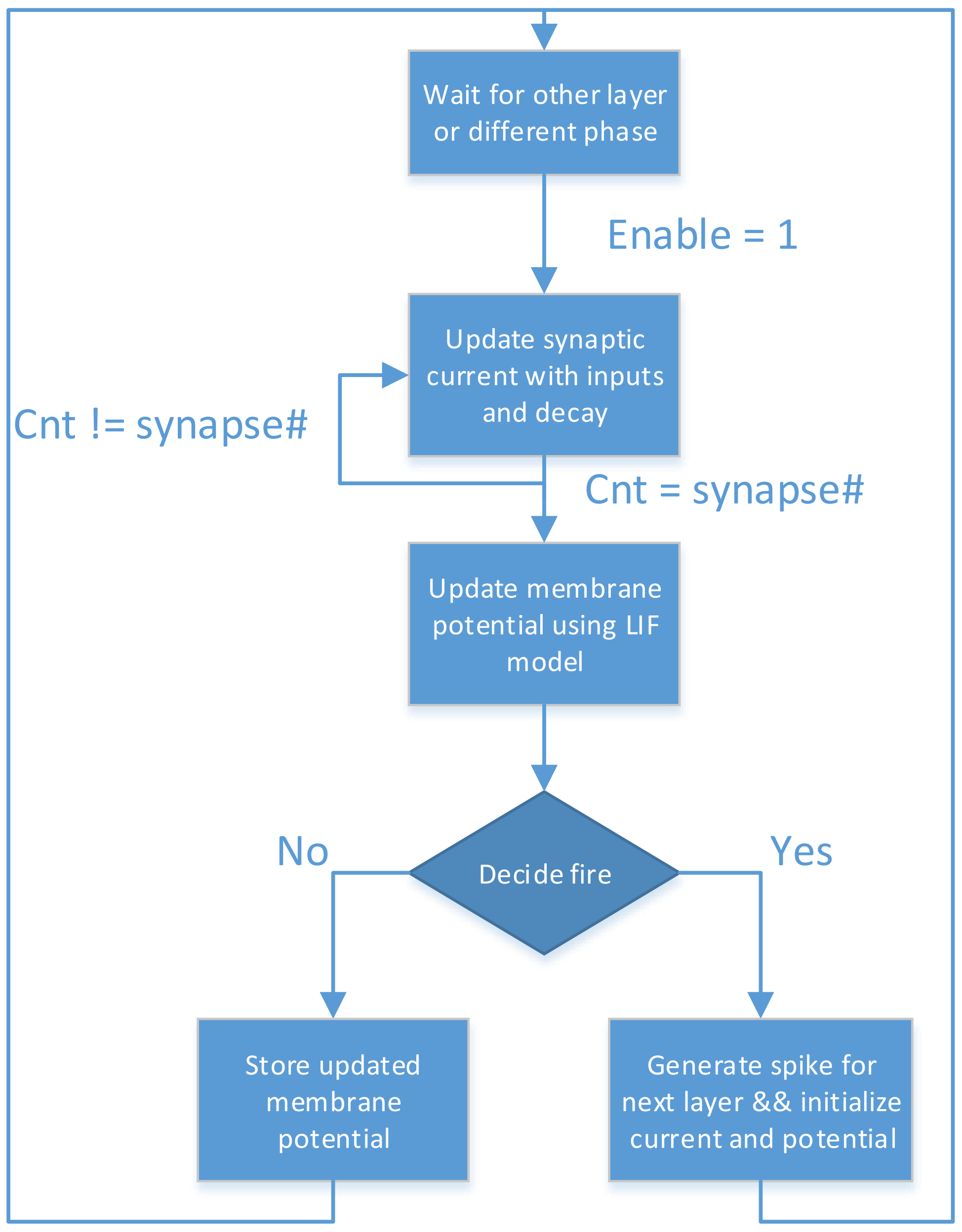
Wait for other layer or different phase
Enable = 1
Update synaptic current with inputs and decay
Cnt != synapse#
Cnt = synapse#
Update membrane potential using LIF model
Decide fire
No
Yes
Store updated membrane potential
Generate spike for next layer && initialize current and potential

