Analysis and Development of Hardware compatible Neural Networks

Independent Project - 1

Dhirubhai Ambani Institute of Information and Communication Technology
Aditya Shah(202003045@daiict.ac.in)
Mentor: Prof. Bakul Gohel(bakul_gohel@daiict.ac.in)

Abstract

Neural networks are most commonly trained with a back-propagation algorithm, which uses gradients for back-propagating the errors in different layers. As per the recent discovery, feedback alignment shows that the propagation of error backwards need not be symmetric with the weight used for forward propagation, random feedback weights perform as well as back-propagation because neural networks learn how to make feedback useful. This kind of learning is one step forward for biologically plausible algorithms because errors remain local to the layer. Back-propagation is highly efficient over the von Neumann architectures but it has scaling limitations. The idea of local computation draws our attention towards neuromorphic computing. Experiments show that the performance of feedback alignment methods on MNIST and CIFAR10 is as good as back-propagation.

Index Terms

Neural Networks, Feedback alignment, Direct Feedback Alignment, Neuromorphic Computing, Back-propagation.

I. Introduction

THE back-propagation algorithm (BP) has been very successful in training deep neural networks for supervised learning. Currently, this approach has a limited number of true rivals because of its effectiveness and simplicity, while there are some substitutes.

Various variations of Boltzmann machine learning are physiologically inspired techniques for neural network training. The techniques simply modify the weights using signals that are readily available locally. Combining these techniques with BP fine-tuning can result in effective discriminating performance.
Recently, feedback alignment (FA), a unique training approach, was introduced [9]. The authors demonstrate that symmetry between the feed-forward weights and the feedback weights, which are used to back-propagate the gradient, is not necessary. To lower the error, the network learns how to employ fixed random feedback weights. It means that the network learns how to learn, which is a really strange outcome.

Asymmetrically weighted back-propagation was also investigated. A key finding of this study is that strong performance can be achieved with a large relaxation of the weight symmetry restriction.
For several reasons, the back-propagation algorithm is not biologically feasible :

1) It needs symmetric weights.
2) It necessitates distinct stages for learning i.e. layers learn recursively.
3) The learning signals are not local but have to be propagated backwards from the output units.

II. Neuromorphic Computing

Neuromorphic computers are non-von Neumann computers made of synapses and neurons that are inspired by the structure and functions of real brains. Von Neumann computers consist of separate central processing units (CPUs) and memory units, which store data and instructions. In Neuromorphic computers, processing and memory are governed by neurons and synapses. Programs in neuromorphic computers are defined by the structure of the neural network.

Neuromorphic computers have some operational differences:

- Highly Parallel processing: They are inherently parallel, neurons and synapses can be operated simultaneously.
- Collocated Processing and memory: This architecture is different from Von Neuman's architecture where processing and memory units are separated.
- Inherent scalability: They are meant to be inherently scalable as adding additional neuromorphic chips increases the number of neurons and synapses.
- Stochasticity: They work with the concept of randomness such as the firing of neurons.

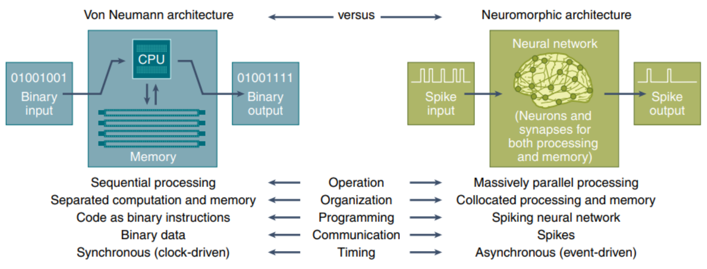
Von Neumann architecture
versus
Neuromorphic architecture
CPU
Memory
01001001 Binary input
01001111 Binary output
Neural network
(Neurons and synapses for both processing and memory)
Spike input
Spike output
Sequential processing
Separated computation and memory
Code as binary instructions
Binary data
Synchronous (clock-driven)
Operation
Organization
Programming
Communication
Timing
Massively parallel processing
Collocated processing and memory
Spiking neural network
Spikes
Asynchronous (event-driven)

Fig1. Comparison of Von Neumann Architecture and Neuromorphic Architecture

In a typical deep-learning network, some signals flow along a forward path through multiple layers of processing units from the input layer to the output, while other signals flow back from the output layer along a feedback path. Forward-path signals perform inference while the feedback path conveys error signals that guide learning. The gradient calculation of one layer depends upon the very next layer's weight matrix which is known as the weight transportation problem.

