Direct Feedback Alignment with Sparse Connections for Local Learning

Preprint · January 2019

CITATIONS
0

READS
35

4 authors, including:

Brian Crafton
Georgia Institute of Technology
3 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Abhinav Parihar
Georgia Institute of Technology
25 PUBLICATIONS **264** CITATIONS
SEE PROFILE

Arijit Raychowdhury
Georgia Institute of Technology
194 PUBLICATIONS **3,388** CITATIONS
SEE PROFILE

Direct Feedback Alignment with Sparse Connections for Local Learning

Brian Crafton, Abhinav Parihar, Evan Gebhardt, Arijit Raychowdhury
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA

Abstract

Recent advances in deep neural networks (DNNs) owe their success to training algorithms that use backpropagation and gradient-descent. Backpropagation, while highly effective on von Neumann architectures, becomes inefficient when scaling to large networks. Commonly referred to as the weight transport problem, each neuron's dependence on the weights and errors located deeper in the network require exhaustive data movement which presents a key problem in enhancing the performance and energy-efficiency of machine-learning hardware. In this work, we propose a bio-plausible alternative to backpropagation drawing from advances in feedback alignment algorithms in which the error computation at a single synapse reduces to the product of three scalar values, satisfying the three factor rule. Using a sparse feedback matrix, we show that a neuron needs only a fraction of the information previously used by the feedback alignment algorithms to yield results which are competitive with backpropagation. Consequently, memory and compute can be partitioned and distributed whichever way produces the most efficient forward pass so long as a single error can be delivered to each neuron. We evaluate our algorithm using standard data sets, including ImageNet, to address the concern of scaling to challenging problems. Our results show orders of magnitude improvement in data movement and $2\times$ improvement in multiply-and-accumulate operations over backpropagation. All the code and results are available under https://github.com/bcrafton/ssdfa.

I. Introduction

The demise of Dennard scaling [11] and decline of Moores Law [27] have exposed the fundamental scaling limitations of the von Neumann models of computing. This transition is accompanied by the realization that in a fast evolving, socially interconnected world, we are observing a seismic shift in the amount of unstructured data that need to be processed in real-time [25] which has heralded the third wave of Artificial Intelligence and the exponential growth of Machine Learning in data-analytics, real-time control, computer vision, robotics and so on. We expect that intelligent systems of the future will be limited by the energy growth of data movement rather than compute. Therefore we need fundamentally new approaches to sustain the exponential growth in performance beyond the end of the current road-map. In particular, we observe that new computing models that deal with data analytics have compute and storage interleaved in a fine grained manner - not separated as in the von Neumann world. Moving forward, computing technology will heavily penalize separation of data and compute and we need to marry them in better ways to handle emergent applications.

The idea of computing locally on data finds its inspiration from the human brain where local processing and updates is preferred to global movement of data. Hence, neuromorphic computing seeks to fundamentally improve the power efficiency of cognitive systems by bringing ideas inspired from biology to electronic hardware, while maintaining the high accuracy and performance that statistical methods have provided. In particular, hardware implementations of deep neural networks (DNNs) and their many variants, either in complementary metal oxide semiconductor (CMOS) [6] or emerging technologies [21], rely on arrays of spatial processors where near-memory [23] or in-memory [8] logic computes inference from layers of neurons connected via dense or sparse synaptic connections. This is schematically shown in Figure 1. As opposed to a von Neumann architecture (Figure 1.A) where all the synaptic weights for a particular layer must be loaded in the memory to compute the activations of the successive layers, in a distributed implementation (figure 1.B) the weights and logic reside locally and avoid the memory bottleneck. The data movement is minimized, with each neuron computing its activation and sending that information to the next layer. In spite of the success of such spatial processing in the inference mode, such a architecture fails to deliver high efficiency when functions that require global information or weights of multiple layers are implemented. This is particularly evident during the training of DNNs where back-propagation (BP) and gradient-descent (GD), which are

quintessentially statistical techniques, have found wide-spread adoption [18]. In BP with GD, the transpose of the weights of the deeper layers in a network are needed to compute error gradients and the weight updates of the shallower layers, thus requiring global movement of data. This problem is commonly referred to as the weight transport problem [13]. As the networks become deeper to keep up with the complexity of the applications, the weight transport problem becomes exacerbated.