Figure 4.2: A local controller containing control logic of synaptic current and membrane potential updating based on LIF neuron model as well as firing deciding to generate inputs of next layer

potential based on the sum of weighted currents of all input synapses and makes the firing decision of current biological time step. The model used in potential updating is the leaky integrate-and-fire (LIF) spiking neuron model. This firing activity can be treated as input of next layer and enable signal for variable updating of S-PSP module. In the backward passing phase of training, the ST-DFA module implements the proposed on-chip ST-DFA training algorithm, the output of which is then fed to the weight update module. Finally, the corresponding synaptic weight is updated and stored back to the weight memory. Similar to the feed forward blocks, the feedback functional modules are also shared among all input synapses.

4.2 Efficient On-chip S-PSP Calculation

One important component in the proposed ST-DFA algorithm is the spike-train level post-synaptic potential (S-PSP), $e_{i|j}$, in (3.3). As demonstrated in (2.6), by definition, $e_{i|j}$ is the effect of all firing events of the pre-synaptic neuron j on the post-synaptic neuron i via the synapse connecting these two neurons. However, direct implementation of (2.6) on hardware is very costly; all firing events of the pre- and post-synaptic neurons need to be stored during the process of training all input examples and excessive multiplication, division and exponentiation operations are involved, incurring much logic complexity and memory usage.

Instead of directly implementing 2.6,an online S-PSP calculation approach is proposed, which can dramatically reduce hardware overhead. Rather than recording all firing events of the two neurons during every biological step of feed forward passing phase and computing $e_{i|j}$ at once in the backward passing phase, in the forward part $e_{i|j}$ is accumulated and updated only when there's a firing event happening at current neuron i. Also, the updated $e_{i|j}$ is stored in the S-PSP memory of each neuron element.

Inspecting (2.3) and (2.6) reveals that $e_{i|j}$ is the normalized (by weight) of the contribution from the postsynaptic neuron j to the aggregated membrane potential of the postsynaptic neuron i. While the aggregated postsynaptic membrane potential is effectively tracked by the LIF model, each individual contribution $e_{i|j}$ to it can be accumulated exactly using the following equations:

$$\begin{aligned}
&\tau_s \frac{p_{i|j}(t)}{dt} = -p_{i|j}(t) + \sum_{t_j^{(f)}} D(t - t_j^{(f)}), \\
&\tau_m \frac{q_{i|j}(t)}{dt} = -q_{i|j}(t) + p_{i|j}(t), \\
&e_{i|j}(t) = \sum_{t_i^{(f)}} q_{i|j}(t_i^{(f)}),
\end{aligned} \tag{4.1}$$

where $p_{i|j}(t)$ is the (normalized) synaptic input from the neuron j to neuron i, which is part of (2.2), and $q_{i|j}(t)$ is interpreted as the (normalized) postsynaptic membrane voltage contribution from the neuron j to neuron i, which shall be reset to zero when the neuron i fires at a particular firing time $t_i^{(f)}$.

The hardware realization of (4.1) is based on discretizing it using the first-order Euler method with a fixed stepsize:

$$\begin{aligned}
&q_{i|j}[t+1] = (1 - \frac{1}{\tau_m})q_{i|j}[t] + p_{i|j}[t+1] \\
&p_{i|j}[t+1] = (1 - \frac{1}{\tau_s})p_{i|j}[t] + \frac{1}{\tau_s}\sum_{t_j^{(f)}} D_n(t - t_j^{(f)}) \\
&\begin{cases} e_{i|j}[t+1] += q_{i|j}[t+1] \\ q_{i|j}[t+1] = 0 \end{cases} \qquad if\ t+1 = t_i^{(f)},
\end{aligned} \tag{4.2}$$

where $D_n(\cdot)$ is the unit sample function and we have abused the notation by using t and $t+1$ to indicate a discrete time step and the step after that.