Recent studies in learning algorithms have focused on the intersection between neuroscience and machine learning by studying more biologically plausible algorithms. One of the main family of methods is known as feedback alignment, which employs distinct forward and feedback synaptic weights. These algorithms also solve weight transportation problems.

III. Feedback Alignment Algorithms

A. Feedback Alignment(FA)

The weight updates are computed in the same fashion as in backpropagation, but the backward weight matrix is a random matrix. Initializing B to have the same distribution and magnitude scale as W helps improve network training convergence.

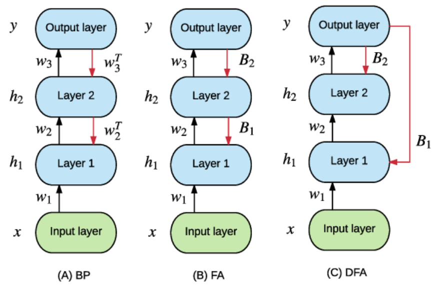

Fig.2 Back-propagation (BP), feedback-alignment (FA) and direct feedback-alignment (DFA) model. Black arrows represent forward activation paths. Red arrows indicate error (gradient) propagation paths.

B. Direct Feedback Alignment(DFA)

While the weight update in FA is computed recursively across layers, it is possible to project the error propagation by directly backward the derivative of the loss at the last layer.

Instead of relying on precisely computed gradients, DFA employs fixed random feedback connections. These connections remain constant during training, and the network adapts to use them effectively.

DFA is considered more biologically plausible than backpropagation because it doesn't require symmetric weights, making it a potentially more realistic model of how learning might occur in biological neural networks.

C. Sparse Direct Feedback Alignment(SDFA)

In terms of training fully connected networks, Direct Feedback Alignment was a significant advancement. The weight transport issue can be avoided and new hardware for neural network training can be made available by substituting a single random matrix for the backward propagation from deeper layers.

In small networks, DFA seems feasible since each neuron requires connections to only a few error signals. However, as the size of the network increases, the size of the feedback matrix also increases and in effect, each weight update needs more information. To relax this problem, we introduce SDFA where the error signal is fed back to all the hidden layer neurons through a highly sparse feedback matrix.

SDFA allows neurons to compute errors using fewer error signals from the last layer.SSDFA is an updation over SDFA, which can enable local learning requiring only a single global error to be transferred per neuron while incurring a small loss of accuracy. Following is the comparison among BP, DFA and SSDFA:

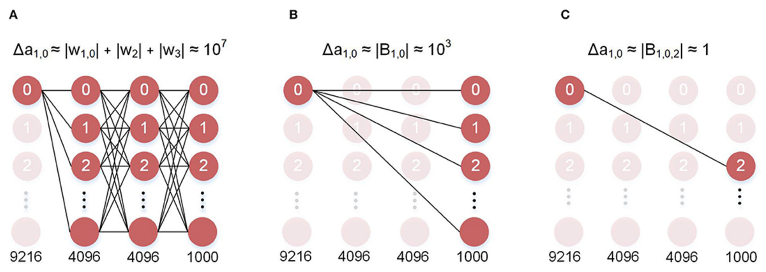
A
$\Delta a_{1,0} \approx |W_{1,0}| + |W_2| + |W_3| \approx 10^7$
B
$\Delta a_{1,0} \approx |B_{1,0}| \approx 10^3$
C
$\Delta a_{1,0} \approx |B_{1,0,2}| \approx 1$
0
1
2
9216
4096
4096
1000

Fig3. Data movement comparison of error assignment algorithms.

Now, we will look at the mathematical formation for BP, FA and DFA in next section.

