IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2020, accepted March 28, 2020, date of publication April 8, 2020, date of current version April 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2986490

Optimal Distribution of Spiking Neurons Over Multicore Neuromorphic Processors

GUHYUN KIM[1,2], VLADIMIR KORNIJCUK[1], JEESON KIM[1], CHEOL SEONG HWANG[2,3], AND DOO SEOK JEONG[1], (Member, IEEE)

[1] 1Division of Materials Science and Engineering, Hanyang University, Seoul 04763, South Korea
[2]Department of Material Science and Engineering, Seoul National University, Seoul 08826, South Korea
[3]Inter-University Semiconductor Research Center, Seoul National University, Seoul 08826, South Korea

Corresponding author: Doo Seok Jeong (dooseokj@hanyang.ac.kr)

This work was supported by a research of the National Research Foundation of Korea under Grant NRF-2019R1C1C1009810.

ABSTRACT In a multicore neuromorphic processor embedding a learning algorithm, a presynaptic neuron is occasionally located in a different core from the cores of its postsynaptic neurons, which needs neuron-to-target core communication for inference through a network router. The more neuron-to-target core connections, the more workload is imposed on the network router, which the more likely causes event routing congestion. Another significant challenge arising from a large number of neuron-to-core connections is data duplication in multiple cores for the learning algorithm to access the full data to evaluate weight update. This data duplication consumes a considerable amount of on-chip memory while the memory capacity per core is strictly limited. The optimal distribution of neurons over cores is categorized as an optimization problem with constraints, which may allow the discrete Lagrangian multiplier method (LMM) to optimize the distribution. Proof-of-concept demonstrations were made on the distribution of neurons over cores in a neuromorphic processor embedding a learning algorithm. The choice of the learning algorithm was twofold: a simple spike timing-dependent plasticity learning rule and event-driven random backpropagation algorithm, which are categorized as a two- and three-factor learning rule, respectively. As a result, the discrete LMM significantly reduced the number of neuron-to-core connections for both algorithms by approximately 55% in comparison with the number for random distribution cases, implying a 55% reduction in the workload on the network router and a 52.8% reduction in data duplication. The code is available on-line (https://github.com/guhyunkim/Optimize-neuron-distribution).

INDEX TERMS Multicore neuromorphic processor, spiking neural network, discrete Lagrange multiplier method, neuron distribution optimization.

I. INTRODUCTION

Neuromorphic hardware usually emulates spiking neural network (SNN), which is a time-dependent model, to accelerate large-scale model simulation (emulation)[1] To date, various prototypes of neuromorphic hardware have been introduced, including TrueNorth [2], SpiNNaker [3], Loihi [4], MorphIC [5], DYNAPs [6], and BrainScaleS [7]. Multicore architecture is commonplace in most prototypes, which employs many cores, each with a group of neurons and their fan-in synapses. A fan-in (fan-out) synapse of a neuron explains a synapse between the neuron and its presynaptic (postsynaptic) neuron. The multicore architecture allows the neurons and their fan-in synapses in the same core to share hardware resources such as arithmetic logic circuits and memory for common variables among them when they are optimally distributed over cores. Because the forward propagation of events (spikes) to a neuron should address its fan-in synaptic data, placing the neuron and its fan-in synapses in the same core avoids unnecessary data transfer between cores. Therefore, all data required for the update on neuronal variables for a neuron are available within the source core, which are referred to as topologically local data.

Communication between cores commonly follows the address-event representation (AER) protocol [8], [9] Upon an event from a particular neuron in a particular core, the addresses of the neuron and core are delivered to the target neurons through communication buses; consequently, the neuronal and synaptic variables are updated in the cores of the target neurons. The forward propagation of events is

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Wang.

the basic principle of inference so that the multicore architecture with topologically local data supports energy-efficient inference.

Inference aside, an important concern is on-chip learning with an appropriate learning rule (learner) embedded in each core of a multicore neuromorphic processor. This enables the hardware to learn in real time without external general-purpose hardware. Such embedded learners need to satisfy strict constraints for energy- and data-efficient learning [10] The constraints include (i) event-driven update, (ii) use of topologically and temporally local data only, and (iii) minimal use of variables in the learner. If events are sparse, it is desirable for the learner to update the weight upon events rather than dedicated update periods as for deep learning, which can avoid unnecessary update processes. Given that the state-of-the-art architecture of neuromorphic hardware allows inter-core communication to transfer the addresses of a firing neuron and its core only, the learner needs the full access to the variables incorporated into the algorithm within the core. However, because the memory capacity per core is strictly limited, the learner is desired to use local (rather than global) data only. Generally, different learners include different numbers of variables. Two-factor learning rules, e.g., Hebb rule [11], [12] and spike timing-dependent plasticity (STDP) rule [13], [14], use two variables (one pre- and one postsynaptic variable), and three-factor learning rules [15] uses one additional variable. It is desired for the learner to use the minimal number of variables because of the limited memory capacity per core.

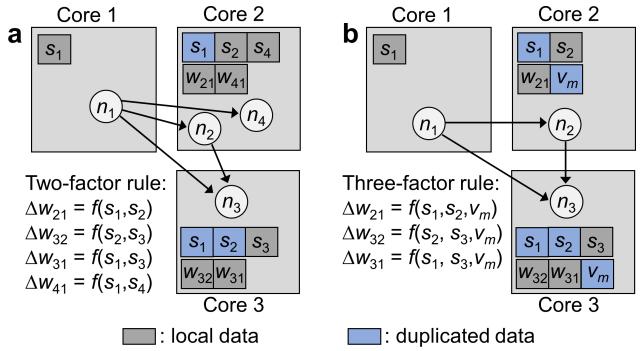


FIGURE 1. Schematic of data allocation to cores of a digital neuromorphic processor embedding (a) a two-factor and (b) a three-factor learning rule. A pre- and postsynaptic state variable are denoted by s_1 and s_2, respectively. A third factor for a three-factor learning rule is denoted by v_m. Neuron i ($i = 1, 2, 3, 4$) is denoted by n_i.

Pre- and postsynaptic neurons are occasionally placed in different cores as illustrated in Fig. 1. For a toy network in Fig. 1(a), when Neuron 1 in Core 1 fires a spike, the addresses of the neuron and its core are delivered to its postsynaptic neurons (Neurons 2 and 4 in Core 2 and Neuron 3 in Core 3) through a network router. The addresses are delivered to a destination core once irrespective of the number of postsynaptic neurons in the same core. This is because the event is re-distributed within the core as for Loihi [4], increasing the bandwidth (events/s) of the network router. Therefore, placing the postsynaptic neurons in the same core alleviates the workload on the router, thereby avoiding traffic congestion (routing delays). Additionally, because the learner needs the full access to the data for weight update, the variable for Neuron 1 should be duplicated in the target cores. For instance, considering a two-factor learning rule, the variable for Neuron 1 (s_1) is duplicated in Cores 2 and 3 for each learner to use the duplicated variable to update. Given that the neurons in the same core can share the duplicated variable, minimizing the number of target cores can minimize data duplication. For a three-factor learning rule, such data duplication and the consequent use of memory are generally severer as shown in Fig. 1(b); each target core (Cores 2 and 3) duplicates the variables in different cores (here, s_1 and v_m) because the learner needs to access them. Therefore, the optimal distribution of neurons over cores reduces the number of neuron-to-core connections, which consequently reduces the probable traffic congestion and data duplication.

A naïve rule for optimal distribution is to place all postsynaptic neurons in the same core. However, generally, the neurons in a network are intertwined; the complexity in network topology renders the naïve rule infeasible. In this regard, we propose a method to minimize neuron-to-core connections and consequent data duplication. This method is based on the Lagrange multiplier method (LMM) with an appropriate objective function and constraints on the neuron- and synapse-accommodation capacity of a core. The model architecture of our concern features as follows:

- Each core has no local channel for local event routing (when pre- and postsynaptic neurons are located in the same core), so that all events are routed by the network router.
- Each event is distributed within a target core when the target core includes multiple postsynaptic neurons.
- Each neuron is placed in the same core as its fan-in synapses and its own state variable(s) that is addressed when updating its fan-in and fan-out synaptic weights.
- The learner in each core has the full access to all state variables of the algorithm such that the state variables located in different cores are duplicated in the core.

