Neurocomputing 272 (2018) 154–162

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Training deep neural networks with discrete state transition

Guoqi Li[a,1,*], Lei Deng[a,1], Lei Tian[a], Haotian Cui[b], Wentao Han[c], Jing Pei[a], Luping Shi[a]

[a] Department of Precision Instrument, Center for Brain Inspired Computing Research, Tsinghua University, Beijing, 100084, China
[b] Department of Biomedical Engineering, Tsinghua University, Beijing, 100084, China
[c] Department of Computer Science, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 9 August 2016
Revised 26 January 2017
Accepted 26 June 2017
Available online 4 July 2017

Communicated by Bo Shen

Keywords:
Deep learning
Neural network applications
Discrete state transition
Discrete weight space

ABSTRACT

Deep neural networks have been achieving booming breakthroughs in various artificial intelligence tasks, however they are notorious for consuming unbearable hardware resources, training time and power. The emerging pruning/binarization methods, which aim at both decreasing overheads and retaining high performance, seem to promise applications on portable devices. However, even with these most advanced algorithms, we have to save the full-precision weights during the gradient descent process which remains size and power bottlenecks of memory access and the resulting computation. To address this challenge, we propose a unified ***discrete state transition (DST)*** framework by introducing a probabilistic projection operator that constrains the weight matrices in a discrete weight space (DWS) with configurable number of states, throughout the whole training process. The experimental results over various data sets including MNIST, CIFAR10 and SVHN show the effectiveness of this framework. The direct transition between discrete states significantly saves memory for storing weights in full precision, as well as simplifies the computation of weight updating. The proposed DST framework is hardware friendly as it can be easily implemented by a wide range of emerging portable devices, including binary, ternary and multiple-level memory devices. This work paves the way for on-chip learning on various portable devices in the near future.

1. Introduction

In the past few years, big datasets, various learning models and GPUs have driven deep neural networks (DNNs) to achieve best results one after another in a wide range of artificial intelligence (AI) tasks, including computer vision [1–4], speech recognition [5–7], natural language processing [8–10], and the incredible success of the AlphaGo [11]. In addition, the applications of neural networks or deep neural networks have also been extended to other disciplines such as physical and chemical problems [12–18]. However, behind the appealing demonstrations, the overheads of hardware resources, training time and power consumption are terribly expensive. As a result, GPUs-based deep learning system is difficult to be embedded into portable devices. On the other side, parallel to the progress of deep learning algorithms, a variety of specialized portable devices for AI applications are emerging with advantages of compact size, fast speed and power-efficient computation [19,19–26], but they usually suffer from severe performance loss when carrying out tasks. In general, high performance and low overheads have formed a contradictory pair, and it requires more fundamental insights from the top-down perspective.

Researchers have proposed a lot of new methods to address the above issue. Pruning the pre-trained neural networks to reduce the parameters often alleviates the computational burden at inference time [27–30]. Designing compact layers [31–33] or sparse connections [34,35] also simplifies the computation. Other efforts focus on quantizing parameters and then replacing the real-valued multiplication of the weight and activation by integer shifts [36–39]. Whereas, performance loss still remains the major disadvantage of these techniques [40–42].

Last year, several interesting reports on binary DNNs seem to balance the contradiction of performance and overheads well [39,43,44]. By constraining the weights (or together with the activations) to binary states $\{-1, 1\}$ when computing (non-updating) in both forward and backward passes, the complex float multiplications and accumulations tend to be very simple logic operations, such as XNOR and bit-count. The nearly state-of-the-art classification accuracy has been achieved over the MNIST, CIFAR10, SVHN datasets.

To take this idea further, [45] introduces a shared scaling factor in order to deterministically binarize the parameters or make some

* Corresponding author.
E-mail addresses: liguoqi@mail.tsinghua.edu.cn (G. Li), peij@mail.tsinghua.edu.cn (J. Pei), lpshi@mail.tsinghua.edu.cn (L. Shi).
[1] Contribute equally.

http://dx.doi.org/10.1016/j.neucom.2017.06.058
0925-2312/

changes for the block order in DNNs. By this way, comparable performance on large datasets, such as ImageNet, has been achieved. Mostly recently, ternary networks [46–48] have also attracted considerable attention. However, challenge remains. In particular, even with the most advanced binarization method, it is unavoidable to save the full-precision weights for gradient descent that involves high memory cost and floating-point calculation, as well as extra power for frequent access of the external memory. Furthermore, it is reported that some novel multi-level memory-based portable devices support more than two states [49,50], which indicates the mentioned binary algorithms cannot fully utilize their capability. Thus, how to train DNNs by always constraining the weights in a discrete weight space (DWS) with multi-level states is yet to be investigated.

In this work, we propose an algorithm named ***discrete state transition (DST)*** by introducing a probabilistic projection operator. Specifically, the probabilistic projection operator is used to constrain the weights in a discrete weight space (DWS) when training DNNs, and the state transition is applied to guarantee that the next state will never jump out the space throughout the whole training process. By this way, we build a new framework regarding direct training DNNs with discrete state transition. The salient features of our framework are summarized as follows:

1. The hidden continuous weight update in full-precision space and repeated sampling to discrete space in existing schemes turns to be direct transitions among discrete states via DST, which doesn't require external memory for storing full-precision weights. This significantly reduces the memory cost and breaks the computation bottleneck in the training process, and promises on-chip learning.
2. Although the training process is always restricted in DWS, our method still obtains comparable performance with state-of-the-art algorithms over the MNIST, CIFAR10 and SVHN data sets.
3. The number of intermediate states is configurable to be compatible with multifarious multi-level memory devices (e.g., various RRAM/PCRAM or memristive devices [49–51]), not limited to binary state, which may take full advantages of these devices' capability. This makes our proposed DST framework more hardware friendly.

To the best of our knowledge, this is the first work that enables the states jumping within a DWS during the whole training process such that the external full-precision memory can be cut down. This really paves the way toward on-chip learning on various portable devices in the near future.

2. Problem formulation

Without loss of generality, assume that we train the weights/states in DNNs in the interval $[-L, L]$. The weight space (DWS) Z_N, by which the weights are constrained, is defined by

$$Z_N = \left\{ z_N^n | z_N^n = \left(\frac{n}{2^{N-1}} - 1 \right) L, \quad n = 0, 1, \ldots, 2^N \right\}, \tag{1}$$

where N is a given non-negative integer $N \in \mathbf{Z}^+$, i.e., $N = 0, 1, 2, \ldots$ and so on. It is shown that, for a given N, the total number of states is $2^N + 1$ and $\Delta z_N = \frac{L}{2^{N-1}}$ is the distance between adjacent states. In contrast to DWS, we define a continuous weight space (CWS) where its element can be an arbitrary full-precision value, and a binary weight space (BWS) where its element belongs to $\{-1, 1\} \times L$. Note that the DWS becomes CWS when $N \to \infty$, and degrades to BWS when $N = 0$.