(4.2) allows $e_{i|j}$ to be accumulated in an online manner with great hardware efficiency and its implementation is shown in Fig. 4.4. At each time step, the value of $p_{i|j}$ is first updated based on synaptic inputs, followed by the updates of $p_{i|j}$ and $e_{i|j}$, controlled by the FSM states of the local controller. The new structure of local controller with online S-PSP variables learning is shown in Fig. 4.3. The shaded blocks in Fig. 4.4 are registers used to store the current-time variable values. Both decay constants τ_s and τ_m are set to be a power of 2 such that multiplications/divisions can be efficiently realized by using shift operations. The updated $e_{i|j}$ is stored in the S-PSP memory and retrieved by the ST-DFA module during the backward training passing phase.

4.3 Efficient On-chip ST-DFA Implementation

Fig. 4.5 depicts the ST-DFA module in hidden neurons shown in Fig. 4.1. As in (3.3), for each hidden neuron i, the inner product between the error vector δ_l^o from the output layer and the i-th column of the random feedback matrix B of the corresponding layer is computed. The inner product is then multiplied with $e_{i|j}$ to produce the weight update value Δw_{ij} for the j-th input synapse. All these inner products for different synapses are computed in series and would result in large hardware and power overheads. Furthermore, if each entry of the feedback matrix is set to be a high-bit resolution random number, high memory usage is required for storage.

To mitigate the above design complexity, A hardware-friendly realization of ST-DFA is proposed, named ST-DFA-2. ST-DFA-2 is based on the key observation from extensive algorithmic experiments that the feedback matrix B need not be generated in a true random manner; setting each entry b_{li} of B to one of a small set of fixed numbers at random is sufficient for achieving good training performance. Furthermore, the set of fixed numbers can be optimized for hardware efficiency. For this, this set is constructed by making each number a signed power of 2 with low-bit resolution such that the multiplications in (3.3) can be implemented by shift operations and storage for B is kept at minimal. With

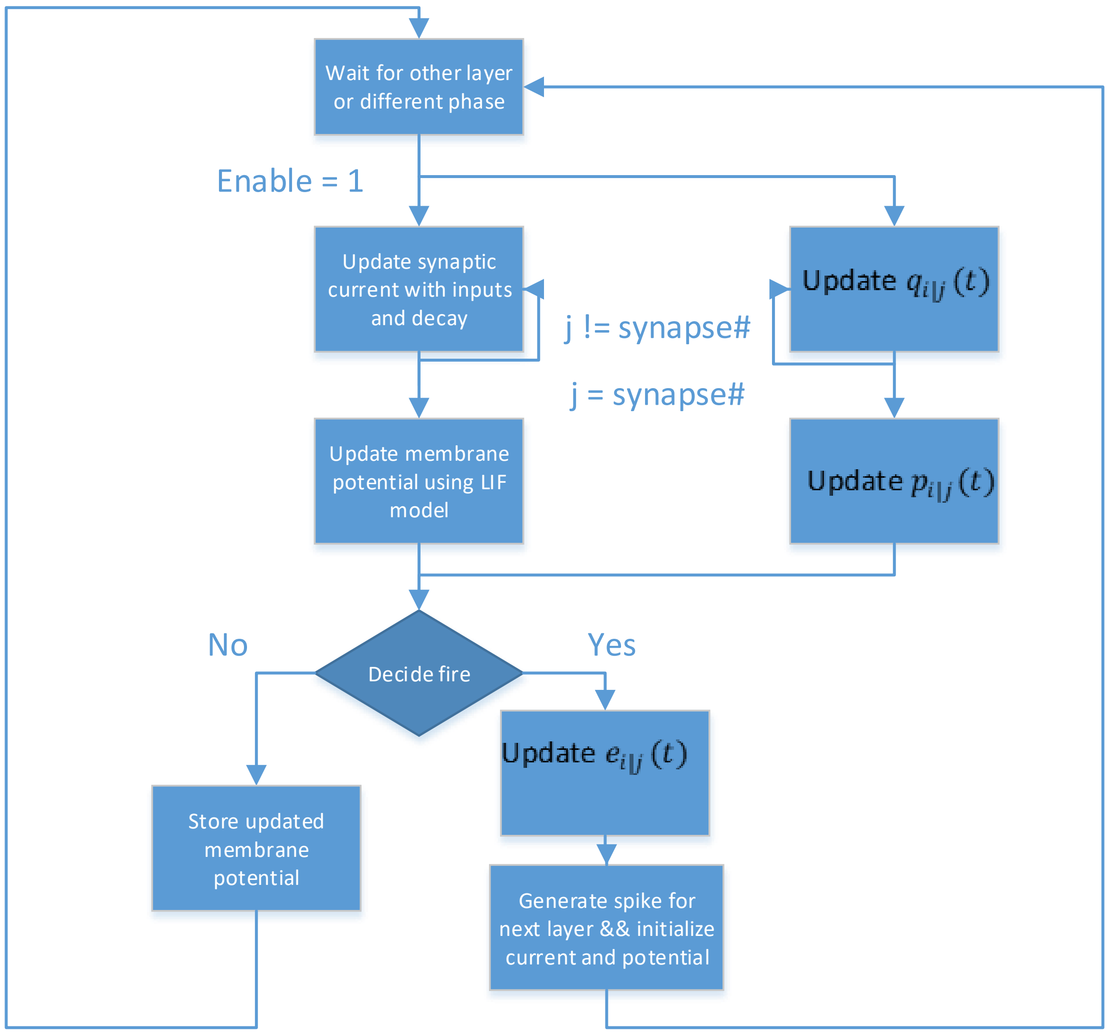
Wait for other layer or different phase
Enable = 1
Update synaptic current with inputs and decay
Update $q_{i|j}(t)$
j != synapse#
j = synapse#
Update membrane potential using LIF model
Update $p_{i|j}(t)$
No
Decide fire
Yes
Update $e_{i|j}(t)$
Store updated membrane potential
Generate spike for next layer && initialize current and potential