In fact, these constraints correspond to those of Loihi [4], which was taken as a benchmark architecture in this study. Loihi is comprised of 128 neuromorphic cores; each maximally includes 1,024 point neurons, 4,096 fan-out connections to other cores, and 4,096 fan-in connections from other cores. A single fan-in connection to a core is virtually wired to multiple target neurons through fan-in synapses. The numbers of neurons and synapses in the core are variable with the limit posed by on-core memory capacity. The cores are interconnected through asynchronous network-on-chips (NoCs), each of which is shared by four neighboring cores. Upon an event from a particular core, a data packet of target core and axon indices is emitted and delivered to the target cores through a chain of NoCs.

Section II introduces the basic principle of the LMM (Section II.A) and elaborates its application to the task

of optimal distribution of neurons over multiple cores (Section II.B), and estimation of additional memory usage due to data duplication (Section II.C). Section III presents the optimization results for two learning algorithms: spike timing dependent plasticity (STDP) rule [13], [14] and event-driven random backpropagation (eRBP) algorithm [16] as a representative two- and three-factor learning rule, respectively. The application of this optimization method to different model architectures is discussed in Section IV. Finally, Section V concludes this study.

II. OPTIMIZATION METHOD

A. LAGRANGE MULTIPLIER METHOD

The LMM finds the local maxima or minima of a function f with given equality constraints c's $= 0$ [17] Assume a task of optimizing $f(x, y)$ with a single equality constraint $c(x, y) = 0$. The maximum or minimum function value k is placed at a contact point between the two functions $f(x, y) = k$ and $c(x, y) = 0$ [18] Because the two functions have a common tangent at the contact point, their gradient vectors are parallel to each other:

$$\nabla_{x,y} f = -\lambda \nabla_{x,y} c. \tag{1}$$

The relative magnitude of the two gradient vectors is defined by λ, which is referred to as a Lagrange multiplier. Therefore, a solution to (1) is an optimal point (x, y) for function $f(x, y)$ given $c(x, y) = 0$. The Lagrangian function L for function $f(x, y)$ is given by $L(x, y, \lambda) = f(x, y) + \lambda c(x, y)$. The gradient vector of L is expressed as $\nabla_{x,y,\lambda} L(x, y, \lambda) = \nabla_{x,y}[f(x, y) + \lambda c(x, y)] + \nabla_{\lambda}[\lambda c(x, y)]$. This equation leads to the following equivalence:

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = 0 \Longleftrightarrow$$
$$\nabla_{x,y} f = -\lambda \nabla_{x,y} c \;\&\; c(x, y) = 0,$$

satisfying the condition in (1) and constraint $c(x, y) = 0$. Therefore, a solution to (1) is acquired by solving $\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$. The same holds for a multivariable function $f(\boldsymbol{x})$ ($\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$) with multiple constraints $\boldsymbol{c}(\boldsymbol{x}) = 0$, where $\boldsymbol{c}(\boldsymbol{x}) = [c_1(\boldsymbol{x}), c_2(\boldsymbol{x}), \ldots, c_k(\boldsymbol{x})]$. Its Lagrangian function is expressed as $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{x})$, where $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_k]$.

For discrete $\boldsymbol{x}$, a discrete Lagrangian function L_d is defined to minimize $f(\boldsymbol{x})$ subject to $c(\boldsymbol{x}) = 0$ as proposed by Wah and Wu [19]:

$$L_d(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot H(\boldsymbol{c}(\boldsymbol{x})), \tag{2}$$

where H is a continuous transformation function satisfying

$$\begin{cases} H(\boldsymbol{c}(\boldsymbol{x})) = 0 & if\ \boldsymbol{c}(\boldsymbol{x}) = 0 \\ H(\boldsymbol{c}(\boldsymbol{x})) > 0 & otherwise. \end{cases} \tag{3}$$

The discrete LMM accepts not only equality but also inequality constraints such as $h(\boldsymbol{x}) \leq 0$. In this case, the inequality constraint is converted to an equality constraint $c(\boldsymbol{x})$ such that $c(\boldsymbol{x}) = \max(0, h(\boldsymbol{x}))$, which indicates $c(\boldsymbol{x}) = 0$ when $h(\boldsymbol{x}) \leq 0$. Similarly, the inequality constraint $h(\boldsymbol{x}) \geq 0$ can be converted to $c(\boldsymbol{x})$ $(= \min(0, h(\boldsymbol{x})))$.

Because the minimal value of $f(\boldsymbol{x})$ is concerned, we evaluate the direction of maximum potential drop (DMPD) for L_{d} at a point $\boldsymbol{x}_i$ with a given λ_i as follows:

$$\Delta_{\boldsymbol{x}} L_d(\boldsymbol{x}_i, \boldsymbol{\lambda}_i) = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i,$$

where $\boldsymbol{x}_{i+1} = \underset{\boldsymbol{x}_{i+1} \in \alpha(x_i)}{argmin} = L_d(\boldsymbol{x}_{i+1}, \boldsymbol{\lambda}_i)$. Therefore, the $\boldsymbol{x}$ value on the subsequent iterative step $(\boldsymbol{x}_{i+1})$ causes the largest decrease in L_{d}, which is chosen from a set of user-defined neighborhood points α including the current point $\boldsymbol{x}_i$. Subsequently, λ is updated such that

$$\boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}_i + \eta H(\boldsymbol{c}(\boldsymbol{x}_i)). \tag{4}$$

Here, the update rate for λ is determined by η. The optimal $\boldsymbol{x}$ and λ are reached by iterating this update step until $H(\boldsymbol{c}(\boldsymbol{x})) = 0$, i.e., $\boldsymbol{c}(\boldsymbol{x}) = 0$, which indicates the local minimal value of $f(\boldsymbol{x})$ [19].

B. APPLICATION OF DISCRETE LMM TO OPTIMIZATION OF SPIKING NEURON DISTRIBUTION

We consider an SNN with Q neurons distributed over M neuromorphic cores. To describe the SNN topology, we define matrix $\boldsymbol{T}$ $\left(\in \mathbb{Z}^{Q \times Q};\ T[i, j] \in \{0, 1\}\right)$. Matrix $\boldsymbol{T}$ indicates the synaptic connection from presynaptic neuron i to postsynaptic neuron j such that $T[i, j] = 1$ when the connection is present, and $T[i, j] = 0$ otherwise. The distribution of Q neurons over M neuromorphic cores is defined by matrix $\boldsymbol{X}$ $\left(\in \mathbb{Z}^{Q \times M};\ X[i, j] \in \{0, 1\}\right)$ such that $X[i, j] = 1$ when neuron i is placed in core j, and $X[i, j] = 0$ otherwise. The product $T[i,:]X[:, j]$ results in the number of synaptic connections from presynaptic neuron i to its postsynaptic neurons in core j. Note that $[i,:]$ ($[:, i]$) denotes all elements in the ith row (column).

The discrete LMM minimizes the objective function f in (2), which should be chosen considering the architecture of a multicore neuromorphic processor. Here, we consider the duplication of presynaptic state variables over multiple cores as a critical cause of inefficient usage of memory. When a core includes multiple postsynaptic neurons of a presynaptic neuron in a different core, the postsynaptic neurons can share the presynaptic state variables [4], [20] Therefore, irrespective of the number of such postsynaptic neurons, the presynaptic state variables are duplicated once in the core. In this case, the number of such duplications is one—equal to the number of neuron-to-core, rather than neuron-to-neuron, connections as for Loihi [4]. Considering this architectural aspect, we chose the total number of neuron-to-core connections (N_{NC}) as the objective function. The neuron-to-core connection number for a given neuron and core is binary; Neuron 1 in Fig. 1(a) has a fan-out connection with Core 2 so that the neuron-to-core connection number is one unlike the neuron-to-neuron connection number, which is two in this specific case. Note that the total neuron-to-core connection

number N_{NC} includes connections of neurons to their own cores, i.e., self-connections.