We further denote that $L_n = \{x | z_N^n \le x < z_N^{n+1}\}$ for $n = 0, 1, \ldots, z^N - 2$, and $L_{2^N-1} = \{x | z_N^{2^N-1} \le x \le z_N^{2^N}\}$. Then, $L_i \cap L_j = \emptyset$ for $i \neq j$ and $\cup_{n=0}^{2^N-1} L_n = [-L, L]$. For a random variable w_i under uniform distribution, that is to say $w_i \in U[-L, L]$, there exist a unique n such that $w_i \in L_n$. Now we consider how to project a random variable $w_i \in U[-L, L]$ to Z_N, i.e., how to project w_i to the discrete states (z_N^n or z_N^{n+1}) with following density function:

$$P(w_i = z_N^n | w_i \in L_n) = \frac{\left| \Delta z_N - |z_N^n - w_i| \right|}{\Delta z_N}, 0 \le n \le 2^N - 1 \tag{2}$$

and

$$\begin{aligned} P\left(w_i = z_N^{n+1} | w_i \in L_n\right) &= 1 - P(w_i = z_N^n | w_i \in L_n) \\ &= \frac{\left| \Delta z_N - |z_N^{n+1} - w_i| \right|}{\Delta z_N}, 0 \le n \le 2^N - 1. \end{aligned} \tag{3}$$

As shown in Fig. 1, the range of DWS is $[-L, L]$ with $(2^N + 1)$ states, and Δz_N represents the distance between adjacent states. Each state is only possible to be transferred to either $z_N^{m+\kappa_i}$ or the $z_N^{m+\kappa_i+sign(\kappa_i)}$, where κ_i and ν_i are determined by ΔW_i^k. The transition probability is $1 - \tau(\nu_i)$ and $\boldsymbol{\tau}(\nu_i)$, respectively. The detailed projection method will be described in the following.

Now we show that the above density function in (2) and (3) is a probability density function.

Lemma 1. *For any random variable $w_i \in U[-L, L]$ where $U[-L, L]$ denotes a uniform distribution in the interval $[-L, L]$, we have*

$$\sum_{n=0}^{2^N} P(w_i = z_N^n) = 1, \tag{4}$$

which means that the density function in (2) and (3) is a probability density function.

Proof. We need to show that (4) can be guaranteed by (2) and (3). From (2) and (3), we have

$$\begin{aligned} \sum_{n=0}^{2^N} P(w_i = z_N^n) &= \sum_{m=0}^{2^N-1} \sum_{n=0}^{2^N} P(w_i = z_N^n | w_i \in L_m) P(w_i \in L_m) \\ &= \sum_{m=0}^{2^N-1} \Big\{ P(w_i = z_N^m | w_i \in L_m) \\ &\quad + P(w_i = z_N^{m+1} | w_i \in L_m) \Big\} \cdot \frac{1}{2^N} \\ &= \sum_{m=0}^{2^N-1} \left\{ \frac{\Delta z_N - (w_i - z_N^m)}{\Delta z_N} \frac{\Delta z_N - (z_N^{m+1} - w_i)}{\Delta z_N} \right\} \cdot \frac{1}{2^N} \\ &= \sum_{m=0}^{2^N-1} \left\{ \frac{\Delta z_N + \Delta z_N + z_N^m - z_N^{m+1}}{\Delta z_N} \right\} \cdot \frac{1}{2^N} \\ &= \sum_{m=0}^{2^N-1} \left\{ \frac{\Delta z_N + \Delta z_N - \Delta z_N}{\Delta z_N} \right\} \cdot \frac{1}{2^N} \\ &= \sum_{m=0}^{2^N-1} 1 \cdot \frac{1}{2^N} \\ &= 2^N \cdot \frac{1}{2^N} \\ &= 1. \end{aligned} \tag{5}$$

This lemma holds. □

When considering a particular iterative training process in DNNs, assume that $W_i^k = z_N^m \in Z_N$ $(m = 0, 1, \ldots, 2^N)$ is the weight state at the kth iteration step, where W^k is viewed as a vector and i denotes the ith element. A weight increment ΔW_i^k on W_i^k is derived from the gradients at the kth iteration. Firstly we establish a boundary constrain on ΔW_i^k:

$$\Delta W_i^k = \begin{cases} min(L - W_i^k, \Delta W_i^k) & \text{if } \Delta W_i^k \ge 0 \\ max(-L - W_i^k, \Delta W_i^k) & \text{else} \end{cases}. \tag{6}$$

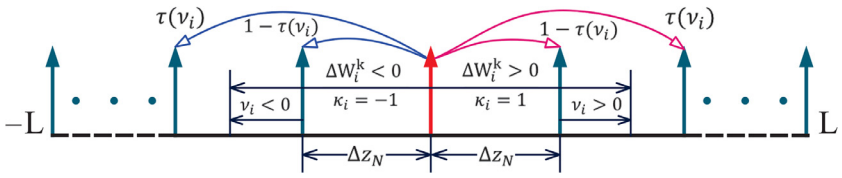

Fig. 1. Illustration of probabilistic gradient projection in DWS during DST training. The DWS is ranging in $[-L, L]$, in which each state is denoted as a up arrow and Δz_N represents the distance between two adjacent states. The direction of state transition is determined by the sign of ΔW_i^k, and the intermediate variables κ_i and ν_i are also determined by ΔW_i^k. Then next possible state can be either $z_N^{m+\kappa_i}$ or $z_N^{m+\kappa_i+sign(\kappa_i)}$. The transition probability to the farther state ($z_N^{m+\kappa_i+sign(\kappa_i)}$) is $\boldsymbol{\tau}(\nu_i)$, and to the closer state ($z_N^{m+\kappa_i}$) is $1-\tau(\nu_i)$. Note that the boundary situation is not considered in this figure.

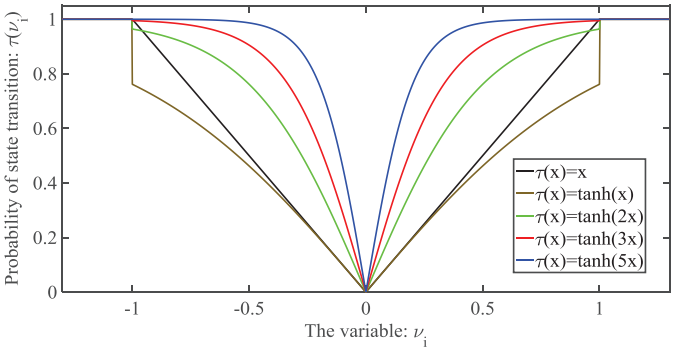

Fig. 2. Relationship between real-valued weight increment and the transition probability. The transition probability is positively associated with the absolute value of ΔW_i^k and is clipped within the range of [0, 1]. The function $tanh(m \cdot x)$ increases the transition possibility compared with the linear mapping, especially when ΔW_i^k is small.