Figure 4.3: The final control module which combines original controller with updating values of $p_{i|j}$ and $p_{i|j}$ to support online S-PSP calculation

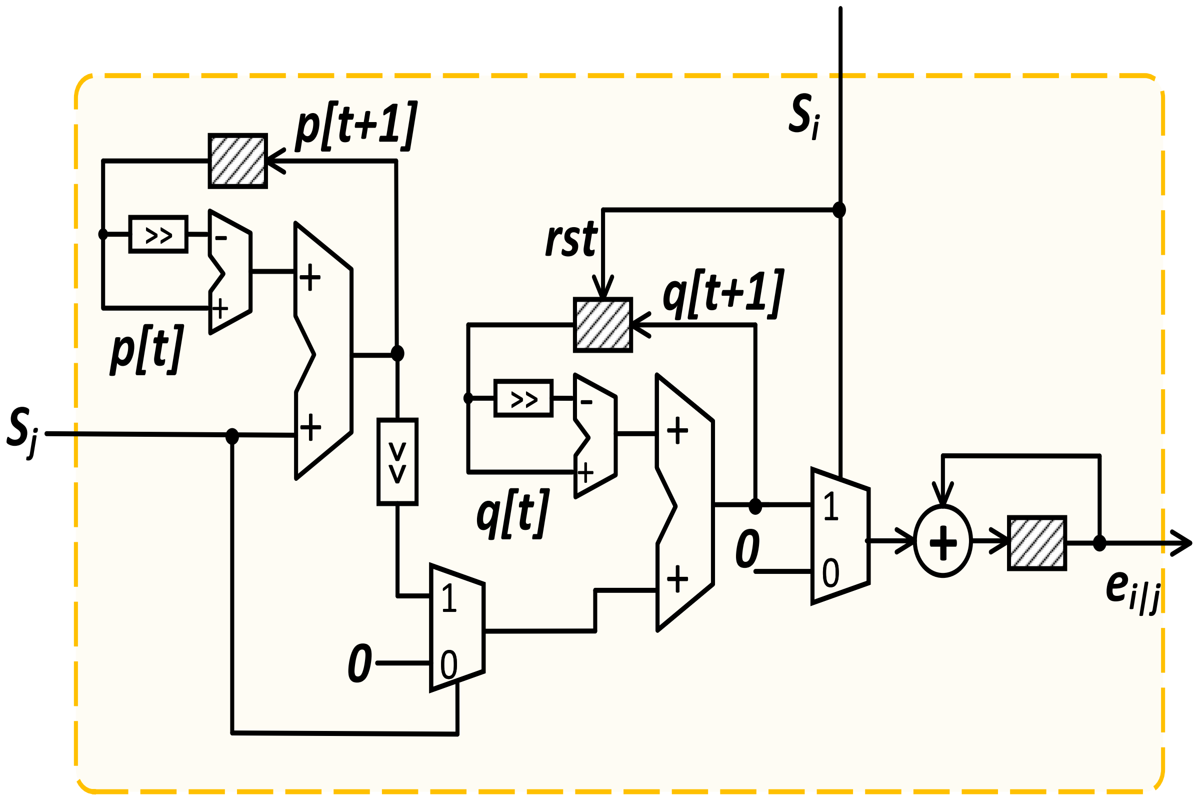

Figure 4.4: On-line S-PSP calculation onchip.

this optimization, not only the storage requirement of internal variables will be reduced, but also the computation complexity and resource cost of whole back propagation phase will decrease.

Fig. 4.5 illustrates the computation of each weight update. The corresponding inner product is computed by