While neural networks require many expensive floating point calculations, the cost of data movement is higher [6]. To make the problem worse, for every MAC operation multiple reads and potentially a write are required. Furthermore, the cost of loading data from off chip DRAM is orders of magnitude more expensive than loading from spatially local on chip memories. [17] show through simulation the breakdown of energy consumption in the different layers of a CNN. The majority of the energy consumption comes from memory reads and writes, rather than MAC operations. The strategy these accelerators employ is data reuse. Because the cost of data movement is so high, it is important that each word of data is reused as many times as possible before being flushed from the cache. Optimizing for data reuse, these accelerators can achieve several times better efficiency over data flows that do not use local reuse. Neuromorphic engineering is another research vector which attempts to minimize data movement by borrowing learning rules from the human brain. For example, in spiking neural networks (SNNs), spike timing dependent plasticity (STDP) has gained popularity because of its local update rule. STDP uses local information, available at a synapse, and has been shown to perform well in unsupervised learning [12]. [9] present a new SNN implementation with tools to perform supervised learning. Rather than seeking to optimize current neural network architectures using data reuse, [9] uses biological constraints on data movement and calls for new approaches to learning. The fundamental constraint is that a weight can only be accessed and modified by its corresponding destination neuron. This constraint promotes local learning because the only data movement that occurs are activation and error signals. With this constraint in place, we define data movement as information a neuron must send or receive for each weight update. Unfortunately, this constraint is violated by backpropagation because of the weight transport problem. To minimize data movement, it is imperative that new local learning rules satisfy this constraint while achieving performance similar to backpropagation.

One such promising recent work is Feedback Alignment (FA) [22] which has shown that we might be able to bypass the weight transport problem while achieving the same accuracy that BP achieves. FA uses fixed random feedback weights to propagate the errors back through the layers of a DNN rather than using the actual current network weights to compute the partial error. Consequently, the weights in the shallow layers of the network no longer need information of the weights of all the deeper layers. Building on top of this, [26] proposes Direct Feedback Alignment (DFA), where it was shown that the feedback to shallow layers need not be propagated through all the layers. Instead the error signal can be fed back to the shallower layers through completely random linear transformations. This further reduces the amount of information required to update the weights in the network. To further describe the weight transport problem and its relationship with local learning [4] describes the concept of a learning channel. The learning channel is a physical way in which information about targets and deep weights are transported in the network. backpropagation uses the forward channel in the backward direction. Using the targets and deep weights we compute partial errors to update the rest of the network. Feedback Alignment instead uses a separate channel to transmit the weights and in doing so avoids the weight transport problem.

Other promising algorithms include target propagation [20] and local error learning [24]. Target propagation bypasses the weight transport problem while still solving the credit assignment problem like the feedback alignment algorithms. Instead of computing a loss gradient, a target value is assigned to each feed forward layer which is done using auto encoders. Local error learning generates local errors at each layer using linear classifiers with fixed random weights. In doing so, errors at the output of the network do not need to be send back and instead local objective functions are solved rather than a global one. Local error learning bypasses the weight transport problem, solves the credit assignment problem, and also does not even need to send errors to the hidden layers. All these algorithms have been shown to perform similarly on benchmarks. While their performance is impressive, they still suffer the same problems when scaled to larger networks. Each of the algorithms fails to match the performance of BP as both the complexity of the network and difficulty of the benchmark increases. [5] highlighted this problem showing that when applied to problems like ImageNet, the biological algorithms failed to come close to backpropagation. In this work, we show this problem comes from convolutional layers rather than the complexity of the network. In fact, when trained layers are re-used from backpropagation and the fully connected layers are trained using feedback

