

Direct Feedback Alignment based Convolutional Neural Network Training for Low-power Online Learning Processor

Donghyeon Han

School of Electrical Engineering
 KAIST, Daejeon, Republic of Korea
 hdh4797@kaist.ac.kr

Hoi-jun Yoo

School of Electrical Engineering
 KAIST, Daejeon, Republic of Korea
 hjyoo@kaist.ac.kr

Abstract

There were many algorithms to substitute the back-propagation (BP) in the deep neural network (DNN) training. However, they could not become popular because their training accuracy and the computational efficiency were worse than BP. One of them was direct feedback alignment (DFA), but it showed low training performance especially for the convolutional neural network (CNN). In this paper, we overcome the limitation of the DFA algorithm by combining with the conventional BP during the CNN training. To improve the training stability, we also suggest the feedback weight initialization method by analyzing the patterns of the fixed random matrices in the DFA. Finally, we propose the new training algorithm, binary direct feedback alignment (BDFA) to minimize the computational cost while maintaining the training accuracy compared with the DFA. In our experiments, we use the CIFAR-10 and CIFAR-100 dataset to simulate the CNN learning from the scratch and apply the BDFA to the online learning based object tracking application to examine the training in the small dataset environment. Our proposed algorithms show better performance than conventional BP in both two different training tasks especially when the dataset is small. Furthermore, we examined the efficiency improvement by real chip implementation, and finally, DNN training accelerator with BDFA shows 35.3% lower power consumption compared with hardware which is optimized for BP.

1. Introduction

Deep learning becomes the core of the machine learning and it has been utilized for many applications such as machine translation, speech recognition, and object classifica-

tion. Training the deep neural network (DNN) is an important portion of the deep learning because we need different pre-trained models to cover the various deep learning tasks. One well-known method of DNN training is the algorithm called back-propagation (BP) ([14]). The BP is the gradient descent based training method which follows the steepest gradient to find the optimum weights. Therefore, BP based training can be applied to any DNN configurations if the network consists of any differentiable operations.

Even though the BP shows the outstanding performance in DNN training, BP based training suffers from the overfitting problem. Since the BP easily sinks into a local minimum, we need a large scale of the dataset to avoid the overfitting. In addition, we use various data augmentation techniques such as flipping and cropping. If the DNN training is done in the limited resources and dataset, BP based DNN training is too slow to be converged and shows low accuracy. One example is MDNet ([12]) which introduces an object tracking algorithm with the online learning concept. Real-time implementation is an important issue in object tracking, so it has the limitation to utilize various data augmentation and large dataset for online learning.

To break the BP based DNN training paradigm, Feedback alignment (FA, [11]), was proposed based on the gradient descent methodology, but without following the steepest gradient. The FA pre-defines a feedback weight before starting the training, and the weight is determined by the random values. It is also known to converge slowly, and shows worse performance compared with the previous BP.

Direct feedback alignment (DFA, [13]) was developed by getting the idea from the FA. Although the FA propagates errors from the last layer back to the first layer step-by-step, the DFA propagates errors from the last layer directly to each layer. This approach gives the opportunity of the parallel processing. Since the errors of every layer are generated independently in DFA, we can immediately calculate each layer's gradient if the DNN inference is finished. Moreover, the number of required computation is reduced in the DFA. This is because the number of neurons in the last

This paper is based on preprint: arXiv:1901.01986 "Efficient Convolutional Neural Network Training with Direct Feedback Alignment" Related algorithm is realized as the ASIC introduced in the [3]. This work was supported by the Samsung Research Funding & Incubation Center for Future Technology under Grant SRFC-TB1703-09.

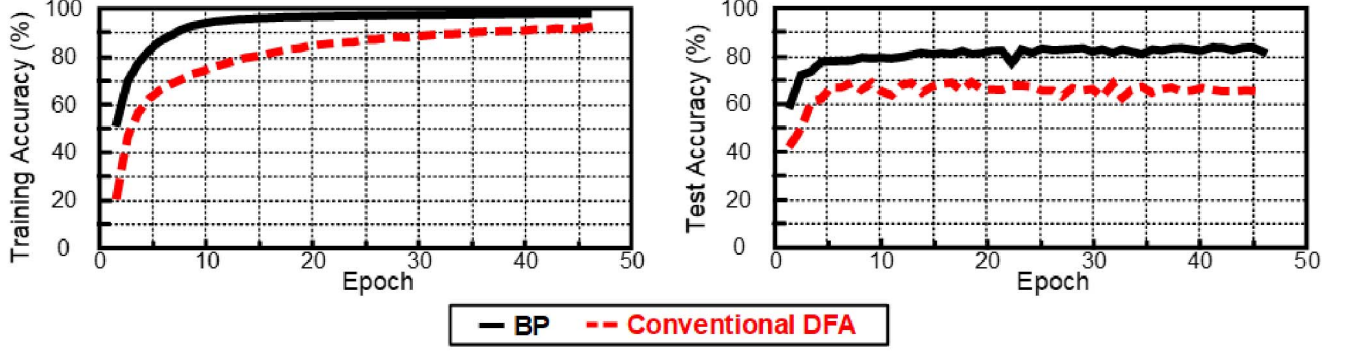


Figure 1. BP (Black) vs Conventional DFA (Read) for CNN Training (Tested in CIFAR-10)