accumulating the element-wise products. The *idx* signal selects a particular element in the error vector δ_l^o and its shift amount m_{il}, which is set by the corresponding b_{li} in the B matrix according to $|b_{li}| = 2^{m_{il}}$. If b_{li} is negative, the shift result is converted to its compliment before added to δ_i. Finally, the resulting δ_i is multiplied with the S-PSP $e_{i|j}$ to get the weight update value Δw_{ij} for the current synapse.

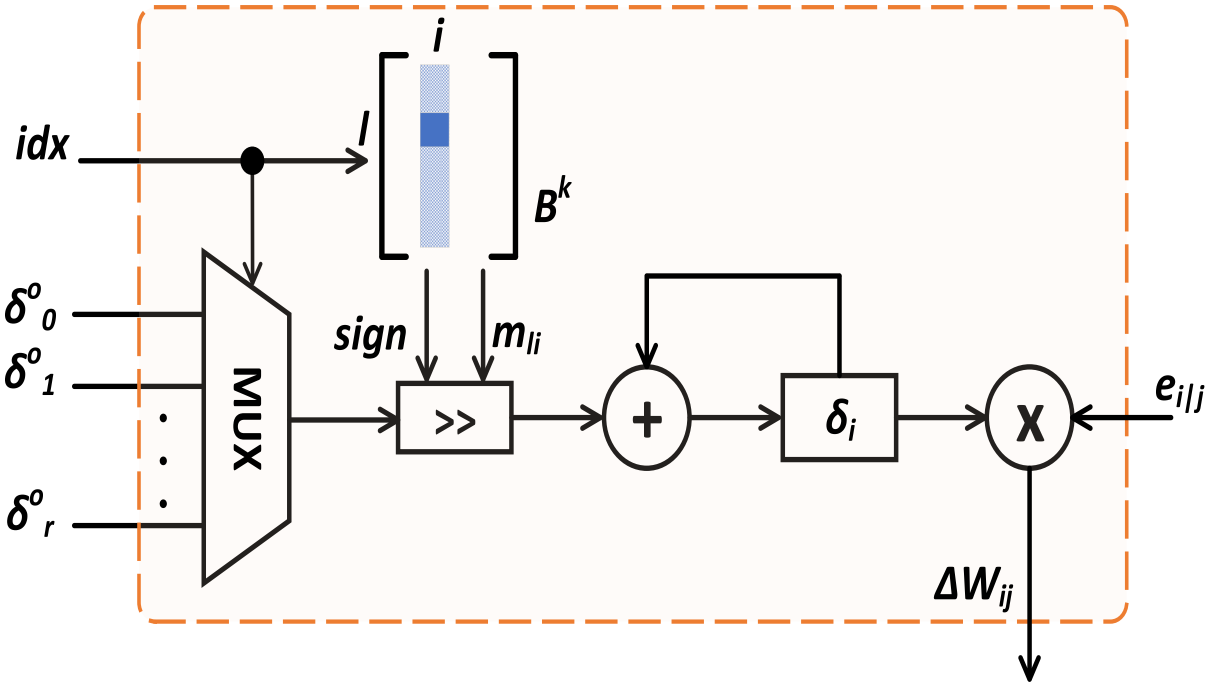

Figure 4.5: On-chip ST-DFA weight update computation.