To evaluate N_{NC}, we express a neuron-to-core connection map using matrix $\boldsymbol{P}$ that is defined as

$$P[i,j] = \begin{cases} 1 & if\ T[i,:]X[:,j] > 0 \\ 0 & otherwise. \end{cases} \quad (5)$$

A nonzero element $P[i, j]$ indicates the presence of connection from neuron i to core j. Therefore, adding all elements in matrix $\boldsymbol{P}$ yields the number of neuron-to-core connections N_{NC}:

$$N_{\mathrm{NC}} = \sum_{i=0}^{Q-1} \sum_{j=0}^{M-1} P[i,j]. \quad (6)$$

Given SNN topology $\boldsymbol{T}$, N_{NC} is a function of $\boldsymbol{X}$ only, which defines the distribution of the neurons over multiple cores.

We consider three constraints (c_1, c_2, and c_3) regarding the architecture of a multicore neuromorphic processor.

- Constraint 1: each neuron is placed in a single core.
- Constraint 2: each core can hold maximal N neurons in total.
- Constraint 3: each core can hold maximal S synapses in total.

Constraint 1 indicates no duplication of a neuron over multiple cores. Constraints 2 and 3 mainly arise from the limited capacity of memory in a core as for Loihi [4]. Note that it is commonplace to place neurons and their fan-in synapses in the same core for efficient evaluation of synaptic transmission [20] Thus, the synapses in Constraint 3 are fan-in synapses. Constraints 1, 2, and 3 can be expressed as

$$\forall i, c_1[i] = \sum_{j=0}^{M-1} X[i,j] - 1 = 0, \quad (7)$$

$$\forall j, c_2[j] = max\left(\sum_{i=0}^{Q-1} X[i,j] - N, 0\right) = 0, \quad (8)$$

and

$$\forall j, c_3[j] = max\left(\sum_{i=0}^{L-1} T[i,:]X[:,j] - S, 0\right) = 0, \quad (9)$$

respectively. Vector $\boldsymbol{c}_1$ with elements in (7) is a Q-long vector because the index i is in the range 0 – (Q-1). Vectors $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ are M-long vectors with elements in (8) and (9), respectively. Similar to $N_{\mathrm{NC}}(\boldsymbol{X})$, $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$ are functions of $\boldsymbol{X}$ given SNN topology $\boldsymbol{T}$.

Considering the objective function N_{NC} and three constraints, a discrete Lagrangian function L_{d} is given by

$$L_d(\boldsymbol{X}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) = N_{NC} + [\boldsymbol{\lambda}_1 \cdot H(\boldsymbol{c}_1) + \boldsymbol{\lambda}_2 \cdot H(\boldsymbol{c}_2) + \boldsymbol{\lambda}_3 \cdot H(\boldsymbol{c}_3)]. \quad (10)$$

The Lagrange multipliers λ_1, λ_2, and λ_3 have the same dimension as $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$, respectively. Transformation function H was considered to be $H(x) = x^2$, satisfying the requirement (3). All elements of matrix $\boldsymbol{X}$ were initialized to zero, whereas the elements of vectors λ_1, λ_2, and λ_3 were initialized to one. The update rate η was set to 0.1. The set of neighborhood points α to find the DMPD at a given point $\boldsymbol{X}$ is defined as $\alpha(\boldsymbol{X}) = \{\boldsymbol{X}' : \|\boldsymbol{X}' - \boldsymbol{X}\|_1 \leq 1\}$, where $||\boldsymbol{x}||_1$ means the $L1$ norm of $\boldsymbol{x}$. Because matrix $\boldsymbol{X}$ is a $Q \times M$ binary matrix, it has $Q \times M$ neighborhoods with an $L1$ distance of one. Therefore, $|\alpha(\boldsymbol{X})| = Q \times M+1$, including $\boldsymbol{X}$'=$\boldsymbol{X}$ ($L1$ norm = 0). For a given SNN topology $\boldsymbol{T}$, matrix $\boldsymbol{X}$ and multipliers λ_1, λ_2, and λ_3 are updated iteratively to find the minimal N_{NC}. The optimal $\boldsymbol{X}$ is reached when $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$ become zero, satisfying the constraints in (7), (8), and (9). At the optimal $\boldsymbol{X}$, the second term of the right hand side of (10) becomes zero so that the N_{NC} value equals the Lagrangian function L_{d}. This optimization process is written in the pseudocode in Algorithm 1.

Algorithm 1 Updating the Neuron Distribution Matrix $\boldsymbol{X}$ $\left(\in \mathbb{Z}^{Q\times M}\right)$ Given Network Topology Matrix $\boldsymbol{T}$ $\left(\in \mathbb{Z}^{Q\times Q}\right)$

Output: Optimized neuron distribution matrix $\boldsymbol{X}_{\mathrm{opt}}$

```
initialize X, λ1, λ2, λ3
while True do
  evaluate L_d and N_NC for X
  if L_d = N_NC then
    X_opt ← X
    break
  end if
  {X^(0), X^(1), ..., X^(Q×M)} ← α(X) // X^(0) ← X
  for i = 1 to Q × M do
    evaluate L_d^(i) for X_t^(i) // using (5) – (10)
    if L_d^(i) < L_d then
      L_d ← L_d^(i)
      X ← X^(i)
  end if
 end for
 λ1 ← λ1 + ηH(c1(X^(0)))
 λ2 ← λ2 + ηH(c2(X^(0)))
 λ3 ← λ3 + ηH(c3(X^(0)))
return X_opt
```

When considering a large SNN, the matrix calculations with $Q \times Q$ topology matrix $\boldsymbol{T}$ need prohibitive memory and calculation costs. Fortunately, a feedforward SNN allows us to use subsets of matrix $\boldsymbol{T}$ instead of the full matrix when the neurons are indexed layer-wise. For instance, the feedforward SNN in Fig. 2(a) has total O layers (each layer is indexed q_k; $1 \leq k \leq O$) each with Q_k neurons. Therefore, $Q = \sum_{k=1}^{O} Q_k$. For convenience, we introduce a new variable sum_k ($1 \leq k \leq O$) that indicates the cumulative number of neurons until the kth layer such that $sum_k = \sum_{m=1}^{k} Q_m$ and $sum_0 = 0$. The neurons in layer q_k are indexed from sum_{k-1} to sum_k-1. Therefore, in matrix $\boldsymbol{T}$, the subset for the topology of layers q_k and q_{k+1} only is $T[sum_{k-1}:(sum_k\text{-}1), sum_k:(sum_{k+1}\text{-}1)]$, which is referred to as submatrix $\boldsymbol{T}_k$ ($Q_k \times Q_{k+1}$) Note that [a: b] denotes the elements from index a to b. A schematic of submatrices in matrix $\boldsymbol{T}$ for the feedforward SNN is shown in Fig. 2(b). In this case the product $T[i,:]X[:,j]$

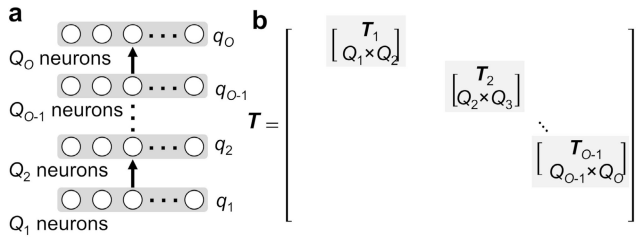


FIGURE 2. Schematic of a feedforward network with total O layers, each with Q_i neurons. (b) Submatrix T_i ($1 \le i \le O-1$) for the feedforward connection from the ith to the $(i+1)$th layer.

in (5) and (9) is modified to $T_k[i\text{-}sum_{k-1},:]X[sum_k:(sum_{k+1}\text{-}1), j]$ for k satisfying $sum_{k-1} \le i < sum_k$, improving the efficiency in the matrix multiplication.