layer is usually fewer than the prior layer, so the size of the feedback weight becomes smaller than the BP. In spite of these advantages, the DFA suffers from accuracy degradation problem. The accuracy degradation problem becomes more serious when the DFA is applied to the CNN case. It is known that the DNN is not learnable with the DFA if the DNN becomes much deeper than AlexNet ([9]).

In this paper, we explain the new DFA algorithm to improve the training accuracy in CNN, and suggest a feedback weight initialization method for the fast convergence. Moreover, we propose the binary direct feedback alignment (BDFA) to maximize the computational efficiency. We verified the training performance in the VGG-16 ([15]) without any data augmentation, and DFA shows higher accuracy compared with the BP. And then, the training with small dataset was proved through the online learning based object tracking application. We also examined the proposed algorithm with the ASIC implementation and proved that our approach enables low-power online learning with better training performance than the conventional BP approach.

The remaining part of the paper is organized as follows. The mathematical notation of the BP, FA, and DFA will be introduced in Section 2. Then, the details of the proposed algorithms will be explained in Section 3. The experiment and hardware implementation result will be followed in Section 4, and 5. The paper will be concluded in Section 6.

2. Preliminaries

2.1. Back-propagation

Back-propagation is a general algorithm for the DNN training, suggested by [14]. Let $\mathbf{o}_i$ be the i^{th} layer's feature map, when $\mathbf{W}_{i+1,i}$ be the weight and bias between the i^{th} layer and the $i+1^{th}$ layer. If the activation function is represented as $f(\cdot)$, and L is the total number of layers, the feature map of the $i+1^{th}$ layer can be calculated as

$$\mathbf{i}_{i+1} = \mathbf{W}_{i+1,i} \mathbf{o}_i, \quad \mathbf{o}_{i+1} = f(\mathbf{i}_{i+1}) \quad (1)$$

Once the inference is over, the inference result is compared with the pre-defined labels, and the error map $\mathbf{e}_L$ is calculated by a loss function such as cross-entropy. The error is propagated gradually from the last layer to the first layer, and the i^{th} layer's error map can be calculated as

$$\mathbf{e}_i = (\mathbf{W}_{i+1,i}^T \mathbf{e}_{i+1}) \odot f'(\mathbf{i}_{i+1}) \quad (2)$$

where $\odot$ is an element-wise multiplication operator and $f'(\cdot)$ is the derivative of the non-linear function. We need the transposed weight matrix, $\mathbf{W}_{i+1,i}^T$, to propagate the errors in MLP case. After the BP, the gradient of each layer is computed by using both the i^{th} layer's feature map and $i+1^{th}$ layer's error map. The i^{th} layer's gradient, $\mathbf{G}_i$ is calculated as following.

$$\mathbf{G}_{i,b} = \mathbf{e}_{i+1} \mathbf{o}_i^T, \quad i \in \{0, 1, \dots, L-1\} \quad (3)$$

$$\mathbf{G}_i = \frac{1}{B} \sum_b \mathbf{G}_{i,b}, \quad \mathbf{W}'_{i+1,i} = \mathbf{W}_{i+1,i} - \eta \mathbf{G}_i \quad (4)$$

If we use the mini-batch gradient descent with the batch size B , total B gradients, $\mathbf{G}_{i,b}$ are averaged and get the $\mathbf{G}_i$. $\mathbf{G}_i$ is used to update the $\mathbf{W}_{i+1,i}$ by multiplying learning rate, η . In the CNN case, the matrix multiplication operations in every step described in MLP is substituted with the convolution operations. One more different thing is that it uses 180°flipped kernel instead of the transposed weight used in the BP. Other operations are as same as in the MLP case.