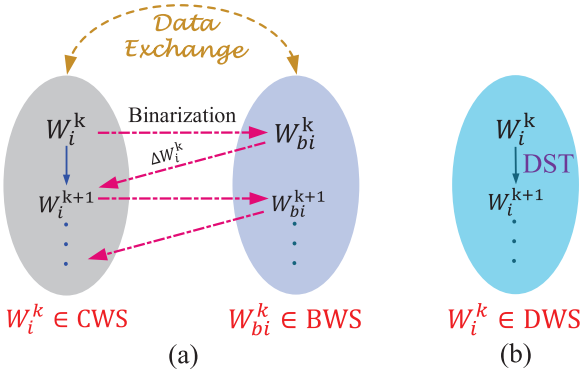

Fig. 3. Comparison between the recent binary weight network (BWN) methods and our DST method. (a) shows that existing BWN schemes frequently switch between two spaces, i.e., CWS and BWS at each iteration. (b) shows that DST is constrained in DWS during the whole training process.

Then we decompose the above ΔW_i^k as:

$$\Delta W_i^k = \kappa_i \Delta z_N + \nu_i, \tag{7}$$

such that

$$\kappa_i = fix(\Delta W_i^k / \Delta z_N) \tag{8}$$

and

$$\nu_i = rem(\Delta W_i^k, \Delta z_N), \tag{9}$$

where $fix(.)$ is a round operation toward zero, and $rem(x, y)$ generates the remainder of the division between two numbers and keeps the same sign with x.

We now discuss how to project $W_i^k + \Delta W_i^k$ in CWS to a state W_i^{k+1} in DWS, i.e., $W_i^{k+1} \in Z_N$. To this end, we define a function $\boldsymbol{\tau}(.)$ which denotes a state transition probability:

$$\tau(\nu) = \frac{|\nu|}{\Delta z_N}. \tag{10}$$

It is seen that $0 \le \boldsymbol{\tau}(\nu) \le 1$. Then, we have

$$\begin{aligned} P(W_i^{k+1} &= z_N^{m+\kappa+sign(\kappa)} | W_i^k = z_N^m) = \tau(\nu) \\ P(W_i^{k+1} &= z_N^{m+\kappa} | W_i^k = z_N^m) = 1 - \tau(\nu), \end{aligned} \tag{11}$$

where $sign(x)$ is a function that

$$sign(x) = \begin{cases} 1, & \text{if } x \ge 0 \\ -1, & \text{else} \end{cases}. \tag{12}$$

We can also apply a nonlinear function $f(x) = tanh(m \cdot x)$ on (10). Thus, we have

$$\tau(\nu) = f\left(\frac{|\nu|}{\Delta z_N}\right) = tanh\left(m \cdot \frac{|\nu|}{\Delta z_N}\right), \tag{13}$$

where $tanh(m \cdot x)$ ranging from zero to one, and m is a parameter that affects the transition probability. Fig. 2 shows the curve of $\tau(\nu)$, and detailedly characterizes how m affect the transition probability when $L = 1$ and $N = 1$.

Now we propose a general optimization model as follows:

$$\begin{aligned} &argmin_W \; E(W) \\ &s.t. \; W_i \in Z_N, \end{aligned} \tag{14}$$

where $E(W)$ denotes the cost function in DNNs depending on particular applications, and W is the weight vector with its element W_i being a state of Z_N for $i = 1, 2, \ldots, \mathcal{N}$, and $\mathcal{N}$ is the dimension of variable W. The objective is to minimize the cost function $E(W)$ during the training of DNNs by restraining all states on the feasible domain Z_N in both forward and backward passes. An algorithm named ***discrete state transition (DST)*** method aiming to solve (14) will be presented in the next section.

3. Algorithm

Set the weight range to $[-L, L]$ and the number of discrete weight states to $2^N + 1$. This implies that the distance between adjacent states is $\Delta z_N = \frac{L}{2^{N-1}}$. We propose an iterative method named ***discrete state transition (DST)*** to solve the optimization model (14). The main difference between DST and the recent binary weight network (BWN) [43–45] is illustrated in Fig. 3. In BWN, frequent switches and data exchanges between the CWS and the BWS are required during the training process. The full-precision weights have to be stored at each iteration. The only difference from traditional methods is that the gradient computation is based on the binary version of the stored full-precision weights, termed as "binarization" step. In stark contrast, the weights during the proposed DST are always constrained in a discrete space DWS. The state transition at each iteration is also discrete to guarantee that the next state will not jump out of the same space. A probabilistic gradient projection operator is introduced to transform a continuous weight increment to a discrete state transition.

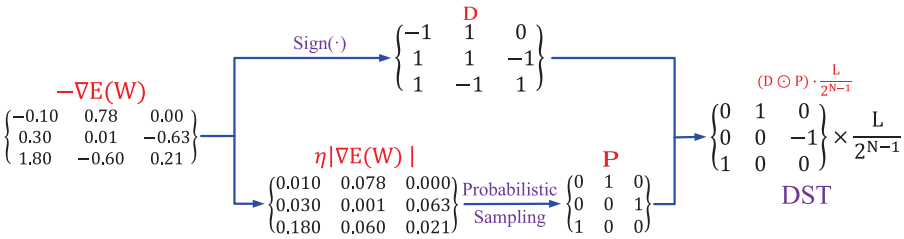

Fig. 4. An example of the DST calculation.