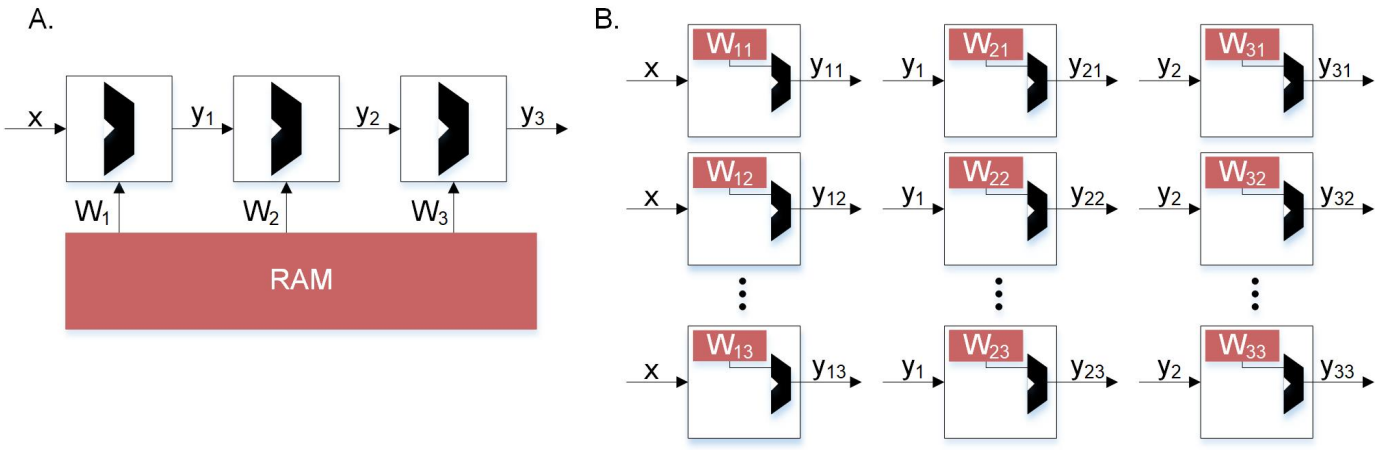

Fig. 1: Hardware implementations of inference. Comparing von Neumann architecture with distributed memory architecture to avoid bottleneck. **A**. Inference constrained by von Neumann architectures. In traditional backpropagation, weight updates not only depend on error but also the other weights. This prevents a distributed architecture. **B.** Unconstrained inference. Using direct feedback alignment, inference can be distributed and parallel because weight updates depend only on the error and random feedback values.

alignment the performance is equal to that of backpropagation.

Although DFA does not require the error signals to be transformed by the weights of the deeper layers of the network, it does require feedback via dense random matrices. This requires a significant number of operations and does not mimic the true local learning, that is a hallmark of bio-mimetic networks. In this paper, we show a modified version of DFA, sparse direct feedback alignment (SDFA) where we show that sparse feedback of the error signals can result is negligible drop in the network's performance but significantly reduces the computational complexity during learning. In an extreme version of SDFA, we demonstrate that even a single error feedback signal can enable the network to learn and allows true local learning with negligible performance loss. We call this *single connection SDFA* (SSDFA). We systematically study, through empirical demonstrations, the role of sparsity and rank of the feedback matrix on the network's performance. Our work demonstrates that SSDFA inherits the computational advantages of local learning similar to bio-mimetic networks, while maintaining the high accuracy of BP with GD (hitherto simply referred to as BP).

II. Sparse Direct Feedback Alignment

Direct feedback alignment marked a remarkable step in learning for fully connected networks, by replacing the partial errors from deeper layers of the network by a single random matrix. In small networks, DFA seems feasible as each neuron would require connections to only a few error signals. However as the size of the network increases, the feedback matrix, though fixed, also increases in size and each weight update in effect needs more and more information. As an illustrative example, consider a three layered network, with 100 hidden neurons which can be trained for classifying the MNIST handwritten data-set. Because each image size is 28 by 28 pixels and their are 10 classes, our network size will be 784 (28x28) - 100 - 10. In this example, each of the 100 hidden neurons require connections to each of the 10 error signals at the output. While this may seem plausible for MNIST, in case of the Alexnet network [16] or VGG [28] that is used to classify the Imagenet data-set, [10] the number of connections becomes much larger. The Imagenet data-set classifies 1000 different classes of images which are re-sized to 227 by 227 pixels. In the case of DFA, each of the 4096 neurons in the first fully connected layer requires a connection to each of the 1000 errors. This quadratic increase in connections prevents DFA from scaling to larger problems without incurring significant computational penalty compared to a truly local learning rule.

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 with a sparse feedback matrix enables each neuron to compute its error using a fewer number of error signals. Consequently, as we will demonstrate, any hardware design that implements SDFA based learning requires a considerably lower number of error propagation, rendering it more efficient both in terms of throughput and power.