2.2. Feedback Alignment

Feedback alignment (FA) which was introduced by [11] substituted the transposed weight with the fixed random matrix in error propagation operation. In other words, they pre-defined the feedback weight which has the same size as the transposed weight, but the values of the feedback weight are determined randomly. Although the weights used in the training are updated for every iteration, the pre-defined random matrix is maintained until the training is finished. Let

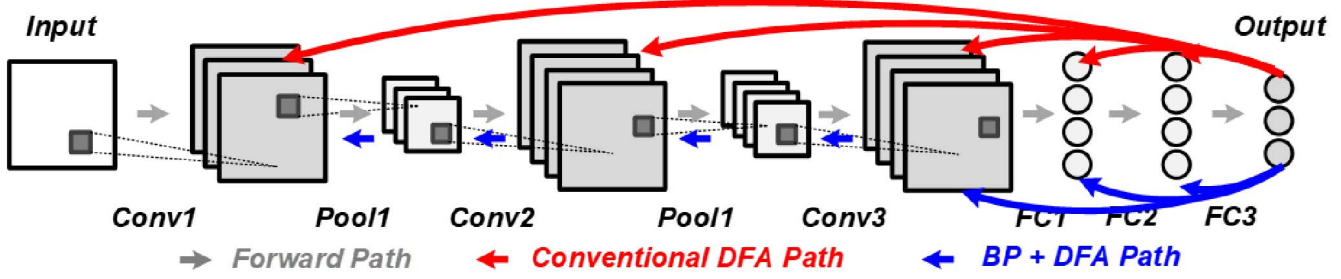


Figure 2. Conventional DFA in CNN and Proposed DFA based Error Propagation

$R_{i+1,i}$ be the feedback weight of the i^{th} layer, the propagated error in the FA is calculated as

$$e_i = (R_{i+1,i}^T e_{i+1}) \odot f'(i_{i+1}) \quad (5)$$

, and the other procedures such as gradient generation and weight updating are same as BP does.

2.3. Direct Feedback Alignment

Even though, both the BP and the FA propagate errors from the last layer to the first layer in order, the DFA ([13]) directly propagates errors from the last layer to other layers. If the number of neurons in the last layer and the i^{th} layer, are represented by N_L and N_i respectively, the size of the feedback weight is determined as $N_i \times N_L$. Let, i^{th} layer's feedback weight in the DFA be D_i^T , then the error is calculated as

$$e_i = (D_i^T e_L) \odot f'(i_{i+1}) \quad (6)$$

One of the interesting characteristics is that there is no data dependency between different errors because the DFA propagates the errors directly from the last layer. This characteristic gives the opportunity of parallel processing in the DFA based error propagation operation.

Compared with the BP, DFA showed similar training performance for the multi-layer perceptron (MLP) structure. Moreover, DFA requires fewer computations because N_L is generally smaller than the number of neurons in the intermediate layers. However, the DFA dramatically degrades the accuracy when it applied in the CNN training as shown in Figure 1. Furthermore, the elements of the intermediate feature map are all connected with the last layer neurons, so it requires much more computations compared with BP based CNN training. In summary, the DFA's computational efficiency can be induced in the MLP training, but not in the CNN.

3. Our Approach

As mentioned in Section 2, the DFA is efficient for MLP training, because it can be computed in parallel without accuracy degradation. However, this advantage is diminished

when the DFA is applied in the CNN. To solve this problem, we propose the new training method to make the DFA applied to the CNN. In addition, the initialization method of the feedback weight is suggested for the fast and stable learning curve. At last, we propose the advanced training algorithm, binary direct feedback alignment (BDFA), which shows high computing efficiency and robust training performance in various conditions.

3.1. CDFA: CNN Training by Combining both BP and DFA

Generally, CNN consists of the convolutional layers and fully-connected (FC) layers. The two different kinds of layers have different roles. For example in object classification, convolutional layers are considered as the feature extractor by using the characteristics of the convolution computation. In contrast, FC layers receive the result of the feature extractor and judge what the object is. However, in the initial iterations of the BP based CNN training, training the convolutional layers can be ambiguous because the FC layers cannot be considered as a good object classifier. During some iterations of BP, the weight of the FC layers will be changed and the convolutional layers should be adaptive to the changed FC layers. In other words, convolutional layers can be confused if the propagated error's domain is changed for every iteration.

If the error is propagated from the last FC layer to convolutional layer through the constant domain shifting, the training of the convolutional layer can be more stable than conventional BP. From this motivation, we use DFA instead of BP in the FC layers. As shown in Figure 2, the network maintains the BP in the convolutional layers but adopts the DFA for the FC layers. Since there is no data dependency among the FC layers, the DFA can directly propagate the error to the FC1. Since the DFA uses fixed feedback weight for error propagation, convolutional layers do not have to be adaptive to various errors which are derived from the different domain. In this method, the error propagation in the convolutional layers can be started even though the errors are not propagated for the remained FC layers. Therefore, the error propagation of both the convolutional layers and