1. Step 1. Randomly initiate weight matrix as $W^0 = [W_1^0, W_2^0, \ldots, W_i^0, \ldots, W_{\mathcal{N}}^0]$ where $W_i^0 \in Z_N$.
2. Step 2. For $W_i^k \in Z_N$ at the kth iteration, we calculate the feedforward output error $\vec{e}$. If $\|\vec{e}\|_F < \varepsilon$ (on the validation set or testing set) for a pre-given parameter ε, stop; otherwise, continue.
3. Step 3. Let the increment vector be $\Delta W^k = [\Delta W_1^k, \Delta W_2^k, \ldots, \Delta W_i^k, \ldots, \Delta W_{\mathcal{N}}^k]$. We would like to point out that ΔW^k can be obtained by different ways. In standard gradient decent method, $\Delta W^k = [-\eta \nabla E(W_1^k), \ldots, -\eta \nabla E(W_i^k), \ldots, -\eta \nabla E(W_{\mathcal{N}}^k)]$ where $\nabla E(W^k)$ denotes the gradient and η is a chosen step. There are also other back propagation (BP) based learning rules, such as Adam [52].
4. Step 4. Establish the boundary constrain on ΔW^k and decompose it to κ and ν according to Eqs (6)–(9).
5. Step 5. At step $k+1$, define a probabilistic projection operator $\mathcal{P}(\cdot)$ which probabilistically projects $W^k + \Delta W^k \in$ CWS to a state $W^{k+1} \in$ DWS.
   (a) Acquire the function of state transition direction $D^k = sign(\nu)$.
   (b) Calculate the state transition probability vector $P^k = [p_1^k, p_2^k, \ldots, p_i^k, \ldots, p_{\mathcal{N}}^k])$, where p_i^k is a random variable in {0, 1} with the following conditional probability for given $\nu = [\nu_1, \nu_2, \ldots, \nu_i, \ldots, \nu_{\mathcal{N}}]$:
   $$P\left(p_i^k = 1 | \nu_i\right) = \tau(|\nu_i|) \tag{15}$$
   and
   $$P\left(p_i^k = 0 | \nu_i\right) = 1 - P\left(p_i^k = 1 | \nu_i\right) = 1 - \tau(|\nu_i|). \tag{16}$$
   (c) Sample the elements of P^k to binary values, denoted as $\mathbf{P}^k = [\mathbf{P}_1^k, \mathbf{P}_2^k, \ldots, \mathbf{P}_i^k, \ldots, \mathbf{P}_{\mathcal{N}}^k]$ where the element $\mathbf{P}_i^k \in \{0, 1\}$, by applying Monte Carlo method.
   (d) Finally, we construct the probabilistic state transition vector given by $(D^k \odot \mathbf{P}^k) \cdot \Delta z_N$, where $\odot$ denotes the multiplication of each element in two matrices with the same dimension.
6. Step 6. Update the weight $W^{k+1} = \mathcal{P}(W^k + \Delta W^k) = W^k + (\kappa + D^k \odot \mathbf{P}^k) \cdot \Delta z_N$. Obviously, W^{k+1} also locates in the discrete weight space Z_N.
7. Step 7. Update $k \leftarrow k+1$, and go to Step 2.

A simple implementation of DST procedure is illustrated in Fig. 4. For simplification, here we assume $\kappa = 0$ which constrains the discrete transition only occurring between neighboring states. The state transition is driven by the transition direction and transition probability. The direction is directly determined by the sign information of the weight increment. In particular, positive weight increment allows transition toward positive direction, negative weight increment transits reversely, and zero weight increment has no effect. The transition probability is positively correlated to the absolute value of the weight increment, then it will be sampled to be either zero or one. A sampled value "1" indicates the occurrence of the state transition event and "0" means that no state transition event happens. Finally, the DST matrix is generated by the element-wise production of the direction and binary sampling result of the transition probability, as well as the distance between adjacent states in DWS, i.e., $\Delta z_N = \frac{L}{2^{N-1}}$.

4. Convergence analysis

We analyze the convergence of DST when $\Delta W^k = [\Delta W_1^k \ \ldots \ \Delta W_i^k \ \ldots \ \Delta W_{\mathcal{N}}^k]$ is obtained by the gradient information $\Delta W_i^k = -\eta \cdot \nabla E(W_i^k)$. Here the probabilistic projection operator is denoted as $\mathcal{P}_{grad}(\cdot)$.

$$W^{k+1} = W^k + \mathcal{P}_{grad}(-\eta \cdot \nabla E(W^k)). \tag{17}$$

The following Theorem 1 establishes the convergence of DST in this case.

Theorem 1. *Assume that ΔW^k is obtained by*

$$\begin{aligned}\Delta W^k &= -\eta \nabla E(W^k) \\ &= [-\eta \nabla E(W_1^k), \ldots, -\eta \nabla E(W_i^k), \ldots, -\eta \nabla E(W_{\mathcal{N}}^k)],\end{aligned}$$

where η is the learning rate. Then the iteration in (17) converges in probability to $E(W^*)$ satisfying first-order optimality condition when N is sufficiently large and η is sufficiently small.

Proof. From (17), we have

$$E(W^{k+1}) = E\left(W^k + \mathcal{P}_{grad}(-\eta \cdot \nabla E(W^k))\right). \tag{18}$$

According to the algorithm part, we obtain

$$E(W^{k+1}) = E\left(W^k + (\kappa + D^k \odot \mathbf{P}^k) \cdot \Delta z_N\right), \tag{19}$$

where κ, D^k and $\mathbf{P}^k$ are determined by the given W^k and $-\eta \cdot \nabla E(W^k)$, and Δz_N is the distance between adjacent states. Note that Δz_N tends to 0 when $N \to \infty$, or elements in vector $\mathbf{P}^k$ probabilistically tend to 0 in probability when $\eta \to 0$. Then we can conduct Taylor expansion when $N \to \infty$ and $\eta \to 0$ as:

$$\begin{aligned}E(W^{k+1}) &= E(W^k) + \Delta z_N \nabla E(W^k)^T (\kappa + D^k \odot \mathbf{P}^k) + o(\Delta z_N) \\ &= E(W^k) + \Delta z_N \nabla E(W^k)^T (\kappa + sign(\nu) \odot \mathbf{P}^k) + o(\Delta z_N) \\ &= E(W^k) + \Delta z_N \nabla E(W^k)^T (|\kappa| sign(\kappa) + sign(\nu) \odot \mathbf{P}^k) \\ &\quad + o(\Delta z_N).\end{aligned} \tag{20}$$

According to the definition of κ and ν in Eqs (7)–(9), both the κ and ν have the same sign information with $-\eta \nabla E(W^k)$. Then we have

$$\begin{aligned}E(W^{k+1}) &= E(W^k) + \Delta z_N \cdot \nabla E(W^k)^T [|\kappa| \cdot sign(-\eta \nabla E(W^k)) \\ &\quad + sign(-\eta \nabla E(W^k)) \odot \mathbf{P}^k] + o(\Delta z_N) \qquad (21) \\ &= E(W^k) - \Delta z_N \cdot \nabla E(W^k)^T [|\kappa| \cdot sign(\nabla E(W^k)) \\ &\quad + sign(\nabla E(W^k)) \odot \mathbf{P}^k] + o(\Delta z_N). \qquad (21)\end{aligned}$$

Furthermore, the $\mathbf{P}^k$ is either zero or one, i.e., non-negative. Then we always have $\nabla E(W^k)^T \cdot [|\kappa| \cdot sign(\nabla E(W^k)) + sign(\nabla E(W^k)) \odot \mathbf{P}^k] \geq 0$. Then we obtain

$$\begin{aligned}E(W^{k+1}) &= E(W^k) - \Delta z_N \cdot \nabla E(W^k)^T [|\kappa| \cdot sign(\nabla E(W^k)) \\ &\quad + sign(\nabla E(W^k)) \odot \mathbf{P}^k] \leq E(W^k).\end{aligned} \tag{22}$$

Since the cost function $E(W^k)$ is nonnegative and $E(W^{k+1}) \leq E(W^k)$, we can draw the conclusion that the iteration is convergent, as long as N is sufficiently large and η is sufficiently small.