IV. Mathemtical Formulation

Let's divide the dataset into batches and (x, y) be one of the batches from the dataset i.e., MNIST or CIFAR10. Here, x is the input and y is the targeted output. For simplicity, we are assuming that the neural network has two hidden layers and one output layer. These batches are used to train neural networks. Let W_i be the weights connecting the layer below a unit in layer i and b_i be the column vector with biases for the units in hidden layer i and $f(.)$ be an activation function for hidden layers and $f_y(.)$ be an activation function for the output layer. We can formulate a neural network as follows:

$$Layer\ 1 : a_1 = W_1 x + b_1, h_1 = f(a_1)$$

$$Layer\ 2 : a_2 = W_2 h_1 + b_2, h_2 = f(a_2)$$

$$Layer\ 3 : a_y = W_3 h_2 + b_3, \hat{y} = f_y(a_y)$$

Assume that hidden layers use a sigmoid or logistic activation function and a cross-entropy loss function is used for the output layer. Hence, the loss function can be written as follows for the output layer:

$$L = -\frac{1}{N} \sum_{m,n} y_{mn} \log(\hat{y}_{mn}) + (1 - y_{mn}) \log(1 - \hat{y}_{mn})$$

where $\hat{y}_{mn}$ is m^{th} predicted value for n^{th} input batch, y_{mn} is m^{th} actual value for n^{th} input batch, N is a batch size.

In the back-propagation approach for the training of neural network we will update the weights and biases of each layer according to the gradients that are calculated, for i^{th} layer :

$$W_i = W_i - \alpha(dW_i)$$

$$b_i = b_i - \alpha(db_i)$$

Where α is a learning rate. For each layer, gradients can be calculated as follows:

$$Layer\ 3 : da_3 = \frac{\partial L}{\partial a_3} = e = (\hat{y} - y),\ dW_3 = \frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial a_3} * \frac{\partial a_3}{\partial W_3} = e{h_2}^T,\ db_3 = da_3$$
$$Layer\ 2 : da_2 = \frac{\partial L}{\partial a_3} = \left({W_3}^T e\right) \odot f'\left(a_2\right),\ dW_2 = \frac{\partial L}{\partial W_2} = da_2 \cdot {h_1}^T,\ db_2 = da_2$$
$$Layer\ 1 : da_1 = \frac{\partial L}{\partial a_1} = \left({W_2}^T da_2\right) \odot f'\left(a_1\right),\ dW_1 = \frac{\partial L}{\partial W_1} = da_1 x^T,\ db_1 = da_1$$

where $\odot$ is an element-wise multiplication operator.
The calculation of the current layer's gradient depends upon the next layer's gradient value. for example, da_1 is a function of ${W_2}^T$ and da_2. The Neuromorphic architecture supports massive parallel processing while the Von Neumann architecture performs operations through sequential processing. The gradient should be calculated locally to avoid sequential and enhance parallel processing.
For feedback alignment(FA), replace the weight matrix of the next layer with the appropriate size of a fixed random matrix B_i, Initializing B_i to have the same distribution and magnitude scale as W_i helps improve network training convergence:

$$da_2 = \left({B_2}^T e\right) \odot f'\left(a_2\right)$$
$$da_1 = \left({B_1}^T da_2\right) \odot f'\left(a_1\right)$$

For Direct Feedback Alignment(DFA), it is an updation over FA. The error propagation over the layers is projected to the derivative of the loss at the last layer e.

$$da_2 = \left({B_2}^T e\right) \odot f'\left(a_2\right)$$
$$da_1 = \left({B_1}^T e\right) \odot f'\left(a_1\right)$$

Now, da_i can be calculated locally using a fixed random matrix and derivative of the loss at the last layer. Using da_i, dW_i is also calculated easily. In the case of large networks, the weight transportation problem is often an issue. FA algorithms avoid it by computing gradients locally.

V. Experimental Results

The experiments are done over the artificial neural network(ANN) and convolutional neural network(CNN). The ANN 1x800 tanh network is trained over the MNIST dataset. ANN is trained using both back-propagation and direct feedback alignment methods. On each epoch randomly 200(batch size) images (28 x 28) are picked from the 60000 sample input images. The network is trained for 100 epochs with the learning rate $\alpha = 10^{-4}$. The following results are obtained for the same:

1x800 tanh ANN	**BP**	**DFA**
training accuracy(%)	97.48	95.00
test accuracy(%)	92.10	92.46

TABLE I
Result of ANN over MNIST dataset trained through BP and DFA

From the above results, DFA achieves almost the same training accuracy as BP. In the testing part, DFA performs slightly better than BP. In the case of DFA, the random matrices for each layer are fixed and generated from the same distribution of weight matrix. This fixed quantity enhances the learning of ANN. Hence, it performs well on the test dataset. The following plot is the comparison of training error between BP and DFA(in %):