5. EXPERIMENTAL SETTINGS AND RESULTS

5.1 Experimental Settings and Benchmarks

The performance vs. hardware overhead tradeoffs of the proposed on-chip ST-DFA training are measured on several feed-forward SNN neural processors on the Xilinx ZC706 platform. The classification performances are evaluated by software simulation of the digital computations with the actual bit resolutions implemented on FPGA. Major SNN variables, for example synaptic weight w, S-PSP $e_{i|j}$ and membrane potential v, are in the fixed-point representation. Each w is a signed 17-bit variable with 12-bit fractional. 11 bits are used for each unsigned variable $e_{i|j}$ with 6-bit fractional and 9 bits are used for each signed variable v with 3-bit fractional. FPGA prototypes of SNN neural accelerators are designed on the Xilinx ZC706 platform for design overhead and power/energy analysis.

Three datasets are employed for evaluation: MNIST[13], N-MNIST, or the neuromorphic version of MNIST [19], and the 16-speaker English letter subset of the TI46 speech corpus [14]. The MNIST handwritten digit dataset [13] contains 60k training and 10k testing examples, each of which is a 28×28 grayscale image. Each pixel value of the MNIST image is converted into a spike train using Poisson sampling and the probability of spike generation is proportional to the pixel intensity. Due to the limited hardware resources available on the Xilinx Zynq ZC706 board, each image is cropped to include only the 14×14 pixels around the center for FPGA evaluation.

The N-MNIST dataset [19] is a neuromorphic version of MNIST. The static digit images of MNIST are converted into spike trains using a dynamic vision sensor (DVS) [20] moving on a pan-tilt unit. The image is resized to 34×34 since the relative shift of images during the saccade process is required. Two kinds of spike events, ON and OFF, are recorded since the intensity can either increase or decrease. Thus, each N-MNIST image has $34 \times 34 \times 2 = 2312$ spike sequences lasting for about $300ms$. We reduce the time resolution of the N-MNIST images by 500x to speed up the processing.

The TI46 Speech corpus [14] contains spoken English letters from 16 speaker. There are 4,142 and 6,628 spoken English letters for training and testing, respectively. The continuous temporal speech waveforms are first pre-processed by the Lyon's ear model [21] and then encoded into 78 spike trains using the BSA algorithm [22].

Among these datasets, MNIST and TI46 are tested on both software and hardware while N-MNIST is only tested on software simulation due to that the available FPGA resources are not sufficient to support the large number of spike trains. Moreover, to thoroughly assess the classification performance and hardware benefits of our proposed spike-train level direct feedback alignment (ST-DFA), we build multiple SNNs with different network depths and widths.

5.2 Classification Accuracies

The proposed spike-train level direct feedback alignment (ST-DFA) algorithm is inspired by the spike-train level backpropagation HM2-BP algorithm. In [10], HM2-BP is compared with other state-of-the-art spiking or non-spiking BP methods such as spike-based BP [6], STBP [9], temporal coding BP [23] and non-spiking BP [24] on MNIST and N-MNIST. Apart from its high efficiency due to the spike-train level processing, HM2-BP outperforms or is on a par with all these recently developed algorithms. For example, with a single hidden layer of 800 neurons, HM2-BP can achieve 98.93% accuracy on MNIST while [24] gets up to 98.30%. HM2-BP obtains 98.88% accuracy on N-MNIST compared with 97.80% by [23]. Moreover, HM2-BP delivers competitive performance on challenging benchmarks such as the 16-speaker spoken English letters of TI46 Speech corpus [14] and 47-class image recognition dataset Extended MNIST (EMNIST) [25].

As presented in Section 3, ST-DFA propagates the errors δ from the output layer to each hidden layer directly without layer by layer error backpropagation through symmetric weights matrices. In Section 4.3, ST-DFA is further optimized by setting each entry of the random feedback matrix B to a power of 2, leading to the hardware-friendly ST-DFA-2 algorithm. In this work, feedback matrix entries are randomly chosen from the set $\{-4, -2, -1, 0, 1, 2, 4\}$ for ST-DFA-2.