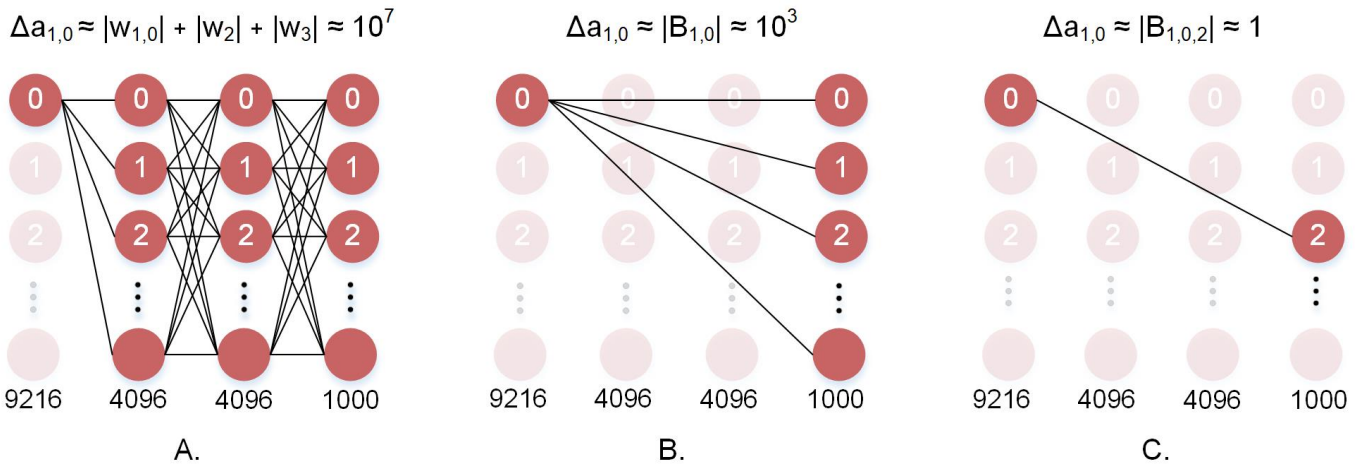

Fig. 3: Neuron-level memory dependence of the different algorithms. **A. Backpropagation:** The error at the first layer is computed using all the weights in the deeper layers. This is the weight transport problem. **B. DFA:** the error at the first layer is only $e \cdot B$. This solves weight transport, however it still requires 1000 FB weights. **C. SSDFA:** The error at the first layer is dependent only on a single error and a single feedback weight.

Incorporating the constraint of sparsity in the feedback matrix affects the performance of the learning algorithm itself. We empirically demonstrate that sparsity plays a negligible role on the network's accuracy as long as the feedback matrix is full-rank or near-full-rank. This leads to SSDFA which can enable true bio-mimetic local learning with negligible loss of accuracy.

We define a feedback as sparse when the values in the feedback matrix which model the connections between the errors and the hidden neurons are mostly 0, or the connection does not exist. It is known that biological networks share similar properties where each neuron updates its weight using local values. In this work, we use the percentage of 0 valued connections to quantify sparsity. In figure 2 we present an example of a sparse feedback matrix where each of the 25 hidden neurons is connected to one of the 10 errors. In this case the sparsity is 90% because only 10 errors are used. We also demonstrate that even an extremely sparse feedback matrix, with 99.9% sparsity, can be used to achieve high accuracy with on the ImageNet dataset.

In figure 3, we schematically compare BP, DFA, and SDFA for a prototypical fully connected network consisting of four layers of neurons. The figure shows the number of connections the pre-synaptic neuron in the first hidden layer needs with the rest of the network for computing a weight update. In BP, the neuron is dependent on each and every neuron it has a direct or indirect connection to. Hence the weight update of the synapse is dependent on all model weights deeper in the network. The number of connections grows as the network gets deeper leading to the weight transport problem. On the other hand in DFA, the neuron is dependent only on the number of errors of the network. While this decouples the forward and backward pass and relieves the weight transport problem, complex networks with many output neurons will still have a large number of feedback connections. In the proposed SDFA, a weight update is dependent only on the few errors it is connected to. Consequently, even if the network scales in both depth and complexity, the number of connections for updating a synaptic weight is low. In the proposed extreme scenario for SDFA, we name SSDFA, each neuron receives only one error signal and can successfully update its weight with a single error signal.

Using only a single error per neuron, we are able to greatly reduce the amount of data movement in the backwards pass. While there will

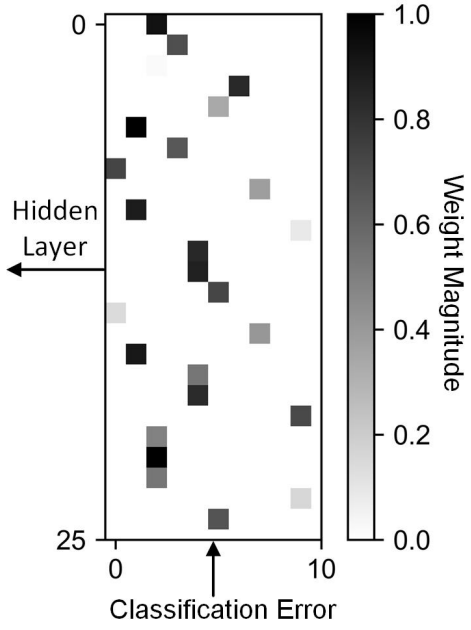

Fig. 2: A sparse feedback matrix where each hidden neuron is connected to a single error. Only one of the ten connections between a neuron is non-zero and trainable.