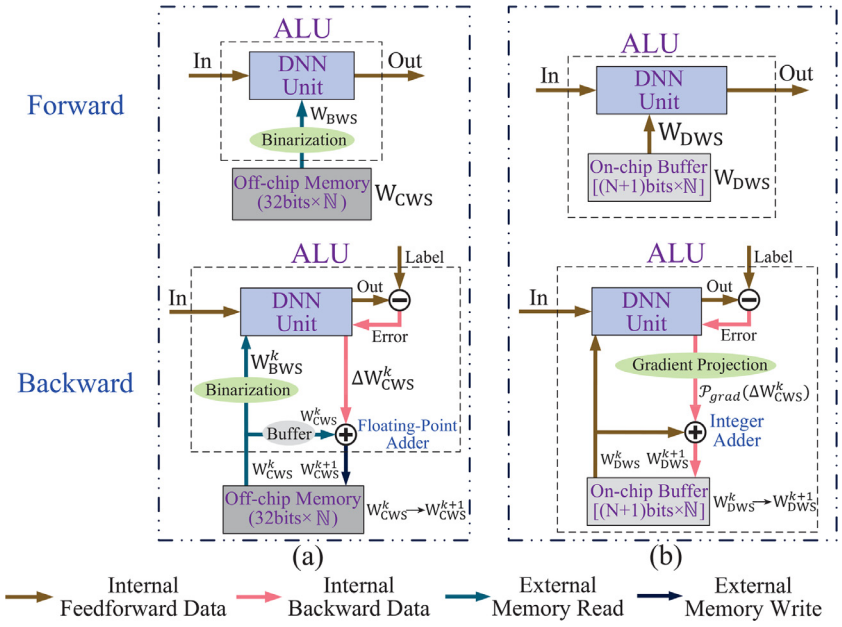

Fig. 5. Data flow in the training architectures of the existing BWN and the proposed DST methods. (a) and (b) illustrate the data flow for existing BWN and our proposed DST, respectively. It is seen that DST only requires internal buffer without accessing the external full-precision memory in both forward computation and backward training, which makes it possible to break through the various restrictions from the cost of memory, training time and power.

The iteration stops when the first order optimality condition that $\nabla E(W^k) = 0$. □

Although Theorem 1 holds only when N is sufficiently large and η is sufficiently small, N and η can be proper chosen in various applications. We would also like to point out that for the non-convex optimization problem (14), the proposed DST method, like any optimization method with a reasonably low complexity, can only guarantee converging to a local minimum. An interesting observation we made in our extensive implementations of DST starting with different initial W^0, however, is that the solutions of different rounds of implementations typically lead to nearly the similar performance. Whether this means that the solution found by a single round implementation of DST is steadily close to the sub-global optimum requests further studies. Unless otherwise specified, hereafter we present the results of an implementation of DST with a randomly given W^0.

5. Results

5.1. Learning architecture

Fig. 5 shows the main differences regarding the learning architecture in training DNNs using BWN and DST methods. In the forward pass of BWN, the full-precision weights have to be read from the external memory and converted to binary values $\{-1, 1\}$, termed as a binarization step which can be based on either a stochastic sampling or just a deterministic sign operation, while DST only consumes a much smaller memory cell for each weight element (only (N+1) bits). As a result, the external full-precision memory device (regarded as off-chip memory) can be replaced by an internal buffer (regarded as on-chip memory) to enable fast data exchange within the ALU (Arithmetic Logical Unit). In addition, the binarization step is unnecessary in the forward pass in DST.

In the backward pass, the case becomes a bit more complicated. In summary, the major differences are listed as follows: (a) Due to the requirement of binary weights at computation time while full-precision weights at updating time, the BWN has to frequently switch between the external CWS and internal BWS, and buffer the full-precision weights for the following updating operation. Unlike this, DST can load the discrete weights from internal buffer for all computations. (b) Although DST introduces a gradient based probabilistic projection operator to transform the real-valued weight to discrete states, this simplify the design of the adder for weight updating. In particular, the BWN requires a floating-point and full-precision adder to generate the new weights based on the old weights and weight increments at each iteration. While the discrete transition in DST indicates that the state variable n in (1) is only possible to be integers, which greatly simplifies the floating-point adder in BWN into an integer version.

5.2. Cost comparisons

The cost comparisons of DST and BWN are shown in Tables 1 and 2. As seen in Table 1, in the forward pass, BWN requires a full-precision external memory cell (usually 32 bits), one external read operation C_R, one space switching step C_S (binarization). Thus, the total computation cost is $(C_R + C_S)$ for BWN methods. While DST only requires a $(N+1)$-bit internal buffer cell, and the on-chip read cost is negligible compared with the mentioned off-chip memory operation. The space switching is no longer necessary. Therefore, the total computational cost in DST forward pass can be ignored compared with the BWN scheme. We conduct a similar analysis for the backward pass case in Table 2. The cost for BWN appears with the same external memory cell (32 bits) and $(C_R + C_W + C_S + \textit{float weight adder})$ computing resources; while for DST, only a $(N+1)$-bit internal buffer and $(C_S + \textit{fixed point weight adder})$ are consumed. In general, under the framework of DST, algorithms could be designed for online training of DNNs without external memory read and write cost. As well, adder cost for updating weights in backward pass can be also significantly reduced. Thus, the DST greatly improve the memory and resulting computing resources which provides a top-down theoretical guidance toward the on-chip training of large scale DNN in the near future.

Table 1
Cost comparisons of the two different methods in the forward pass. Here the cost is only for one weight element during one training iteration. The cost includes the consumption of memory, time and energy. C_R and C_S denote the cost of memory read and weight space switching, respectively.

Methods	Memory	Read	Write	Computation in forward pass		Total cost
				Space switching	Weight adder	
BWN	32bits	C_R	0	C_S	0	$C_R + C_S$
DST	(N+1)bits	≈ 0	0	0	0	≈ 0

Table 2
Cost comparisons of the two different methods in the backward pass. Here C_R and C_S have the same meanings in Table 1, and C_W is the cost of memory write.

Methods	Memory	Read	Write	Computation in backward pass		Total cost
				Space switching	Weight adder	
BWN	32bits	C_R	C_W	C_S	Float	$C_R + C_W + C_S +$ *float adder*
DST	(N+1)bits	≈ 0	≈ 0	C_S	Fixed-point	$C_S +$ *fixed point adder*