All symbols in this study are listed in Table 1.

TABLE 1. Symbols.

Symbol	Explanation
$\boldsymbol{T}$	SNN topology matrix
$\boldsymbol{T}_i$	ith submatrix of $\boldsymbol{T}$
$\boldsymbol{X}$	Neuron distribution matrix
$\boldsymbol{P}$	Neuron-to-core connection matrix
q_i	Index of the ith layer in an SNN
Q_i	Number of neurons in the ith layer in an SNN
O	Number of layers in an SNN
sum_i	Cumulative sum of neurons until the ith layer in an SNN
Q	Total number of neurons in an SNN
s	Total number of synapses in an SNN
M	Number of neuromorphic cores
N	Maximal number of neurons per core
S	Maximal number of synapses per core
N_{NC}	Neuron-to-core connection (including self-connection) number
N_{ONC}	Output neuron-to-core connection (including self-connection) number
N_{NC1}	Neuron-to-core connection (excluding self-connection) number
N_{NC2}	Temporal average of workload of the network router
r_{dup}	Fraction of memory usage caused by data duplication
$\boldsymbol{A}$	Matrix of neurons' activity

C. ESTIMATION OF DATA DUPLICATION AND MEMORY USE

The data (presynaptic state variables) duplication caused by separate pre- and postsynaptic neurons can be evaluated using optimal matrix $\boldsymbol{X}$ given topology matrix $\boldsymbol{T}$. The number of data duplications equals the number of neuron-to-core connections except self-connections, denoted by N_{NC1}. The N_{NC1} value is evaluated as follows:

$$N_{\mathrm{NC1}} = \sum_{i=0}^{Q-1} \sum_{j=0}^{M-1} P[i,j]\,(1 - X[i,j]). \qquad (11)$$

Assuming a bit width of n_{v} for each synaptic state variable, additional memory usage due to the data duplication ($\mathrm{mem}_{\mathrm{dup}}$) is $n_{\mathrm{v}}N_{\mathrm{NC1}}$ bits. The toal memory allocated to the synaptic state variables (excluding the duplicated ones) over all cores is $n_{\mathrm{v}}Q$. Assuming that each neuron (membrane potential) and each synapse (weight) consumes n_{n}- and n_{s}-bit memory, the memory allocated to the all neurons and synapses are $n_{\mathrm{n}}Q$ and $n_{\mathrm{s}}s$, respectively. Adding them up yields the memory usage (mem_0) that is not subject to optimization and generally occupies the majority of memory capacity. The ratio of $\mathrm{mem}_{\mathrm{dup}}$ to mem_0 (r_{dup}) is

$$r_{dup} = n_v N_{NC1} / [(n_n + n_v)\,Q + n_s s]\,.$$

Although there are other data required for membrane potential and synaptic weight updates, they are often shared among all neurons and synapses in the same core instead of being assigned to each neuron and synapse. Therefore, their fraction is negligible.

For simplicity, we assume that all variables are given the same bit width ($n_{\mathrm{v}} = n_{\mathrm{n}} = n_{\mathrm{s}}$) unless otherwise stated.

III. OPTIMIZATION RESULTS

Different learning algorithms base weight update on different data. The data that are duplicated for the learner to access depend on learning algorithm. Here, we optimized the spatial distribution of neurons for STDP [13], [14] and eRBP [16], which are renowned two- and three-factor learning rules, respectively. The algorithm was implemented in Python on a workstation (CPU: Intel Xeon Silver 4110 2.10GHz, GPU: Titan RTX). The codes to run on CPU and GPU are available on-line (https://github.com/guhyunkim/Optimize-neuron-distribution). All optimization results are summarized in Table 2.

A. SPIKE TIMING-DEPENDENT PLASTICITY RULE

The simple STDP rule is an event-driven learning algorithm of locality, which bases the direction of a weight change on the temporal order of pre- and postsynaptic events [13], [14] The weight change is determined by the temporal distance between pre- and postsynaptic events, which decays exponentially with the temporal distance. This rule is an event-driven algorithm because the weight is ad hoc updated upon both pre- and postsynaptic events, i.e., a fan-in synaptic weight is updated when the post- or the presynaptic neuron fires a spike. Each neuron is given a state variable that reads low-pass filtered spikes. A schematic of the simple STDP rule for a pair of pre- and postsynaptic neurons in Fig. 3(a) is shown in Fig. 3(b). Consider pre- and postsynaptic neurons placed in two different cores. When the postsynaptic neuron fires, the presynaptic state variable is read to determine the weight change (potentiation), whereas when the presynaptic neuron fires, the postsynaptic variable is addressed to determine the degree of depression. In case of depression, the learner has the direct access to the current fan-in synaptic weight and postsynaptic state variable in the same core. However, in case of potentiation, the learner needs the presynaptic state variable in the different core; therefore, the presynaptic

TABLE 2. Optimization results.

	Network	M	N	S	Objective function	Optimal connection	Random connection	Optimization time (s)
STDP								
Case 1	64-64 FCFFN	4	128	4096	N_{NC}	N_{NC}: 64 r_{dup}: 0	$<N_{NC}>$: 256 $<r_{dup}>$: 0.0442	0.95
Case 2		4	40	1500	N_{NC}	N_{NC}: 192 r_{dup}: 0.0386	$<N_{NC}>$: 256 $<r_{dup}>$: 0.0442	0.97
Case 3	1024-256-64-16 FCFFN	16	128	32768	N_{NC}	N_{NC}: 9536 r_{dup}: 0.033	$<N_{NC}>$: 21074 $<r_{dup}>$: 0.07	1623
Case 4	2048 recurrent net	16	256	4096	N_{NC}	N_{NC}: 15765 r_{dup}: 0.311	$<N_{NC}>$: 23648 $<r_{dup}>$: 0.481	951
Case 7	1024-256-64-16 FCFFN	16	128	32768	N_{NC1}	N_{NC1}: 7680 r_{dup}: 0.0272	$<N_{NC1}>$: 19757 $<r_{dup}>$: 0.07	1657
Case 9	1024-256-64-16 FCFFN	16	128	32768	N_{NC2}	N_{NC2}:14148 r_{dup}: 0.0331	$<N_{NC2}>$: 42701 $<r_{dup}>$: 0.07	2385
eRBP								
Case 5	1024-256-64-16 FCFFN with feedback	16	128	32768	N_{NC}	N_{NC}: 9680 N_{ONC}: 144 r_{dup}: 4.45×10^{-4}	$<N_{NC}>$: 21581 $<N_{ONC}>$: 256 $<r_{dup}>$: 8.35×10^{-4}	1661
Case 6					N_{ONC}	N_{NC}: 9808 N_{ONC}: 144 r_{dup}: 4.45×10^{-4}	$<N_{NC}>$: 21581 $<N_{ONC}>$: 256 $<r_{dup}>$: 8.35×10^{-4}	289
Case 8					N_{NC1}	N_{NC1}: 8720 r_{dup}: 4.45×10^{-4}	$<N_{NC1}>$: 20347 $<r_{dup}>$: 8.35×10^{-4}	1679

state variable should be duplicated in the core with the fan-in synapse and postsynaptic neuron.

For the STDP rule, the number of neuron-to-core connections (N_{NC}) is proportional to the number of state variable duplications, so that the duplication number is minimized by minimizing N_{NC}. We applied the optimization method to the following four cases.