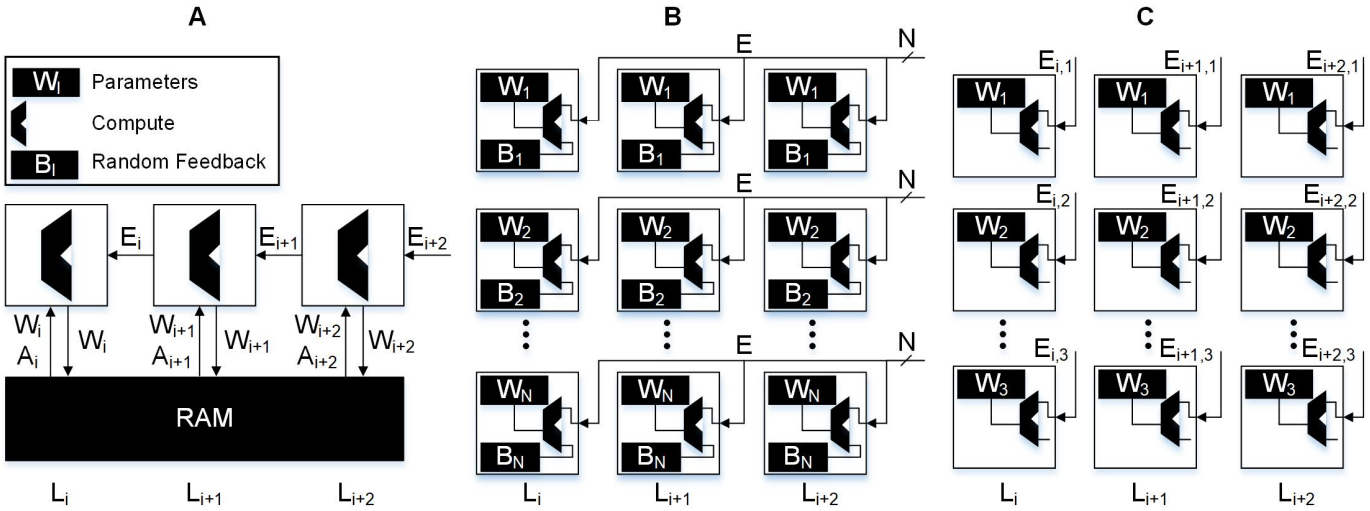

Fig. 4: Data movement through the substrate of different algorithms. **A. Backpropagation:** In traditional von Neumann architectures weights and activations from the forward pass must be accessed from main memory. The majority of data transfer occurs when moving the large weight matrices from main memory to compute. **B. DFA:** In a local learning implementation only the error vector needs to be sent to the neurons. In this case the neuron must receive all N errors and store an additional N random feedback weights. **C. SSDFA:** In the single sparse connection implementation of DFA, only a single error needs to be sent to each neuron and only a single random feedback constant needs to be stored. This reduces the bandwidth requirement and feedback weight storage by a factor of N.

TABLE I: Data Movement Comparison of Error Assignment Algorithms

Method	Reads	Writes	MACs	Movement
BP	$\|W\| + \|A\|$	$\|W\|$	$2\|W\|$	$\|W\| + \|A\| + \|E_{i+1}\|$
DFA	$\|W\| + \|B\|$	$\|W\|$	$\|W\| + \|B\|$	$\|E\|$
SDFA	$\|W\| + \|b\|$	$\|W\|$	$\|W\| + \|b\|$	$\|e\|$

be significant savings in memory accesses and MAC operations, the key improvement is in data movement. However, in order to take advantage of this, a von Neumann architecture is not an ideal choice for bio-mimetic algorithms. In particular, recent advances in spatial array processors with near-memory computing can achieve significant advantages in both performance and energy-efficiency [7], [14]. In figure 4 we show a comparison of the different algorithms and the data movement required in the backwards pass. We illustrate in figure 4.A to 4.C how the data movement greatly reduces. With backpropagation on a von Neumann architecture, 4.A, we must read the weights and activations to and from the main memory. DFA on a neuromorhpic architecture 4.B, relaxes the weight transport problem and keeps the read, write, and MAC operations local to the neuron itself with the only data movement occurring when the error is sent backwards. Lastly, in figure 4.C the error sent backwards is only a single scalar reducing data movement in the backwards pass to its minimal form. In table I we compare the number of reads, writes, MACs, and data movement accross the different algorithms as a function of the neurons and weights in the network. $|A|$, $|W|$, $|B|$, and $|E|$ refer to the number of neurons, weights, feedback weights, and errors, respectively. We also use $|b|$ and $|e|$ to refer to the reduced number of weights and errors from the sparse feedback matrix.