As mentioned in last chapter, in terms of complexity, ST-DFA-2 costs much less compared to HM2-BP particularly for hardware implementation. First, it reduces the complexity of HM2-BP by adding feedback alignments. Second, in this particular setting of feedback matrix entries, each value of ST-DFA-2 feedback alignments only costs 4 bits of storage rather than that for a floating point number. Finally, with ST-DFA-2, backpropagating the error δ to each hidden layer can be done with only additions and shifts while high-resolution multiplications are required in HM2-BP.

Table 5.1 compares the inference accuracies of HM2-BP, ST-DFA, and ST-DFA-2 on MNIST, N-MNIST, and TI46. Compared to HM2-BP, ST-DFA and ST-DFA-2 still maintain rather competitive performance while the low computational cost and hardware-friendliness of ST-DFA-2 translate into huge hardware resources and energy overhead savings as shown later. It shall be noted that in comparison with ST-DFA, ST-DFA-2 does not necessarily degrade performance; it can even slightly outperform ST-DFA in practice.

5.3 FPGA Hardware Evaluations

Several FPGA SNN accelerators are built on the targeted Xilinx ZC706 platform, the sizes of which are decided considering the available resources onchip. Table 5.2 shows the resource and energy overhead as well as the inference accuracies of these SNN accelerators with on-chip ST-DFA-2.

As shown in the table, the implemented networks have either one or two hidden

Table 5.1: Inference accuracy comparison of HM2BP, ST-DFA and ST-DFA-2. All SNNs are fully connected networks with a single hidden layer of 800 neurons. MNIST: 28x28 input resolution; N-MNIST: 2,312 input spike trains; 16-speaker TI46: 78 input spike trains.

Dataset	Learning rule & Network structure	Accuracy
MNIST	HM2-BP: 784-800-10	98.93%
MNIST	ST-DFA: 784-800-10	98.64%
MNIST	ST-DFA-2: 784-800-10	98.74%
N-MNIST	HM2-BP: 2312-800-10	98.88%
N-MNIST	ST-DFA: 2312-800-10	98.47%
N-MNIST	ST-DFA-2: 2312-800-10	98.59%
TI46	HM2-BP: 78-800-10	89.92%
TI46	ST-DFA: 78-800-10	87.00%
TI46	ST-DFA-2: 78-800-10	87.31%

layer(s), and each hidden layer has 50 or 100 neurons. Numbers of input and output neurons are application-dependent. Training powers are estimated by the Xilinx Power Analyzer based on application-specific workloads. The training latency and training energy are for training a representative input example of the corresponding dataset using one iteration of forward and backward passes. Table 5.2 indicates that the SNNs integrated with ST-DFA-2 in general have efficient FPGA resource utilization as well as low training energy dissipation. Furthermore, with a trimmed down input size and/or constrained network size, the FPGA SNNs with on-chip ST-DFA-2 can still deliver competitive classification performance in reference to the simulated accuracies achieved at full input size and by larger networks reported in Table 5.1.

To better illustrate the cost-effectiveness of the proposed ST-DFA algorithm, we also compare the overheads of implementing HM2-BP vs. ST-DFA-2 in a fully-connected SNN FPGA with two hidden layers in Table 5.3. Training latency of the backward pass of the corresponding SNN neural processor is also presented in the table. We do not consider forward pass latency and inference latency since they do not differ significantly in the two cases. The results in the table indicate that ST-DFA is much more efficient in terms of hardware implementation on both resource utilization and backward pass latency compared with HM2-BP. The ST-DFA-2 based SNN neural processor saves 18% on LUTs, 76.7% on DSPs and 31.6% on backward phase latency compared with the HM2-BP based SNN. The large additional hardware overhead and backward latency of HM2-BP mainly come from the layer-by-level error propagation and the required multiplication operations. Moreover, as the network goes deeper, the backward phase latency grows proportionally in HM2-BP, while in ST-DFA the backward latency will not affect by the network depth since the error processing is concurrently executed in all hidden layers. With the proposed ST-DFA algorithm, we have sidestepped the complex backpropagation and enabled cost-effective on-chip training for multi-layer SNNs.