Case 1: *Single-layer SNN over cores* ($N \geq Q; S \geq s$). We first optimized the distribution of neurons in a single-layer SNN over four neuromorphic cores ($M = 4$). The SNN consists of 64 input neurons and 64 output neurons (64-64 network). Note that a Q_1-Q_2-…-Q_O network means a fully connected feedforward network (FCFFN) of total O layers, each with Q_i neurons. Unless otherwise stated, the FCFFN in this study is not given feedback connections. Each core accommodates maximal 128 neurons and 4,096 synapses ($N = 128; S = 4,096$). Fig. 3(c) shows the changes in N_{NC}(grey dots) and L_d (red dots) over iteration. On the 695th step, N_{NC} became the same as L_d, satisfying the constraints (6), (7), and (8); consequently, the iteration terminated. Fig. 3(d) points to N_{NC} minimized to 64. The distribution of neurons over four cores is shown in Fig. 3(e); all 128 neurons are located in the identical core, which is the optimal distribution to minimize N_{NC} given the sufficient capacity of a single core ($N = 128; S = 4,096$) to accommodate all neurons and synapses. Note that because N_{NC} includes self-connections, the value does not fall to zero. For comparison, we evaluated the average N_{NC} value ($< N_{NC} >$) for random distribution of 128 neurons in the 64-64 network over the four cores, which was averaged on 5,000 trials. The $< N_{NC} >$ and Q values are co-plotted in Fig. 3(d).

Case 2: *Single-layer SNN over cores* ($N < Q; S < s$). Another task is to distribute the same network (64-64) over four cores with insufficient core capacity to map all neurons and synapses onto a single core ($M = 4; N = 40; S = 1,500$). The changes in N_{NC} and L_d with iteration number are shown in Fig. 3(f); the optimal distribution was acquired on the 709th iteration step, yielding an N_{NC} of 192. For the optimal distribution, the N_{NC} value and the distribution over four cores are shown in Fig. 3(g) and (h), respectively. Unlike the previous case, the insufficient capacity of each core causes the 128 neurons distributed over all four cores. Fig. 3(g) also shows the $< N_{NC} >$ value for random distribution of the neurons; its comparison with the optimal N_{NC} indicates a decrease in N_{NC} by 25%, implying a 25% decrease in neuron-to-core communication and 12.5% decrease in data duplication in r_{dup}.

Case 3: *Multilayer SNN over cores* ($N < Q; S < s$). A 1024-256-64-16 FCFFN was distributed over 16 cores

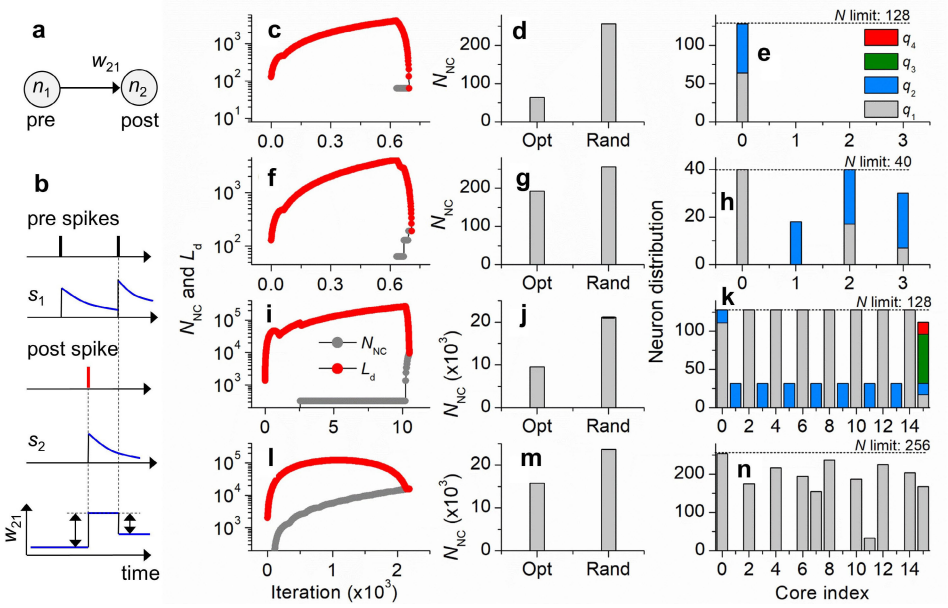


FIGURE 3. Optimal distribution of neurons over the given number of cores, each with an STDP learner. (a) Topology of a presynaptic neuron (n_1) and postsynaptic neuron (n_2). (b) Weight (w_{21}) change upon a presynaptic spike and two postsynaptic spikes considering the state variables s_1 and s_2 for neurons n_1 and n_2, respectively. Case 1 (64-64 network; $M = 4$; $N = 128$; $S = 4,096$): (c) changes in the number of neuron-to-core connections N_{NC} and discrete Lagrangian function L_d with iteration. When optimal (satisfying the constraints), the two function values are equal. (d) Optimal N_{NC} value in comparison with the average value for random distributions. (e) Optimal distribution of neurons over four cores. Case 2 (64-64 network; $M = 4$; $N = 40$; $S = 1,500$): the same set of data is given in (f), (g), and (h) as for Case 1. Case 3 (1024-256-64-16 network; $M = 16$; $N = 128$; $S = 32,768$): (i), (j), and (k). Case 4 (random recurrent network; $Q = 2,048$; $M = 16$; $N = 256$; $S = 4,096$): (l), (m), and (n). Missing N_{NC} data points in (c), (f), and (i) indicate that $N_{NC} = 0$.

($M = 16$), each with $N = 128$ and $S = 32,768$. The optimization process is plotted in Fig. 3(i); the process is completed on the 10,520th iteration step, resulting in an N_{NC} of 9,536. The optimization reduces the N_{NC} value by 55% compared with the $< N_{NC} >$ value on 5,000 trials (Fig. 3(j)). The distribution is shown in Fig. 3(k). Additionally, the optimization reduces the data duplication in r_{dup} by 52.8%.

Case 4: *SNN with random sparse connections* ($N < Q$; $S < s$). This network is equivalent to a recurrent SNN ($Q = 2,048$). The connections within the network were randomly given at a probability of 0.01; the average number of synapses $< s >$ is approximately 41,900 because $< s > \approx 0.01Q^2$. This network was mapped onto 16 cores ($M = 16$), each with $N = 256$ and $S = 4,096$. The optimization process and optimal distribution over the 16 cores are shown in Fig. 3(l) and (n), respectively. Compared with the $< N_{NC} >$ value, the optimization process reduces the N_{NC} value by 44% (Fig. 3(m)) and the r_{dup} value by 35.5%.

B. EVENT-DRIVEN RANDOM BACKPROPAGATION ALGORITHM

The eRBP is an example of a three-factor learning rule that bases the weight change on (i) presynaptic event, (ii) postsynaptic state variable, and (iii) sum of feedback error signals[16]. Here, the postsynaptic state variable (factor ii) is the weighted sum of all inputs from its presynaptic neurons. A topology of pre- and postsynaptic neurons with an error signaling terminal is illustrated in Fig. 4(a). Fig. 4(b) shows a schematic of the eRBP algorithm considering the three factors. Unlike the STDP rule, in the eRBP algorithm, the weight is updated upon a presynaptic event only. Therefore, the presynaptic variable is unnecessarily duplicated in the core of the postsynaptic neuron. Instead, the sum of feedback error signals (factor iii) from the output neurons is duplicated in the associated cores. In this regard, the choice of the objective function f in (2) is twofold: (**Case 5**) the total number of neuron-to-core connections including both feed-forward and feed-back connections as in the previous section (N_{NC}) and (**Case 6**) the number of output neuron-to-core connections only (N_{ONC}).