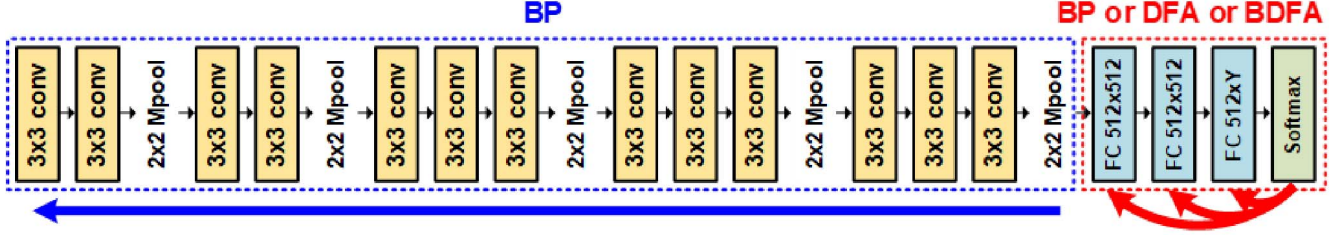


Figure 3. Overall network configuration for training

the FC layers can be computed in parallel right after DFA is done for the first FC layer.

[13] shows that the randomly initialized feedback weight can be used for DFA based DNN training. However, both the FA and the DFA are sensitive to the initialization method of the feedback weight because it affects the training accuracy and the convergence speed significantly. As shown in [10], it is observed that the batch-normalization (BN) seems to make the FA become not sensitive to the initialization method, but it is still a problem because of the slow convergence. To make the DFA robust to the initialization method, we fixed the feedback weight as the multiplication of the transposed weights in multiple layers. To sum up, the feedback weight of the i^{th} layer, D_i , can be calculated as

$$D_i = W_{L,L-1} \dots W_{i+2,i+1} W_{i+1,i} \quad (7)$$

, and finally, the error propagation operation can be summarized as follow.

$$e_i = (D_i^T e_L) \odot f'(i_{i+1}) \quad (8)$$

The suggested initialization method is suitable for other various functions such as sigmoid, tanh and ReLU. Moreover, other normalization or optimization methods such as BN and dropout ([16]) are also applicable with the proposed initialization with the equation (7).

3.2. Binary Direct Feedback Alignment (BDFA)

The DFA needs the feedback weight addition to the forward weight matrix, and it occupies a larger memory to store both two matrices. Moreover, the DFA has the chance to be computed in parallel, but it requires much larger memory bandwidth. Since the throughput in the FC computing is vulnerable to the memory bandwidth, loading the additional feedback weight degrades the throughput compared with the unlimited bandwidth case.

To solve the throughput bottleneck problem caused limited bandwidth, we propose the binary direct feedback alignment (BDFA) algorithm. BDFA uses the binarized feedback weight, B_i , whose values are determined as either +1 or -1. In other words, the B_i can be stored as a single bit to represent only the sign value of the feedback weight's element. As a result, required memory to store the B_i is

reduced by 96.9% compared with the 32-bit floating point representation which now becomes the general numeric representation in CPU or GPU. As we determined in the DFA, BDFA's feedback weight, B_i can be similarly defined by a modification of the equation (7). B_i is determined as

$$B_i = \text{sign}(D_i), \quad e_i = (B_i^T e_L) \odot f'(i_{i+1}) \quad (9)$$

, when the $\text{sign}(\cdot)$ is the function which indicates the sign value of each element. The difference between equation (7) and (9) is only whether the $\text{sign}(\cdot)$ is applied or not. By applying equation (9), BDFA shows faster and stable training convergence compared with the random initialization case. The effect of the binarization and initialization will be discussed in section 4.

4. Experiments

In this section, we compared the training accuracy of the conventional BP and suggested training algorithm. We measured the relative accuracy by training CNN from the scratch in CIFAR-10 and CIFAR-100 dataset ([8]). We used the network configuration as described in Figure 3. The base network follows the VGG-16 configuration, but has one additional FC layer. To sum up, it consists of 13 convolutional layers with BN and ReLU activation functions, followed by three FC layers without BN. The number of neurons in the last FC layer is determined by the number of classification categories in each different dataset. In the BP based approach, both the convolutional layers and the FC layers are trained by using BP. In contrast, the training method of the last three FC layers is substituted with the DFA or the BDFA to measure the performance of the proposed training algorithm. The simulation was based on mini-batch gradient descent with the batch size 100 and uses momentum([17]) for the optimization method. The parameters of the network are initialized as introduced by [4], and the learning rate decay and the weight decay method is adopted. Other hyper parameters are not changed for fair comparison.