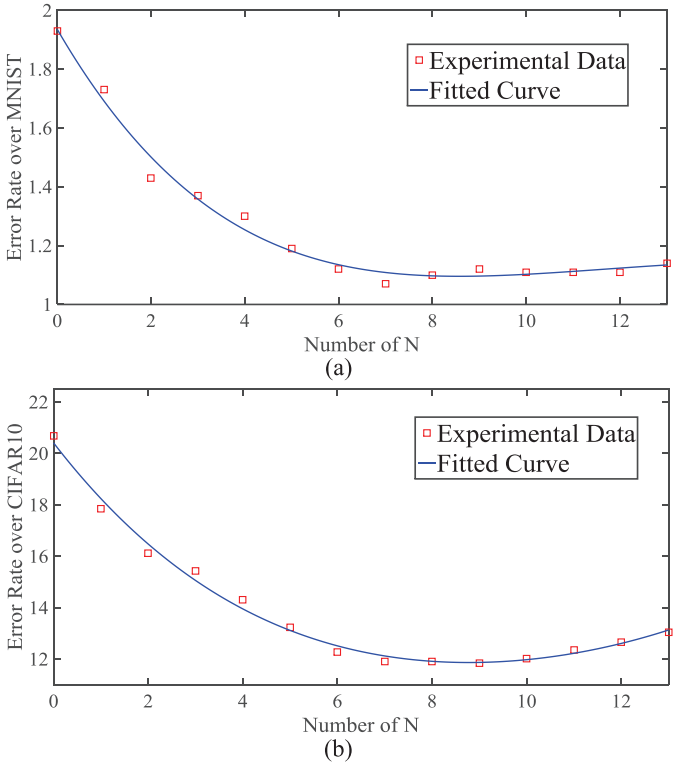

Fig. 6. Simulation results over MNIST and CIFAR10 with different number of discrete states. The number of discrete states in DST method can be flexibly reconfigured, which is hardware friendly. The comparable performance is successfully achieved. More states often results in smaller error rate but this is not always the case. Our experiments show that when N becomes too large, the performance may degrades due to overfitting. There often exists a most suitable space Z_N for a specified task.

5.3. Simulation results and discussion

We test our DST algorithm over the MNIST, CIFAR10 and SVHN datasets. The network structures similar to [43] are adopted. From the results in Fig. 6, we draw two key conclusions as follows. (a) With the state variable N relatively larger, the DNNs usually perform better, i.e., achieving smaller error rates; while the performance gradually degrades when N becomes further larger. (b) There often exists a most suitable space Z_N space for a specified task. As seen in Fig. 6 more detailedly, generally more states often results in smaller error rate but this is not always the case. Our experiments show that when N becomes too large, the performance may degrades due to overfitting. For instance, we observe a best accuracy for $N \approx 7$ over MNIST, $N \approx 8$ over SVHN and $N \approx 9$ over CIFAR10. This indicates training with full-precision weights may be not the best choice as overfitting may occur. In this sense, DST is another way to avoid the overfitting trap in training DNNs. By using the DST algorithm, the comparable performance could be obtained. In this paper, the error rate of 1.07% over MNIST, 11.84% over CIFAR10 and 2.45% over SVHN are obtained. The comparisons of DST and some state-of-the-art algorithms are listed in Table 3. It is seen that DST is very close to existing reported results [43,44] over all data sets, though the weights in DST are always constrained in a discrete weight space (DWS) during the whole training process in which existing methods become inapplicable. In addition, it is predictable that DST framework can be extended to large scale networks.

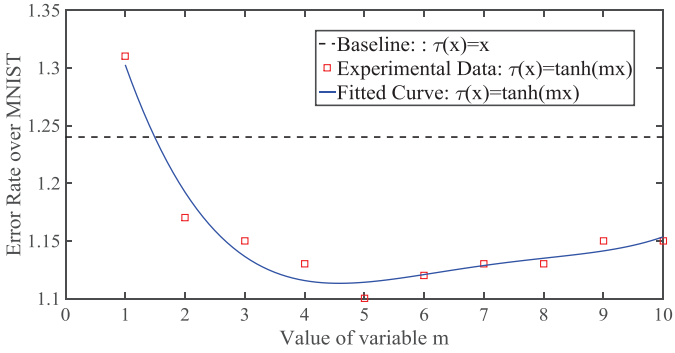

Fig. 7. Illustration of how does m in Eq. (13) affect the performance of DST. The baseline denotes the case when $\tau(x)$ is linear with $N = 3$.

We also notice that different values of m may affect the performance of DST when a nonlinear function $f(x) = tanh(mx)$ is applied on (10). Experimental observations show that a slightly increase of m ($m > 1$) could improve the performance of DST while further increase does not improve any further. For example, Fig. 7 shows the results in MNIST for the case that $N = 3$, where it is seen that when $m = 5$, DST attains its best performance. The best value of m may be slightly different for different specific applications.

Based on the simulation results, we finally envisage an ideal training curve during finding the best weight space Z_N, as shown in Fig. 8. All the mentioned conclusions are still valid here. More importantly, we emphasize that training DST in the best Z_N could outperform the training in CWS. This is due to the fact that the latter one is easy to fall into the overfitting trap, especially in large and complicated models. Although DST does not beat the

Table 3
Comparisons with state-of-the-art algorithms. Noting that the DWS and CWS are short for discrete weight space and continuous weight space respectively. Only the proposed DST method is trained in DWS, while all the other ones have to exploit the weight information trained in CWS.

Methods	Datasets			Notes
	MNIST	CIFAR10	SVHN	
DST	1.07% (N = 7)	11.84% (N = 9)	2.45% (N = 8)	–
BinaryConnect [39]	1.23%	12.04%	2.47%	–
BinaryConnect [43]	1.29 ± 0.08%	9.90%	2.30%	Deterministic
BinaryConnect [43]	1.18%± 0.04%	8.27%	2.15%	Stochastic
BNN [44]	0.96%	11.40%	2.80%	Theano
BNN [44]	1.40%	10.15%	2.53%	Torch7
Maxout networks [53]	0.94%	11.68%	2.47%	–

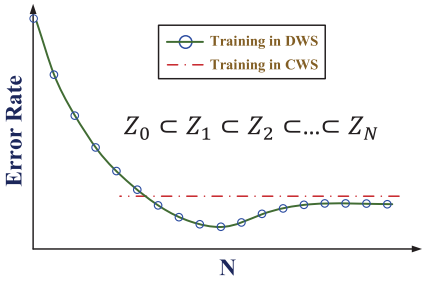

Fig. 8. Ideal training curve in DWS. Training in CWS may be easy to fall into the over-fitting trap, especially in large models. While training in DWS under the DST frame is really helpful to find a better weight space with limited states which promises to beat the full-precision training.

full-precision training in experiments (about $1\% - 2\%$ gap over CIFAR10), we should note that it depends on an ideal adjustment of other parameters in each Z_N space to achieve this interesting conclusion, including the training batch size, learning rate, training epochs, boundary of weight range, scaling factor of internal project function, an so on. It is quite promising to verify this assumption with more experiments over other data sets and with careful parameter adjustment. Another advantage of the DST training method locates at the flexibly reconfigurable property on the number of discrete states, which is hardware friendly for many emerging memory devices, such as memristors [49,50]. Only binary or ternary states in the reported methods would limit the hardware capability. While under our DST framework, the number of the intermediate states is easy to be reconfigured according to the supported level of the memory device for weight storage.