We chose a 1024-256-64-16 FCFFN with full feedback connection (output layer fully connected to all hidden layers) to optimize its distribution over multicores ($M = 16$; $N = 128$; $S = 32,768$). The feedback connections were considered in topology matrix $\boldsymbol{T}$. The optimization process for **Case 5** ($f = N_{NC}$) is shown in Fig. 4(c). The optimal N_{NC} value is below 45% of the average N_{NC} for random distributions of neurons over 16 cores as shown in Fig. 4(d). The optimal distribution of neurons is plotted in Fig. 4(e). The optimization process with another objection function f ($= N_{ONC}$) (**Case 6**) yielded the evolution of optimal N_{ONC} (144) and L_d with iteration as plotted in Fig. 4(f). Note that

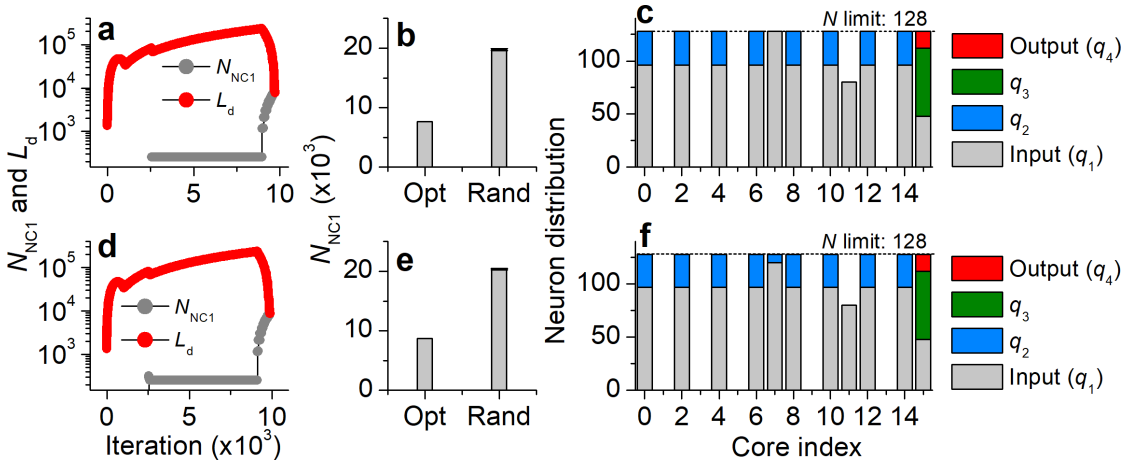


FIGURE 4. Optimal distribution of neurons over the given number of cores, each with an eRBP learner. (a) Topology of a presynaptic neuron (n_1) and postsynaptic neuron (n_2) with an error signaling terminal. (b) Weight (w_{21}) change upon a presynaptic spike considering the three factors. A 1024-256-64-16 feedforward network was mapped optimally onto 16 cores ($M = 16$; $N = 128$; $S = 32,768$) with an objective function of neuron-to-core connection number N_{NC}. (c) Changes in N_{NC} and L_d with iteration. (d) Optimal N_{NC} value compared with average N_{NC} value for random distributions. (e) Optimal distribution of the neurons over 16 cores. The same network was mapped onto the same cores with an objective function of output neuron-to-core connection number N_{ONC}. For this case, the data corresponding to (c), (d), and (e) are shown in (f), (g), and (h), respectively. Missing N_{NC} data points in (c) and (f) indicate that $N_{NC} = 0$.

update rate η was set to 0.01 for **Case 6**. The optimal N_{ONC} value compared with $< N_{ONC} >$ for random distributions of neurons is shown in Fig. 4(g). Fig. 4(h) shows the optimal distribution of neurons over 16 cores. It is noteworthy that the distribution in Fig. 4(h) is considerably similar to that in Fig. 4(e). Specifically, the final N_{NC} value in **Case 6** is 9,808, which is comparable to the optimal N_{NC} value (9,680) in **Case 5**. In addition, the optimal N_{ONC} value in **Case 6** is 144, which is equal to the final N_{ONC} value in **Case 6**. This is because either objective function leads to the neurons (in the same layer) congregated over the least number of cores.

IV. DISCUSSION

Different multicore neuromorphic processors differ for their architecture, and thus the model architecture assumed in this study needs to be modified accordingly. Given the diversity in architecture, it is prohibitive to apply the optimization method to every architecture. Instead, we consider another commonplace architecture of multicore processors, which needs a different objective function. Frequently, multicore neuromorphic processors are given local channels for local event routing when a pre- and postsynaptic neuron are placed in the same core [1], [21] In this case, local events barely impose workloads on the network router that is dedicated to inter-core routing only; therefore, an appropriate objective function is the number of neuron-to-core connections other than self-connections N_{NC1} in (11). Because data duplication for the learner is proportional to this connection number, optimizing this number also minimizes data duplication.

With the same constraints as (7), (8), and (9), the discrete LMM optimally distributes the neurons over cores with (**Case 7**) the STDP and (**Case 8**) eRBP learner. Fig. 5(a), (b), and (c) show the optimization for a 1024-256-64-16 FCFFN embedding the STDP learner ($M = 16; N = 128; S = 32,768$). The optimal N_{NC1} value is 7,680 (39% of the average N_{NC1} value on 5,000 trials). Recalling that the optimal N_{NC} for the same case was 9,472, the optimal N_{NC1} value is smaller than the N_{NC} because self-connections are excluded. The same objective function successfully applied to a 1024-256-64-12 SNN with feedback connections to incorporate the eRBP leaner into the core as for the network in Section III.B (Fig. 5(d), (e), and (f)). All data are listed in Table 2.

Equal activities of all neurons underlie the choice the objective functions N_{NC}, N_{ONC}, and N_{NC1}. Practically, neurons vary in activity, and neurons with high activities give more workloads to network routers than those with low activities. Therefore, inhomogeneous neuronal activity should be considered to optimize neuron-to-core connections to minimize the workload of the network router. Because neuronal activity is the temporal average of spikes, the temporal average of the workload is proportional to the activity. We introduce another objective function N_{NC2}, implying the temporal average of the workload given a neuronal activity vector $\boldsymbol{A}$, where $A[i]$ indicates the activity of neuron i. The objective function N_{NC2} is defined as

$$N_{NC2} = \sum_{i=0}^{Q-1} \sum_{j=0}^{M-1} P[i,j]\,A[i].$$

With this objective function and the constraints (7), (8), and (9), the discrete LMM optimally distributes the inhomogeneous neurons over cores. We applied the objective function N_{NC2} to a 1024-256-64-12 FCFFN with the STDP learner (**Case 9**; $M = 16; N = 128; S = 32768$). Half the neurons in each layer were given an activity of two, whereas the others were given an activity of one. The results are

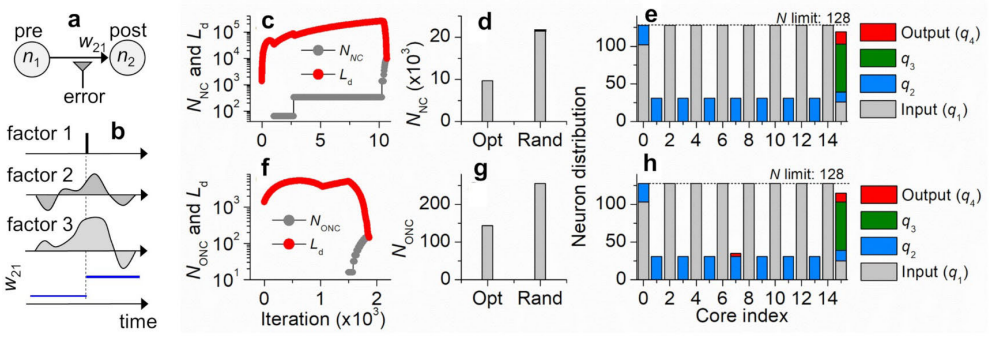


FIGURE 5. Optimization with the number of neuron-to-core connections except self-connections (N_{NC1}) as an objective function. (a) Optimization procedure of distribution of neurons in a 1024-256-64-16 FCFFN over 16 cores ($M = 16$; $N = 128$; $S = 32,768$), each with the STDP learner. (b) Optimal N_{NC1} value compared with the average N_{NC1} value for random distributions. (c) Optimal distribution of the neurons over 16 cores. A 1024-256-64-16 FCFFN with full feedback connection was mapped onto the same cores, each with the eRBP learner. For this case, the data corresponding to (a), (b), and (c) are shown in (d), (e), and (f), respectively. Missing N_{NC} data points in (a) and (d) indicate that $N_{NC} = 0$.

shown in Fig. 6 and Table 2. The reduction in N_{NC2} by the optimization exceeds 66%, implying a 66% reduction in the workload of the router.