4.1. CNN Training from the Scratch

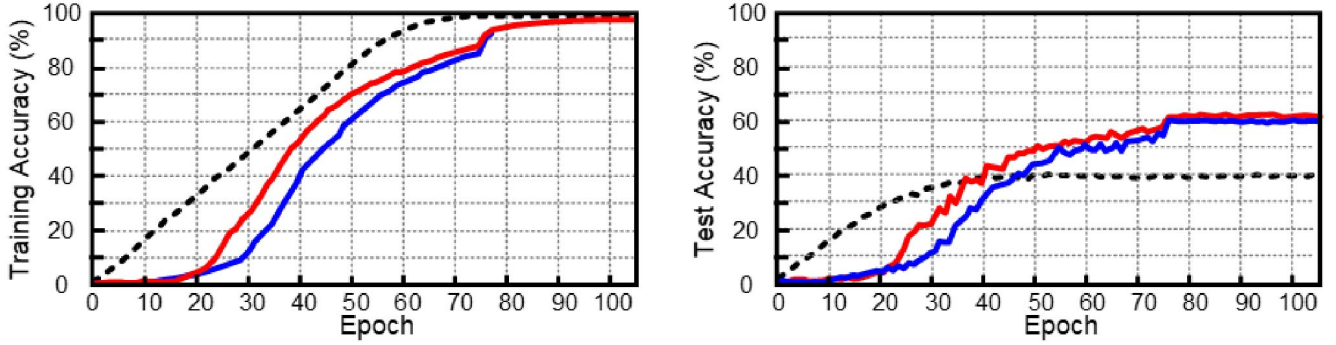
The CNN training with the DFA and BDFA, are renamed as CDFA and CBDFA respectively for the simple explana-

Table 1. CNN Training Result in CIFAR-10 & CIFAR-100 (Small Learning Rate)

CIFAR 10	BP	BP w/ BN	CDFA Random	CDFA Eq (7)	CDFA w/ BN	CBDFA Random	CBDFA Eq (9)	CBDFA w/ BN
Top5	98.63	98.24	98.42	98.55	98.56	98.63	98.88	98.83
Top1	81.11	76.91	88.68	86.36	87.41	89.39	87.65	86.46

CIFAR 100	BP	BP w/ BN	CDFA Random	CDFA Eq (7)	CDFA w/ BN	CBDFA Random	CBDFA Eq (9)	CBDFA w/ BN
Top5	67.80	63.91	77.05	72.82	77.55	75.07	71.92	76.85
Top1	40.29	37.80	61.42	48.24	55.11	59.92	47.48	54.47

< CIFAR-100 Training Result – Random Initialization & Small LR>



< CIFAR-100 Training Result – Determined Initialization & Small LR>

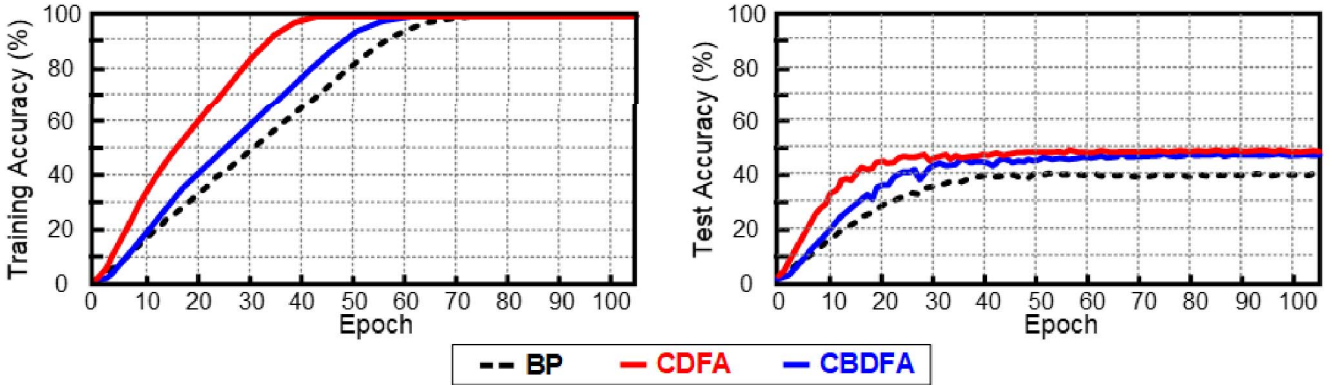


Figure 4. Training and Test Accuracy with Proposed Training Algorithm

tion. Table 1 shows Top5 and Top1 test accuracy after the CNN training is done in the two different CIFAR datasets. In this simulation, there is no data augmentation to make an environment which has a limited dataset. This condition can examine whether the algorithm is robust to the training in the small dataset. To sum up, only 50,000 images in the CIFAR-10 and CIFAR-100 is only used for DNN training and the other 10,000 images are tested to evaluate the test accuracy. As a result, both the CDFA and the CBDFA show higher test accuracy compared with the conventional BP even though the feedback weight is randomly initial-

ized. In CIFAR-10, the CDFA and CBDFA are 7.5% and 8.3% higher in Top1 test accuracy than the BP respectively. The accuracy improvement by the CDFA and the CBDFA seems much more remarkable in CIFAR-100. As shown in Figure 4, the training curve of the CDFA and CBDFA is much slower, but they achieve 21.3% and 19.6% better performance respectively compared with the BP.