6. Conclusion

In the work [44], Courbariaux et al. pointed out the recent schemes have to save the values of the full-precision weights, which leads to the outstanding memory and computation bottleneck during DNNs training process. We have introduced a unified DST framework based on probabilistic projection method that constrains the weight matrices in a DWS with configurable number of states. Simulation results on MNIST, CIFAR10 and SVHN datasets show the effectiveness of our proposed algorithms. Under DST, algorithms could be designed for online training of DNNs without external memory access cost, and computational cost could also be significantly reduced, such as simplified weight adders. When training DNNs, the proposed method does allow the weights to be not only binary values, but also multi-level states in a discrete weight space. This is very important especially when we are considering how to realize the proposed algorithm on portable devices. Because the DST framework enables us to fully utilize the emerging memory devices which naturally provides multiple levels of memory states, rather than only binary states. In addition, our method can be easily combined with the recent Binarynets (or XNOR-nets) to further propose network architecture with both discrete weights and discrete activations, which deserves further investigations. In short, this work enables low-cost training of DNNs and really paves the way toward on-chip learning for various portable devices in the near future.

Acknowledgments

The work was partially supported by National Natural Science Foundation of China (No. 61475080, No. 61603209), Beijing Natural Science Foundation (No. 4164086) and Independent research plan of Tsinghua University (No. 20151080467).

References

[1] S. Yu, S. Jia, C. Xu, Convolutional neural networks for hyperspectral image classification, Neurocomputing 219 (2017) 88–98.
[2] Y. Liang, J. Wang, S. Zhou, Y. Gong, N. Zheng, Incorporating image priors with deep convolutional neural networks for image super-resolution, Neurocomputing 194 (2016) 340–347.
[3] F. Nian, T. Li, Y. Wang, M. Xu, J. Wu, Pornographic image detection utilizing deep convolutional neural networks, Neurocomputing 210 (2016) 283–293.
[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.
[5] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Proc. Mag. 29 (2012) 82–97.
[6] Z. Huang, S.M. Siniscalchi, C.H. Lee, A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition, Neurocomputing 218 (2016) 448–459.
[7] X. Li, Y. Yang, Z. Pang, X. Wu, A comparative study on selecting acoustic modeling units in deep neural networks based large vocabulary chinese speech recognition, Neurocomputing 170 (2015) 251–256.
[8] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul, Fast and robust neural network joint models for statistical machine translation, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2014, pp. 1370–1380.
[9] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, ArXiv:1409.0473.
[10] F. Richardson, D. Reynolds, N. Dehak, Deep neural network approaches to speaker and language recognition, IEEE Signal Proc. Let. 22 (2015) 1671–1675.
[11] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[12] P. Baldi, K. Bauer, C. Eng, P. Sadowski, D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, Phys. Rev. D 93 (2016) 094034.
[13] M.A.Z. Raja, M.A.R. Khan, T. Mahmood, U. Farooq, N.I. Chaudhary, Design of bio-inspired computing technique for nanofluidics based on nonlinear JefferyCHamel flow equations, Can. J. Phys. 94 (2016) 474–489.
[14] Z. Masood, K. Majeed, R. Samar, M.A.Z. Raja, Design of mexican hat wavelet neural networks for solving Bratu type nonlinear systems, Neurocomputing 221 (2017) 1–14.
[15] K.T. Schutt, F. Arbabzadah, S. Chmiela, K.R. Muller, A. Tkatchenko, Quantum–chemical insights from deep tensor neural networks, Nat. Commun. 8 (2017) 13890.
[16] K.T. Kim, Optical physics: ultrashort light pulses shake atoms, Nature 530 (2016) 41–42.
[17] I. Ahmad, M.A.Z. Raja, M. Bilal, F. Ashraf, Neural network methods to solve the LaneCEmden type equations arising in thermodynamic studies of the spherical gas cloud model, Neural Comput. Appl. (2016) 1–16.

[18] M.A.Z. Raja, F.H. Shah, M. Tariq, I. Ahmad, S.I. Ahmad, Design of artificial neural network models optimized with sequential quadratic programming to study the dynamics of nonlinear Troesch's problem arising in plasma physics, Neural Comput. Appl. (2016) 1–27.
[19] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, O. Temam, Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning, in: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM Sigplan Notices 49, 2014, pp. 269–284.
[20] 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, D.S. Modha, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (2014) 668–673.
[21] F. Alibart, E. Zamanidoost, D.B. Strukov, Pattern classification by memristive crossbar circuits using ex situ and in situ training, Nat. Commun. 4 (2013) 2072.
[22] L. Deng, G. Li, N. Deng, D. Wang, Z. Zhang, W. He, H. Li, J. Pei, L. Shi, Complex learning in bio-plausible memristive networks, Sci. Rep. 5 (2015) 10684.
[23] G. Li, L. Deng, D. Wang, W. Wang, F. Zeng, Z. Zhang, H. Li, S. Song, J. Pei, L. Shi, Hierarchical chunking of sequential memory on neuromorphic architecture with reduced synaptic plasticity, Front. Comput. Neuro. 10 (2016) 136.
[24] M. Prezioso, F. Merrikh-Bayat, B.D. Hoskins, G.C. Adam, K.K. Likharev, D.B. Strukov, Training andoperation of an integrated neuromorphic network based on metal-oxide memristors, Nature 521 (2015) 61–64.
[25] S. Park, M. Chu, J. Kim, J. Noh, M. Jeon, B.H. Lee, H. Hwang, B. Lee, B.G. Lee, Electronic system with memristive synapses for pattern recognition, Sci. Rep. 5 (2015) 10123.
[26] 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, D.S. Modha, Truenorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip, IEEE Trans. Comput. Aid. D. 34 (2015) 1537–1557.
[27] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, 2015, ArXiv preprint arXiv:1510.00149.
[28] H. van Nguyen, K. Zhou, R. Vemulapalli, Cross-domain synthesis of medical images using efficient location-sensitive deep network, Proceedings of the International Conference on Medical Image Computing and ComputerAssisted Intervention (MICCAI), Springer International Publishing, 2015, pp. 677–684.
[29] E.L. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, Adv. Neural Inf. Process. Syst. (NIPS) (2014) 1269–1277.
[30] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, 2014, ArXiv preprint arXiv:1405.3866.
[31] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 770–778.
[32] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, Am. Assoc. Artif. Intell. (AAAI) (2017) 4278–4284.
[33] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1MB model size, 2016, ArXiv preprint arXiv:1602.07360.
[34] X. Pan, L. Li, H. Yang, Z. Liu, J. Yang, L. Zhao, Y. Fan, Accurate segmentation of nuclei in pathological images via sparse reconstruction and deep convolutional networks, Neurocomputing 229 (2017) 88–99.
[35] S. Arora, A. Bhaskara, R. Ge, T. Ma, Provable bounds for learning some deep representations, in: Proceedings of the International Conference on Machine Learning (ICML), 2014, pp. 584–592.
[36] E.L. Machado, C.J. Miosso, R. von Borries, M. Coutinho, P.D.A. Berger, T. Marques, R.P. Jacobi, Computational cost reduction in learned transform classifications, 2015, ArXiv preprint arXiv:1504.06779.
[37] Y. Gong, L. Liu, M. Yang, L. Bourdev, Compressing deep convolutional networks using vector quantization, 2014, ArXiv preprint arXiv:1412.6115.
[38] S. Anwar, K. Hwang, W. Sung, Fixed point optimization of deep convolutional neural networks for object recognition, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1131–1135.
[39] Z. Lin, M. Courbariaux, R. Memisevic, Y. Bengio, Neural networks with few multiplications, 2015, ArXiv preprint arXiv:1510.03009.
[40] H.K. Kwan, C.Z. Tang, Multiplierless multilayer feedforward neural network design suitable for continuous input-output mapping, Electron. Lett. 29 (1993) 1259–1260.
[41] P.Y. Simard, H.P. Graf, Backpropagation without multiplication, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 1994, pp. 232–239.
[42] M. Courbariaux, Y. Bengio, J.P. David, Training deep neural networks with low precision multiplications, 2014, ArXiv preprint arXiv:1412.7024.
[43] M. Courbariaux, Y. Bengio, J.P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2015, pp. 3105–3113.
[44] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1, 2016, ArXiv preprint arXiv:1602.02830.
[45] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Proceedings of the XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, European Conference on Computer Vision, Springer International Publishing, 2016, pp. 525–542.
[46] F. Li, B. Zhang, B. Liu, Ternary weight networks, 2016, ArXiv preprint arXiv:1605.04711.
[47] C. Zhu, S. Han, H. Mao, W.J. Dally, Trained ternary quantization, 2016, ArXiv preprint arXiv:1612.01064.
[48] S.K. Esser, P.A. Merolla, J.V. Arthur, A.S. Cassidy, R. Appuswamy, A. Andreopoulos, D.J. Berg, J.L. McKinstry, T. Melano, D.R. Barch, C.D. Nolfo, P. Datta, A. Amir, B. Taba, M.D. Flickner, D.S. Modha, Convolutional networks for fast, energy-efficient neuromorphic computing, in: Proceedings of the National Academy of Science of the United States of America (PNAS) 113, 2016, pp. 11441–11446.
[49] D. Kuzum, S. Yu, H.S.P. Wong, Synaptic electronics: materials, devices and applications, Nanotechnology 24 (2013) 382001.
[50] H. Kim, M.P. Sah, C. Yang, L.O. Chua, Memristor-based multilevel memory, in: Proceedings of the IEEE International Workshop on Cellular Nanoscale Networks and their Applications (CNNA), 2010, pp. 1–6.
[51] L. Deng, D. Wang, Z. Zhang, P. Tang, G. Li, J. Pei, Energy consumption analysis for various memristive networks under different learning strategies, Phys. Lett. A 380 (2016) 903–909.
[52] D. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, ArXiv preprint arXiv:1412.6980.
[53] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, ICML 28 (2013) 1319–1327.