The data duplication metric r_{dup} is a relative quantity to the major memory usage (mem_0)—mostly for synaptic weights. For the STDP learner (**Cases 1**, **2**, **3**, **4**, **7**, **9**), the reduction in data duplication and memory usage in absolute scale varies within a range of $0.0055mem_0$ – $0.17mem_0$, depending on network topology (Table 2). Notably, the recurrent network (**Case 4**) takes the largest advantage of optimization in terms of memory saving (reduction in memory usage by 17% of mem_0), while a reduction by approximately 4% of mem_0 is expected for the FCFFN. The large r_{dup} for **Case 4** is due to a small mem_0 value arising from sparse synaptic connections. It should be noted that the estimation assumes the same bit-width ($n_v = n_n = n_s$). The reduction in memory usage by the neuronal distribution optimization becomes more obvious by decreasing the precision of synaptic weight, e.g., $n_v = n_n = 2n_s$ and $n_v = n_n = 4n_s$ (Fig. 7(a)).

Compared with the STDP, the eRBP algorithm inherently needs negligible data duplication. This is because the total number of feedback connections is much lower than that of feedforward connections ($q_4 \times (q_2 + q_3) \ll q_1q_2 + q_2q_3 + q_3q_4$). Therefore, the reduction in memory usage is negligible irrespective of the precision of synaptic weight as shown in Fig. 7(b).

There have been a number of learning rules proposed to date besides the two rules addressed in our study. They vary in complexity, e.g., the number of variables and the locality of variables (global or local). Particularly, they differ in the goal to achieve; accordingly, the rules can be categorized as two groups (Groups 1 and 2) by and large. The rules in Group 1 aim at reproducing biological synaptic plasticity behaviors in activity and temporal domains, e.g., STDP [13], [14], [22] and Bienenstock-Cooper-Munro rule [23], [24]. The vigorous studies that have been conducted over the past decades have enriched the diversity in model to various degrees of biological plausibility [25] As a rule of thumb, the reproducibility tends to increase with the rule complexity. Thus, high reproducibility as a performance metric needs large hardware resources.

The rules in Group 2 are rather oriented towards high-level functionalities, such as classification, prediction, and recognition, which are currently dominated by deep learning. Although attempts to seek the biological plausibility of these rules are still made, the accuracy of such high-level functionalities is a prior metric of performance. This trend in rule development is relatively new so that the master rule that is universally applicable to domain-specific SNNs has not been found yet. However, recent studies have proposed several feasible methods to train SNNs using real-world datasets [16], [26]–[29]

As stated in Introduction, learning rules that are potentially embedded in neuromorphic processors need to meet the constraints imposed by neuromorphic hardware design. In this regard, the spike-dependent synaptic plasticity (SDSP) rule satisfies the constraints [30] The SDSP rule is a presynaptic event-driven learning rule with two key postsynaptic variables: membrane potential and calcium concentration. The calcium concentration is a low-pass filtered postsynaptic spikes. The direction of weight change is determined by these neuronal variables when a presynaptic spike arrives at the postsynaptic neuron. Because no data in a source core needs to be duplicated in other cores, this learner is free from any data duplication issues. A modified SDSP learner was successfully embedded in a recent neuromorphic prototype (MorphIC) with four cores [5].

The last issue is to map a biologically plausible massive SNN onto the multicore architecture of concern in this study. Such an SNN includes large numbers of fan-in and fan-out synapses, which amount to approximately 4,000 synapses per neuron on average [31] Recently, a real-time simulation of a massive SNN has successfully been done on the SpiNNaker neuromorphic hardware, which is flexible in the distribution

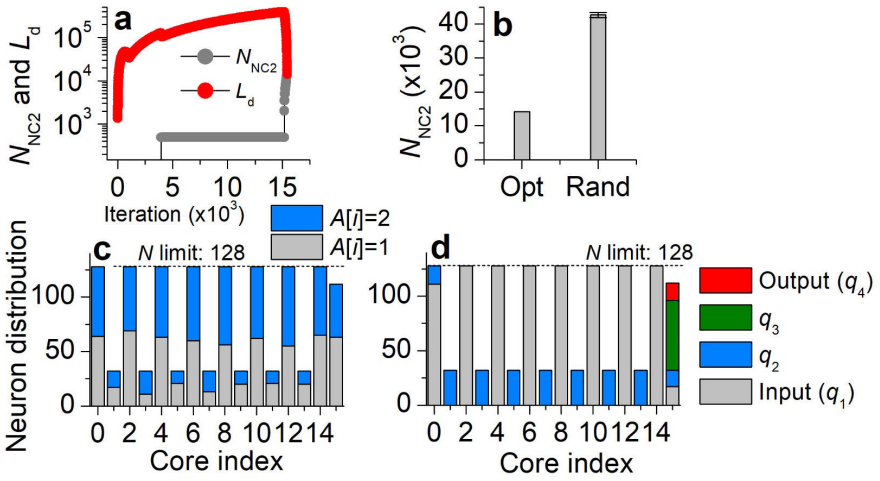


FIGURE 6. Optimization with the number of neuron-to-core communications (N_{NC2}) as an objective function. (a) Optimization procedure of distribution of neurons in a 1024-256-64-12 FCFFN over 16 cores ($M = 16$; $N = 128$; $S = 32768$), each with the STDP learner. (b) Optimal N_{NC2} value compared with the average N_{NC2} value for random distributions. Optimal distribution of the neurons over 16 cores, sorted according to (c) neuronal activity and (d) layer index. Missing N_{NC2} data points in (a) indicate that $N_{NC2} = 0$.

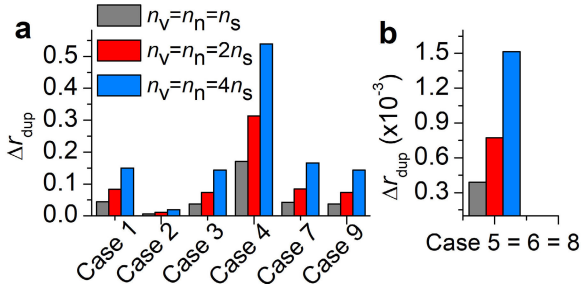


FIGURE 7. Reduction in total memory usage by the optimization of neuronal distribution for (a) the STDP (Cases 1, 2, 3, 4, 7, and 9) and (b) eRBP (Cases 5, 6, and 8). The reduction was evaluated for different precisions of synaptic weight.

of neurons and synapses [32] The hypothetical architecture of our concern may not be able to host such a massive SNN mainly because of the limited on-core memory capacity. One of the key assumptions on neuronal and synaptic distributions is that neurons and their fan-in synapses are placed in the same core, which is undermined when each neuron has a large number of fan-in synapses beyond the capacity of the core S. A feasible workaround may be to employ a multi-compartment neuron (rather than point neuron) model. Because a single neuron consists of multiple compartments (dendrites), the total fan-in synapses of a neuron are distributed over the compartments. Given the limited number of synapses per core S, the compartments are distributed over multiple cores with their fan-in synapses. In this method, a challenge is communication between a somatic compartment and multiple dendritic compartments (separate from the somatic one). Event data packets are the only data allowed in inter-core communication; therefore, sending dendritic potential data to the somatic compartment may be impossible. An indirect way to let the soma know the dendritic potential is to send a dendritic spike to the soma. Then, the soma can evaluate the dendritic potential (= preset threshold for dendritic spiking), which decays out subsequently at a given somatic time constant. The same holds for the other dendrites; therefore, the soma can integrate all dendritic potentials in time.