However, the feedback weight with the random initialization has critical problems for training. One of the problems is the slow training curve described in Figure 4. DFA requires time to be adaptive to the randomly initialized feed-

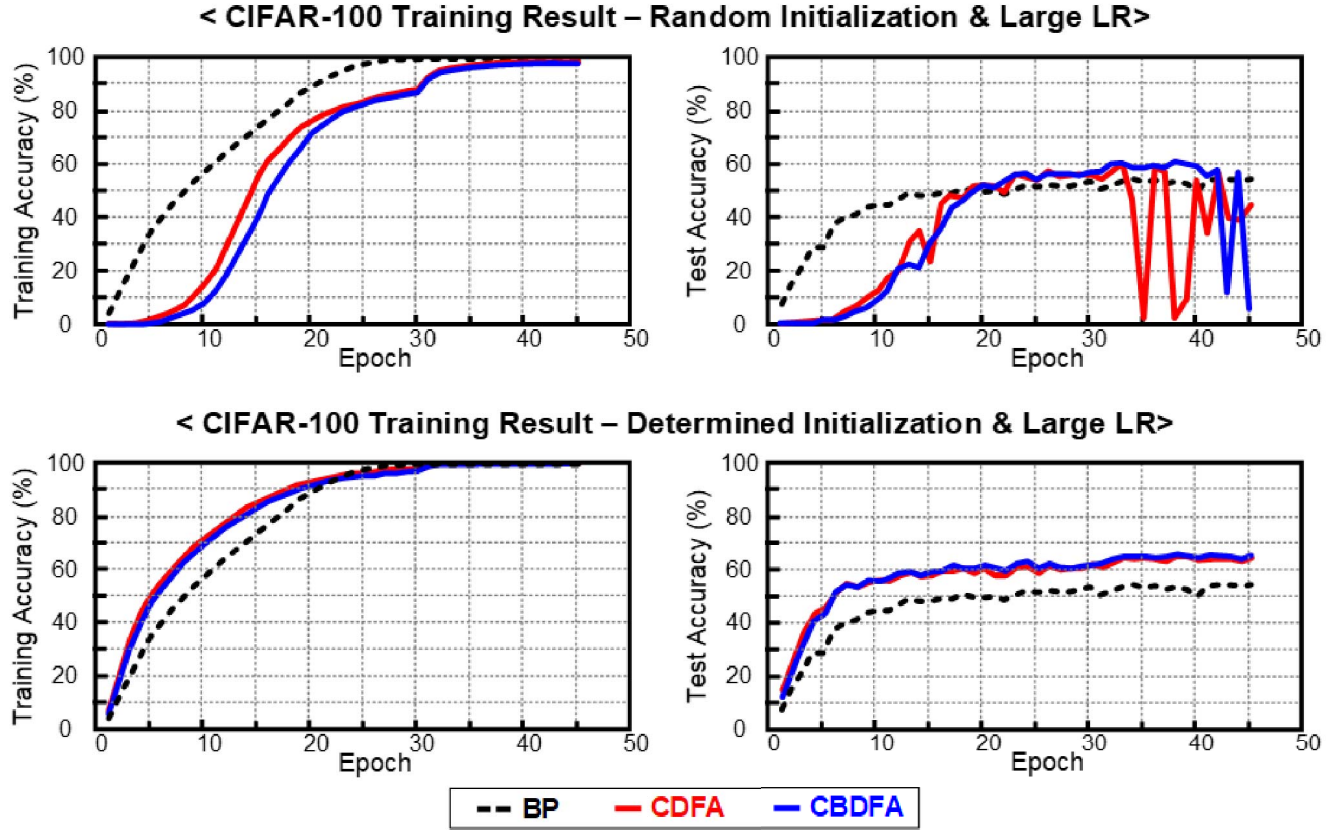


Figure 5. Training and Test Accuracy with Proposed Training Algorithm

Table 2. CNN Training Result with Data Augmentation (CIFAR-10)

	w/o Data Augmentation			w/ Data Augmentation		
	BP	CBDFA	Conv. only Training	BP	CBDFA	Conv. only Training
Top5	99.15	99.07	98.84	99.33	99.49	99.46
Top1	82.33	87.35	82.06	87.97	90.13	88.48

back weights, so it takes a long latency to be converged. In the BP approach, we generally take the larger learning rate to make the training faster. However, the test accuracy of the DFA and BDFA is swung up and down dramatically when the large learning rate is applied. Moreover, it still spends a long time to converge. In this problem, the initialization with the equation (7) and (9) can be useful to solve the learning speed and stability problem. After the feedback weight is initialized by the proposed equations, it shows faster and more stable convergence characteristic as shown in Figure 5. When the proposed initialization method is combined with the large learning rate, it shows the best training performance compared with the other results.