III. Mathematical Formulation

The primary contribution of this work is to show the benefits of using sparse feedback that can eventually enable local learning in neuromorphic hardware implementations. We show, empirically on standard networks and data-sets that sparsity, even extreme-sparsity (as in the case of SSDFA), results in negligible loss of accuracy; while reducing data movement, during training, by orders of magnitude.

We investigate BP, DFA, and SDFA for the fully connected network architecture where all neurons are connected to all other neurons in the previous and next layers. The feed forward computation can be written as:

$$y_1 = W_1 \cdot x, a_1 = f(y_1) \tag{1}$$

$$y_2 = W_2 \cdot a_1, a_2 = f(y_2) \tag{2}$$

$$y_n = W_n \cdot a_{n-1}, a_n = f(y_n) \tag{3}$$

Where x is the feature vector and W_i is the weight matrix connecting layer $i-1$ to layer i ($y_0 = x$). The dot product of x and W_i gives us y_i, and applying the non-linear activation function f results in the activation at neuron i, a_i. Each of these algorithms computes the error at a specific layer. BP computes the error at a layer *l*, δA_l, by transposing the weight matrices W, multiplying by the gradient of the activation function, and propagating the error back 5. Where $\odot$ is element-wise multiplication.

$$\delta a_n = e \tag{4}$$

$$\delta a_2 = W_2 \cdot \delta a_3 \odot f^{'}(a_2) \tag{5}$$

$$\delta a_1 = W_1 \cdot \delta a_2 \odot f^{'}(a_1) \tag{6}$$

BP requires all the deeper weights in the network in order to compute the error at a layer earlier in the network. DFA bypasses this dependence, only requiring a random matrix B to be multiplied with the error vector:

$$\delta a_n = e \tag{7}$$

$$\delta a_2 = B_2 \cdot e \odot f^{'}(a_2) \tag{8}$$

$$\delta a_1 = B_1 \cdot e \odot f^{'}(a_1) \tag{9}$$

In the proposed SDFA, the update rule is identical to equations 7 8 9, except with an added constraint that B is sparse. At the neuron level this represents an important detail in the physical implementation of neuromorphic hardware. In DFA the update rule for a neuron i in layer l:

$$\Delta a_{li} = B_{li} \cdot e \cdot f^{'}(a_{li}) \tag{10}$$

To update a single neuron's weights in DFA, a linear combination of all the errors is required. This is modeled by the dot product in equation 10. This is computationally challenging for complex networks. For example, when we use VGG network on the ImageNet data-base, the number of multiply-and-accumulate (MAC) operations for each neuron is 1000 corresponding to the number of output classes. On the other hand, in SDFA, when B is sparse, the number of MACs required for each update will be significantly less, depending on the number of non-zero entries in the corresponding row of B. In an extreme scenario, when each row of B has only 1 non-zero entry, the update rule reduces to:

$$\Delta a_{li} = B_{lij} \cdot e_j \cdot f^{'}(a_{li}) \tag{11}$$

Thus the error propagated to a neuron is the product of just three scalars. Of which, all are locally available when the error received by the network is considered a local variable. The significance of this comes from past work in local learning and bio-plausible 3-factor learning rules. Locality is a constraint on the learning rule that requires each synaptic weight must be

updated with only locally available variables. [3] simplifies learning rules to the following forms:

$$\delta W_{ij} = f(O_i, O_j, W_{ij}) \tag{12}$$

$$\delta W_{ij} = f(T_i, O_i, O_j, W_{ij}) \tag{13}$$

An example of equation 12 is Oja's rule. In Oja's rule the synapse uses only the activations of the neurons that it is connected to, and its own state variable. Commonly referred to as the *3 factor learning rule* [3], equation 13 requires a target value. It should be noted that, while backpropagation can be simplified to this form, the target value computed by backpropagation is not a local variable due to the weight transport problem, as discussed above. In the case of direct feedback alignment, the target values can be computed using local variables if all errors and feedback weights can be placed close to the neuron. However as we have seen, as the size of the network scales this is not feasible. When the feedback matrix is made sparse (SDFA), we can scale the size of the network without requiring dense feedback connections. When only a single feedback connection is used (SSDFA) the 3 factor update rule is satisfied where the target value is a function of only a single error and a single feedback weight. This results in a bio-mimetic learning approach, similar to STDP, but can be used for supervised learning to achieve accuracy similar to BP.

IV. Results

To evaluate the performance of SDFA and SSDFA, we benchmark the proposed algorithms on standard vision data-sets vis-a-vis BP and DFA. We empirically evaluate the accuracy of the models with respect to the desired characteristics of the feedback matrix.