V. CONCLUSION

The success in optimal distribution of neuron over cores in a multicore neuromorphic processor demonstrated the discrete LMM as a simple solution to a large reduction in event routing congestion, which is a critical obstacle to scaling up SNN. The discrete LMM employed the number of neuron-to-core connections as an objective function with several realistic constraints: no duplication of neuron in multiple cores, and the limited numbers of neurons and synapses per core. When a multicore neuromorphic processor embedding the STDP or eRBP learner was optimized, a 55% reduction in the number of neuron-to-core connections was achieved in **Case 3** (STDP learner) and **Case 5** (eRBP learner) compared with random distribution of neurons over cores. This reduces the workload on the network router and data duplication by 55%. Although the objective function and constraints may need modifications depending on the specific processor architecture, the discrete LMM likely offers a robust solution to the optimal distribution of neurons as shown for a different architecture with local event routing channels.

REFERENCES

[1] V. Kornijcuk, J. Park, G. Kim, D. Kim, I. Kim, J. Kim, J. Y. Kwak, and D. S. Jeong, "Reconfigurable spike routing architectures for on-chip local learning in neuromorphic systems," *Adv. Mater. Technol.*, vol. 4, no. 1, Jan. 2019, Art. no. 1800345.

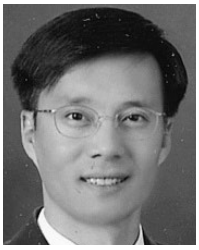
[2] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[3] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.
[4] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.
[5] C. Frenkel, J.-D. Legat, and D. Bol, "MorphIC: A 65-nm 738 k-Synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 5, pp. 999–1010, Jul. 2019.
[6] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb. 2018.
[7] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1947–1950.
[8] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Trans. Neural Netw.*, vol. 4, no. 3, pp. 523–528, May 1993.
[9] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.
[10] E. O. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, vol. 5, pp. 52–68, Jul. 2018.
[11] P. Dayan and L. F. Abbott, *Theoretical Neuroscience*. London, U.K.: MIT Press, 2001.
[12] D. O. Hebb, *The Organization of Behavior*. New York, NY, USA: Wiley, 1949.
[13] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: Taming the beast," *Nature Neurosci.*, vol. 3, no. S11, pp. 1178–1183, Nov. 2000.
[14] N. Caporale and Y. Dan, "Spike timing-dependent plasticity: A Hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, Jul. 2008.
[15] Ł. Kuśmierz, T. Isomura, and T. Toyoizumi, "Learning with three factors: Modulating Hebbian plasticity with errors," *Current Opinion Neurobiol.*, vol. 46, pp. 170–177, Oct. 2017.
[16] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017.
[17] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York, NY, USA: Academic, 2014.
[18] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. Berlin, Germany: Springer, 1984.
[19] B. W. Wah and Z. Wu, "The theory of discrete Lagrange multipliers for nonlinear discrete optimization," in *Proc. Int. Conf. Princ. Pract. Constraint Program.* Berlin, Germany: Springer, 1999, pp. 28–42.
[20] V. Kornijcuk and D. S. Jeong, "Recent progress in real-time adaptable digital neuromorphic hardware," *Adv. Intell. Syst.*, vol. 1, no. 6, Oct. 2019, Art. no. 1900030.
[21] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.
[22] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998.
[23] E. Bienenstock, L. Cooper, and P. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *J. Neurosci.*, vol. 2, no. 1, pp. 32–48, Jan. 1982.
[24] L. N. Cooper and M. F. Bear, "The BCM theory of synapse modification at 30: Interaction of theory with experiment," *Nature Rev. Neurosci.*, vol. 13, no. 11, pp. 798–810, Nov. 2012.
[25] V. Kornijcuk, D. Kim, G. Kim, and D. S. Jeong, "Simplified calcium signaling cascade for synaptic plasticity," *Neural Netw.*, vol. 123, pp. 38–51, Mar. 2020.
[26] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," 2019, *arXiv:1901.09948*. [Online]. Available: http://arxiv.org/abs/1901.09948
[27] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," 2018, *arXiv:1803.09574*. [Online]. Available: http://arxiv.org/abs/1803.09574
[28] J. Kaiser, A. Friedrich, J. C. V. Tieck, D. Reichard, A. Roennau, E. Neftci, and R. Dillmann, "Embodied neuromorphic vision with event-driven random backpropagation," 2019, *arXiv:1904.04805*. [Online]. Available: http://arxiv.org/abs/1904.04805
[29] P. Tak Peter Tang, T.-H. Lin, and M. Davies, "Sparse coding by spiking neural networks: Convergence theory and computational results," 2017, *arXiv:1705.05475*. [Online]. Available: http://arxiv.org/abs/1705.05475
[30] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, Nov. 2007.
[31] T. C. Potjans and M. Diesmann, "The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model," *Cerebral Cortex*, vol. 24, no. 3, pp. 785–806, Mar. 2014.
[32] O. Rhodes, L. Peres, A. G. D. Rowley, A. Gait, L. A. Plana, C. Brenninkmeijer, and S. B. Furber, "Real-time cortical simulation on neuromorphic hardware," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2164, Feb. 2020, Art. no. 20190160.

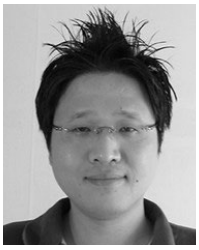
GUHYUN KIM received the B.S. and Ph.D. degrees in material science and engineering from Seoul National University, Seoul, South Korea, in 2015 and 2020, respectively, He is currently a Postdoctoral Scholar with Hanyang University, Seoul. His research interests include learning algorithms for spiking neural networks and deep neural networks.

VLADIMIR KORNIJCUK received the B.S. degree in telecommunication physics and electronics from Vilnius University, Vilnius, Lithuania, the M.S. degree in materials science and engineering from the Seoul National University of Science and Technology, Seoul, South Korea, and the Ph.D. degree in nano and information technology from the University of Science and Technology, Seoul, in 2018. He is currently with SK Hynix, South Korea. His current research interest includes digital neuromorphic processor design.

JEESON KIM received the B.Eng. degree in semiconductor systems engineering from the Catholic University of Korea, South Korea, in 2011, and the M.S. and Ph.D. degrees from RMIT University, Australia, in 2014 and 2019, respectively. She is currently a Postdoctoral Scholar with Hanyang University, South Korea. Her research interests include nano-enabled hardware security, low-power solutions for cyber-physical systems, and analog and digital designs for neuromorphic systems.



CHEOL SEONG HWANG received the Ph.D. degree from Seoul National University, Seoul, South Korea, in 1993. Since 1998, he has been a Professor with the Department of Materials Science and Engineering, Seoul National University. His current research interests include high-k gate oxides, dynamic random access memory capacitors, new memory devices, including resistive RAM devices, and ferroelectric materials and devices, energy storage capacitors, and neuromorphic computing.



DOO SEOK JEONG (Member, IEEE) received the B.E. and M.E. degrees in materials science from Seoul National University, in 2002 and 2005, respectively, and the Ph.D. degree in materials science from RWTH Aachen, Germany, in 2008. He was with the Korea Institute of Science and Technology, from 2008 to 2018. He is an Associate Professor with Hanyang University, South Korea. His research interest includes spiking neural networks for real-time learning, ranging from building blocks to learning algorithms.