When the dataset is augmented with the flipping and cropping, the training performance of the BP and CBDFA becomes higher than before. The simulation uses the large learning rate, and CBDFA takes the initialization with equa-

tion (9). In table 2, the performance of the CBDFA shows the highest accuracy compared with not only BP but also the training suggested by [5]. It trains only the convolutional layers, and the parameters of the FC layers are fixed. Even considering the data augmentation, CBDFA still shows higher training accuracy compared with the other two methods. As a result, CBDFA seems robust to the size of the dataset.

There are some interesting observations in our simulation. First of all, the CBDFA shows negligible accuracy degradation compared with the CDFA based training. Sometimes, the CBDFA has a rather better performance than the CDFA case. Refer to the learning curve described in both Figure 4 and Figure 5, the CBDFA's learning speed is slightly degraded, but the final training results are approximately the same. Therefore, CBDFA can improve the training performance and take the hardware benefits such

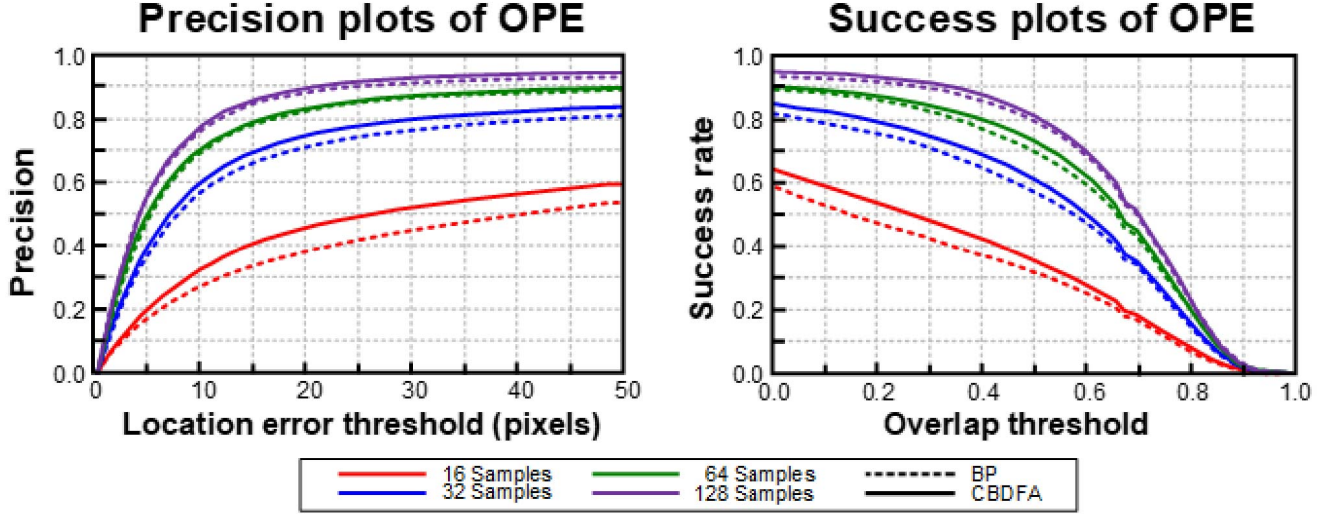


Figure 6. Object Tracking Result with the Proposed Training Algorithm, CBDFA

as smaller memory bandwidth by adopting binarized feedback weights. The second one is the effect of the BN. Even though the suggested initialization method achieves the fast and stable training, it has a little accuracy degradation compared with the random initialization method. This accuracy degradation can be reduced when the BN layer is added after the FC layer. This result is counter characteristic compared with the BP case because the BP shows the worse training result when the BN layer is followed right after the FC layer. To sum up, the equation (7) and (9) are much more powerful when the BN is followed after the convolutional layer.

4.2. Example of Online Learning with Small Dataset: Object Tracking

[12] suggested online FC learning based object tracking algorithm, MDNet. In the MDNet, both the convolutional layers and FC layers are pre-trained with the VOT ([7], [6]) and OTB ([18]) object tracking dataset. However, the last layer of the FC layers is randomly initialized for the new tracking task. The convolutional layers do not need to be trained during the tracking but the FC layers are fine-tuned by using the BP. To apply BDFA to FC online learning with the small dataset, we replace the BP by the BDFA for FC layers in the MDNet.

We compared the object tracking performance by drawing the precision and the success plots of the one-pass evaluation (OPE) in the OTB dataset. As shown in Figure 6, the object tracking with the BDFA based online learning shows similar performance compared with the BP case. However, the BDFA shows better performance than BP when the batch size becomes smaller. Since the object tracking application is very sensitive to the online learning speed and the BDFA has a chance to be computed in parallel, BDFA based

online learning becomes much more beneficial than conventional BP. Moreover, the BDFA can dramatically reduce the required data transaction in error propagation because of the fewer neuron interconnections and the binarization. As pointed out in the paper, [2], BP based online learning is inefficient for online learning in the devices which have limited memory bandwidth, computing resources, and small dataset. In this case, the effect of the BDFA can be maximized because of its profits.