Guoqi Li received the B.Eng. degree and M.Eng. degree from Xi'an University of Technology and Xi'an Jiaotong University, P R China, in 2004 and 2007, respectively, and Ph.D. degree from Nanyang Technological University, Singapore in 2011. He was a Scientist with Data Storage Institute and Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), Singapore, from September 2011 to March 2014. Since March 2014, he has been an Assistant Professor with the Department of Precision Instrument, Tsinghua University, P. R. China. His current research interests include brain inspired computing, complex systems, neuromorphic computing, machine learning and system identification. Dr. Li has published more than 40 journal and conference papers. He services as a reviewer for a number of international journals and has also been actively involved in professional services such as serving as an International Technical Program Committee member, and a Track Chair for international conferences.

Lei Deng received his B.Eng. degree and Ph.D. degree from University of Science and Technology of China and Tsinghua University in 2012 and 2017, respectively. His research interests include neuromorphic chip, brain-like system, memristive device, machine learning, computing neuroscience, complex networks.

Lei Tian received his B. Eng. degree in the department of Precision and Instrument, Tsinghua University, Beijing, China in 2015. He is currently a Ph.D. candidate in the department of Precision and Instrument, Tsinghua University, Beijing, China. His research interests include Brain inspired computing and neuromorphic engineering.

Haotian Cui is currently a graduate student in the Department of Biomedical Engineering, Tsinghua University, Beijing, China. He received his bachelor degree in Tsinghua University in 2015. He has acquired awards of several nationwide engineering contest and has internship work experience about deep learning algorithms in AI industry. Currently he is doing research in the field of Recurrent Neural Networks and Reinforcement Learning.

Wentao Han is currently a postdoctoral researcher in Department of Computer Science and Technology, Tsinghua University. He received his B.Eng. and Ph.D. degrees in computer science from Tsinghua University in 2008 and 2015, respectively. His research interests include big data processing systems and neuromorphic systems.

Jing Pei is currently an associate professor at Tsinghua University, Beijing, China. He worked for Tsinghua University since 1990. His research interests include optical information storage system, signal processing, channel coding and neural network coding methods. He has published over 50 papers in international journals and and obtained over 40 invention patents. He won once second prize of national invention award and once second prize of national scientific and technological progress award. Recently, his work is focusing on artificial cognitive memory (ACM) for brain-inspired computing based on neuromorphic engineering, including information characterization, transfer and codec theory and method in the system and chip design.

Luping Shi received the B. S. degree and M. S. degree in Physics from Shandong University, China in 1981 and 1988, respectively, and Ph. D. degree in Physics from University of Cologne, Germany in 1992. He was a postdoctoral fellow at Fraunhofer Institute for Applied Optics and Precision Instrument, Jela, Germany in 1993 and a research fellow in Department of Electronic Engineering, City University of Hong Kong from 1994 to 1996. From 1996 to 2013, he worked in Data Storage Institute (DSI), Singapore as a senior scientist and division manager, and led nonvolatile solid-state memory (NVM), artificial cognitive memory (ACM) and optical storage researches. He is the recipient of the National Technology Award 2004 Singapore, the only one awardee that year. He joined THU, China, as a national distinguished professor and director of Optical Memory National Engineering Research Center in 2013. By integrating 7 departments (Precision Instrument, Computer Science, Electrical Engineering, Microelectronics, Medical School, Automatics, Software School), he established CBICR, THU in 2014, and served as the director. His research interests include brain-inspired computing, neuromorphic engineering, NVM, memristor devices, optical data storage, photonics, etc. He has published more than 150 papers in prestigious journals including Science, Nature Photonics, Advanced Materials, Physical Review Letters, filed and granted more than 10 patents and conducted more than 60 keynote speech or invited talks at many important conferences during last 10 years. Dr. Shi is a SPIE fellow, and general co-chair of the 9th Asia-Pacific Conference on Near-field Optics2013, IEEE NVMTS 2011–2015, East-West Summit on Nanophotonics and Metal Materials 2009 and ODS009. He is a member of the editorial board of Scientific Reports (Nature Publishing Group).