A. Benchmarks and Network Architectures

In our simulations, we use 4 standard benchmarks of varying complexity that have been used in prior work. These benchmarks are MNIST [19], CIFAR10, CIFAR100 [15], and ImageNet [10]. ImageNet is by far the most exhaustive data-set and like [5] we find that using this benchmark reveals scaling issues related to the feedback alignment algorithms. For each of these benchmarks we use two networks. For MNIST, CIFAR10, and CIFAR100 we use one fully connected network and one convolutional network which are further described in table II. For Imagenet we use two standard convolutional networks: Alexnet [16] and VGG16 [28], which have produced high accuracy with BP. The data-sets and the corresponding model architectures are summarized in table II.

B. SDFA and SSDFA for Fully Connected Network

We start by reporting our findings on fully connected networks. The three properties of the feedback matrix that we study are:

1) **Rank:** The number of linearly independent vectors in the feedback matrix.
2) **Sparsity:** The percentage of zero-valued weights in the feedback matrix connecting each hidden layer to the classification error. We are particularly interested in the case where each hidden neuron is connected to a single error (SSDFA).
3) **Angle:** The angle between the vectorized weight matrix (W) and the vectorized feedback matrix (B).

The evaluation platform is setup in Tensorflow. For MNIST and CIFAR10 there are 10 output neurons and hence 10 error signals. In these networks, the maximum possible rank and sparsity of the feedback matrix is 10. For each combination of rank and sparsity we simulate the models 10 times and statistically observe the accuracy of the network. To ensure that the network has feedback from all ten errors, the product of rank and sparsity must be equal to or greater than 10. Consequently, we are unable to generate results for cases where the product is less than ten. If the network does not have feedback from all ten errors, then the network's performance is impacted, not because of lower rank or higher sparsity, but because only some of the error signals are being propagated back.

Rank: Empirically, we observe that the rank of a feedback matrix has the largest impact on the resulting accuracy of the network. As we increase the rank, the accuracy of the network increases and finally saturates. In figure 5 we show our results

TABLE II: Network Architectures

Dataset	Fully Connected Network	Convolutional Network
MNIST	FC 400 Softmax 10	Conv (3x3 1x1 32) Conv (3x3 1x1 64) Pool (2x2 2x2) FC 128 Softmax 10
CIFAR (10, 100)	FC 1000 FC 1000 FC 1000 Softmax (10, 100)	Conv (5x5 1x1 96) Pool (3x3 2x2) Conv (5x5 1x1 128) Pool (3x3 2x2) Conv (5x5 1x1 256) Pool (3x3 2x2) FC 2048 FC 2048 Softmax (10, 100)
ImageNet	-	Alexnet ([16])
ImageNet	-	VGG16 ([28])

The format for Convolutional layers is as follows: Conv (kernel size, number output channels, stride size). The format for Pool layers is as: Pool (kernel size, stride size). The networks used for the CIFAR10 and CIFAR100 datasets were both derived from [26].

for the MNIST and CIFAR10 data sets. In figure 5.A and 5.B, we show the accuracy and angle versus the rank of the feedback matrix, for varying sparsity. For the MNIST data-set, we observe that the test accuracy saturates, as expected to 97.5% and is maximum for the full-rank matrix. However for CIFAR10, the accuracy continues to increase with rank and is maximum for full-rank feedback matrices. This shows that the rank of the feedback matrix is a critical design parameter and the feedback needs to be a full-rank matrix to maximize the network's accuracy.

Sparsity: Our results show that sparsity of the feedback matrix has very little impact on the resulting accuracy of the network in both, the MNIST and CIFAR10 networks (Figure 5). We also see negligible impact of sparsity on the angle between the feedback matrix used, and the resulting feed forward weights from the learning algorithm (Figure 6). For many resource constrained systems, a small difference in accuracy for large improvements in power and performance is an excellent trade-off. We observe that highly sparse feedback matrices (SSDFA), even with a single error feedback performs very well – while significantly reducing the computational demand, as we describe later.

Angle: [22] shows that the angle between the gradient of BP and FA aligns to an angle less than 90 degrees. Instead of looking at the angle between the gradient of the two algorithms, we look at the angle between the vectorized version of resulting weight matrix, and the vectorized feedback matrix. In case of propagating errors to shallow layers of the network, the corresponding weight matrix used for the angle calculation is the product of all the weights matrices following this layer. Hence, for the angle calculation for a layer l in a n layer network, the angle is given by:

$$\angle(B_l, w) = \angle(B_l, \prod_{i=l+1}^{n} w_i) \tag{14}$$

In figure 5 we show a strong correlation of accuracy and angle between the feedback matrix and the weight matrix. This correlation is more apparent for CIFAR10 because it does not plateau as it approaches a full-rank matrix. This correlation is also illustrated in figure 6 because neither the angle nor the accuracy changes significantly with the sparsity.