5. Hardware Implementation

5.1. Overall Architecture

We have fabricated BDFA based DNN learning processor [3] with the 65nm CMOS technology. The main computation required for DNN inference is accelerated by DNN Core (DNNC). DNNC fetches the feature maps from OMEM and computes matrix multiplication based on multiply-accumulate units (MACs). The feature maps stored in OMEM are encoded by run-length compression [1] and it can reduce the required memory space by only remembering nonzero data. Proposed processor can support DNN training with 32 batchsize and uses dedicated momentum optimizer for updating weights. Direct error propagation core (DEPC) is in charge of error propagation to support DNNC during the online learning. It computes new errors by using the proposed BDFA algorithm.

5.2. Direct Error Propagation Core

Since the feedback weight of the BDFA indicates only the sign values, the matrix multiplication will be substituted with the adder-only computation. We implemented adder-tree based error propagation computing block instead of designing MAC array. DEPC can reduce the overall comput-

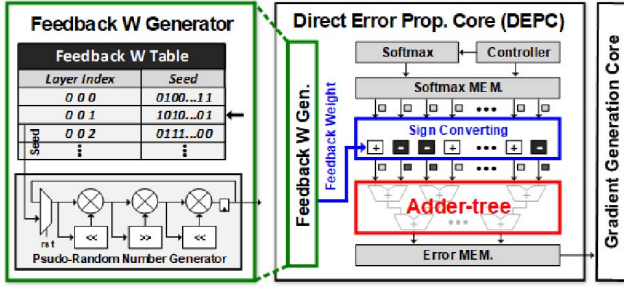


Figure 7. ASIC Implementation of the DNN Learning Processor with the Proposed BDFA Algorithm

ing power required for error propagation because the adder shows much lower power consumption compared with the multiplier. In addition, we designed built-in pseudo-random number generator (RNG) to generate feedback weights. With the built-in RNG, it does not need to load feedback weights from external memory such as DRAM and we can reduce power caused by DRAM access. Both adder-based error computing and RNG based feedback weight fetching achieve the 35.3% lower power consumption compared with BP based error propagation core design.

6. Conclusion

In this work, we propose the new DNN learning algorithms to maximize the computation efficiency without accuracy degradation. We adopt one of the training method, DFA and combine it with the conventional BP. The combination of the DFA and the BP shows much better test accuracy in CNN training through the simulation in the CIFAR-10 and CIFAR-100 dataset. BDFA takes one step further, binarizing the feedback weight while maintaining a similar performance compared with the full-precision DFA. The stability problem induced in the DFA and BDFA simulation (Figure 5) can be solved by the new feedback weight initialization method, equation (7) and (9). The BDFA is also simulated in the object tracking application, and it shows better tracking results compared with the conventional BP based online FC tuning. Additionally, we proved the efficiency improvement effect of BDFA by implementing ASIC and possible to design low-power online learning processor.

References

- [1] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energyefficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52:127–139, 2017.
- [2] D. Han, J. Lee, J. Lee, S. Choi, and H.-J. Yoo. A 141.4 mw low-power online deep neural network training processor for real-time object tracking in mobile devices. *In Proceedings of International Symposium on Circuits and Systems (ISCAS)*, 2018.

- [3] D. Han, J. Lee, J. Lee, and H.-J. Yoo. A 1.32 tops/w energy efficient deep neural network learning processor with direct feedback alignment based heterogeneous core architecture. *Symposium on VLSI Circuits*, 2019.
- [4] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- [5] E. Hoffer, I. Hubara, and D. Soudry. Fix your classifier: the marginal value of training the last weight layer. *In International Conference on Learning Representations*, 2018.
- [6] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, and G. F. et al. The visual object tracking vot2014 challenge results. *European Conference on Computer Vision Workshops (ECCVW)*, 2016.
- [7] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, and T. V. et al. The visual object tracking vot2013 challenge results. *International Conference on Computer Vision Workshops (ICCVW)*, 2013.
- [8] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In Proceedings of Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.
- [10] Q. Liao, J. Z. Leibo, and T. Poggio. How important is weight symmetry in backpropagation? *arXiv:1510.05067*, 2015.
- [11] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random feedback weights support learning in deep neural networks. *Nature Communications*, 7, 2016.
- [12] H. Nam and B. Han. Learning multi-domain convolutional neural networks. *In Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [13] A. Nokland. Direct feedback alignment provides learning in deep neural networks. *In Proceedings of Neural Information Processing Systems (NIPS)*, 2016.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [17] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. *In Proceedings of International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.
- [18] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *Transactions on Pattern Analysis and Machine Intelligence*, 37:1834–1848, 2015.