Table 5.2: Overheads and inference performances of the fully-connected SNNs with on-chip ST-DFA-2.

MNIST (14x14 input resolution) @100 MHz							
	Resource Utilization			**Training Power (mW)**	**Training Latency (mS)**	**Training Energy (mJ)**	**Accuracy**
	LUTs	**FFs**	**DSPs**				
196-50-10	33484	6836	60	113	3.998	0.452	95.48%
195-50-50-10	62989	12516	110	125	4.836	0.604	95.87%
196-100-10	73027	12329	110	224	4.802	1.076	96.86%
196-100-100-10	126482	23331	210	275	6.445	1.772	97.23%
TI46 (16-speaker Spoken English Letters) @100 MHz							
	Resource Utilization			**Training Power (mW)**	**Training Latency (mS)**	**Training Energy (mJ)**	**Accuracy**
	LUTs	**FFs**	**DSPs**				
78-50-26	38220	8826	76	73	3.688	0.269	73.34%
78-50-50-26	74709	14641	126	87	5.123	0.445	76.45%
78-100-26	64280	14096	126	113	5.089	0.575	77.64%
78-100-100-26	145452	30546	226	185	7.929	1.467	87.40%

Table 5.3: Overheads of an FPGA SNN with on-chip HM2-BP vs. ST-DFA-2 (Network size:196-100-100-10)

	LUTs	**FFs**	**DSPs**	**Backward Phase Latency (uS)**
HM2-BP	154477	23462	900	17.560
ST-DFA	126482	23331	210	12.010
	Normalized LUTs	**Normalized FFs**	**Normalized DSPs**	**Normalized B-P Latency**
HM2-BP	122%	101%	429%	146%
ST-DFA	100%	100%	100%	100%

6. CONCLUSION

This work proposes a novel spike-level direct feedback alignment (ST-DFA) algorithm for training multi-layer spiking neural networks (SNNs) with improved bio-plausibility and scalability over traditional backpropagation algorithms. Moreover, it is demonstrated that the ST-DFA algorithm with its hardware-friendly optimized implementation enable efficient on-chip training of FPGA SNN neural processors while delivering competitive classification performance for practical speech and image recognition tasks.

7. REFERENCES

[1] Z. Y. Xiurui Xie, Hong Qu and J. Kurths, "Efficient training of supervised spiking neural network via accurate synaptic-efficiency adjustment method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1411 – 1424, 2017.

[2] S. J. Saeed RezaKheradpisheh, Mohammad Ganjtabesh and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Nerual Networks*, vol. 99, no. 1, pp. 56–67, 2018.

[3] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[4] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[5] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[6] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[7] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[8] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[9] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, 2018.

[10] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems*, pp. 7005–7015, 2018.

[11] P. J. Werbos *et al.*, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[12] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in neural information processing systems*, pp. 1037–1045, 2016.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] M. Liberman, R. Amsler, K. Church, E. Fox, C. Hafner, J. Klavans, M. Marcus, B. Mercer, J. Pedersen, P. Roossin, D. Walker, S. Warwick, and A. Zampolli, "TI 46-word LDC93S9," 1991.

[15] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, p. 13276 EP, 11 2016.

[16] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers in neuroscience*, vol. 11, p. 324, 2017.

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

[18] D. E. Rumelhart, J. L. McClelland, P. R. Group, *et al.*, *Parallel distributed processing*, vol. 1. MIT press Cambridge, 1988.

[19] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[20] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[21] R. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7, pp. 1282–1285, IEEE, 1982.

[22] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 4, pp. 2825–2830, IEEE, 2003.

[23] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2018.

[24] D. Neil, M. Pfeiffer, and S.-C. Liu, “Phased lstm: Accelerating recurrent network training for long or event-based sequences,” in *Advances in Neural Information Processing Systems*, pp. 3882–3890, 2016.

[25] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: an extension of mnist to handwritten letters,” *arXiv preprint arXiv:1702.05373*, 2017.