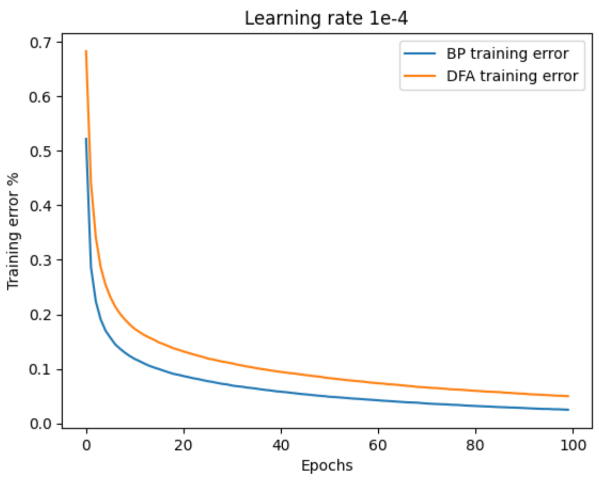

Fig1. Comparison of Back-propagation and Direct Feedback Alignment

The CNN model with the following configuration is used on the CIFAR10 dataset :

Dataset	Convolutional network
CIFAR 10	Conv $(5 \times 5\ 1 \times 1\ 96)$
	Pool $(3 \times 3\ 2 \times 2)$
	Conv $(5 \times 5\ 1 \times 1\ 128)$
	Pool $(3 \times 3\ 2 \times 2)$
	Conv $(5 \times 5\ 1 \times 1\ 256)$
	Pool $(3 \times 3\ 2 \times 2)$
	FC 2048
	FC 2048
	Softmax 10

TABLE II
IN THE TABLE EACH ROW CORRESPONDS TO A LAYER IN CNN. THE FOLLOWING NOTATION IS USED FOR EACH LAYER: CONV(KERNEL SIZE, STRIDE SIZE, NUMBER OUTPUT CHANNELS) AND POOL(KERNEL SIZE, STRIDE SIZE)

The mini-batch training is used for BP, DFA and SSDFA. The batch size used is 64 with each image of dimension (32 x 32 x 3), picked randomly from 60000 images in the CIFAR10 dataset. The network is trained for 50 epochs and with the constant learning rate $\alpha = 10^{-4}$.

The following results are obtained for BP, DFA and SSDFA:

CONV tanh CNN	**BP**	**DFA**	**SSDFA**
training accuracy(%)	99.12	99.38	99.26
test accuracy(%)	73.53	79.97	78.94

TABLE III
RESULT OF CNN OVER CIFAR10 DATASET TRAINED THROUGH BP, DFA AND SSDFA

The training of the CNN model through BP, DFA and SSDFA performed equally well over the training dataset. For the test dataset, SSDFA and DFA performed nearly the same but back-propagation performed litter poor on the same parameters.

VI. CONCLUSION

We can greatly reduce the amount of data movement in the backward pass using feedback alignment algorithms(SSDFA). While there will be significant savings in memory accesses and multiply-and-accumulate (MAC) operations operations, the key improvement is in data movement. However, to take advantage of this, a von Neumann architecture is not an ideal choice for bio-mimetic algorithms or feedback alignment algorithms. Neuromorphic architecture is a better alternative.

All methods were able to fit the training set on all experiments performed on MNIST and Cifar-10. The performance on the test sets also remains the same.

One can remove the constraint imposed by back-propagation, which requires the backward pass to visit every neuron from the forward pass. Even in situations when the feedback path isn't connected to the forward path, learning can still occur.

References

[1] Yoshua Bengio, Dong-Hyun Lee, Jörg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. CoRR, abs/1502.04156, 2015.

[2] Geoffrey E. Hinton and Terrence J. Sejnowski. Optimal Perceptual Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1983.

[3] Direct Feedback Alignment Provides Learning in Deep Neural Networks, Arild Nøkland , Trondheim, Norway

[4] Activation Sharing with Asymmetric Paths Solves Weight Transport Problem without Bidirectional, Sunghyeon Woo Jeongwoo Park Jiwoo Hong Dongsuk Jeon,Seoul National University

[5] Opportunities for neuromorphic computing algorithms and applications Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa1, J. Parker Mitchell, Prasanna Date and Bill Kay.