In Table IV we summarize the results for fully connected networks. We show the results collected from running BP, DFA, and SSDFA on MNIST, CIFAR10, and CIFAR100 with fully connected networks of different sizes. The network parameters are shown in the Method Section. BP and DFA results are similar to previous work [26], which show that DFA performs nearly the same as BP for the fully connected networks. Our results show that SSDFA performs similar to both DFA and BP.

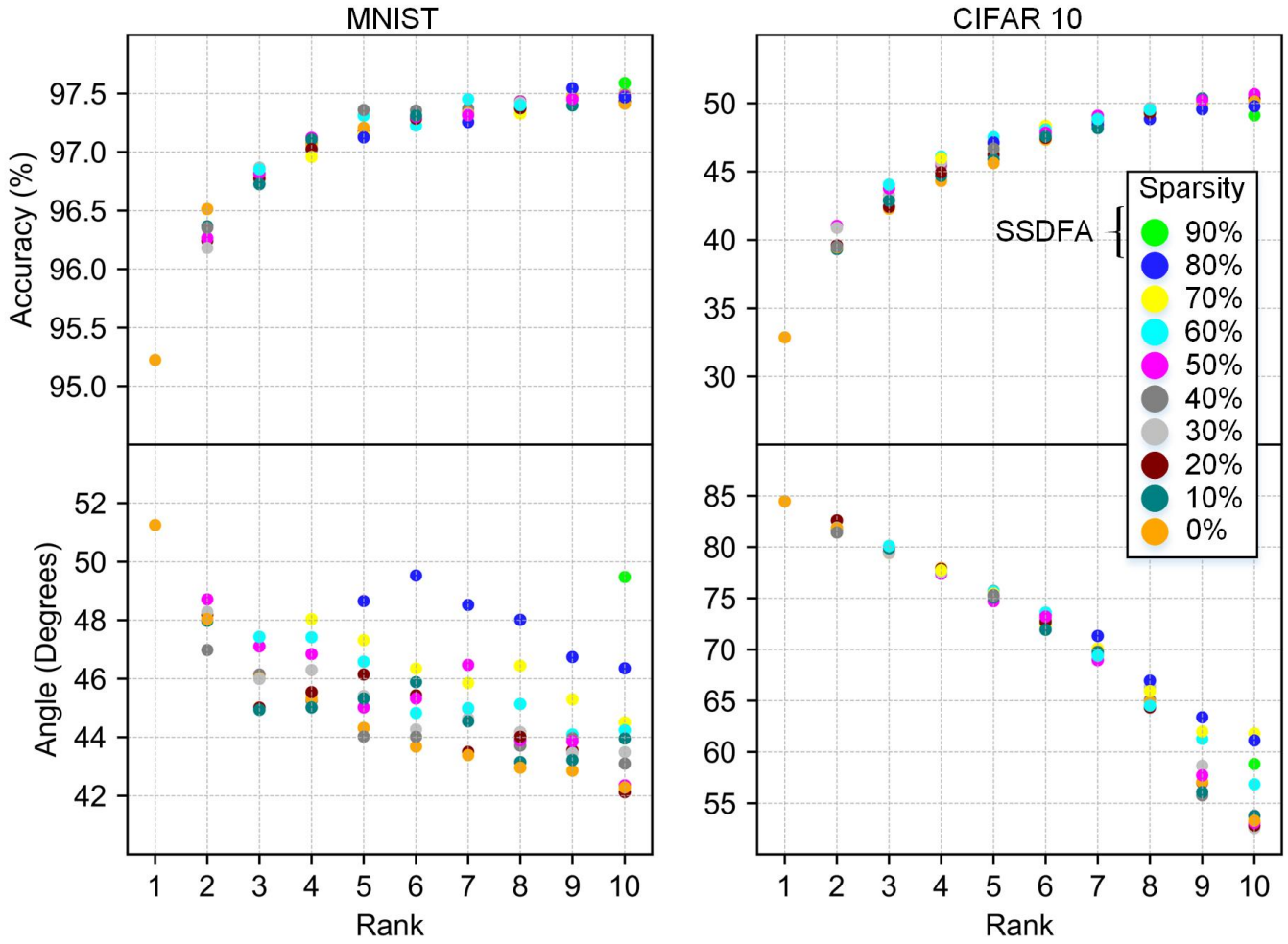

Fig. 5: Accuracy and angle (in degrees) versus rank for MNIST and CIFAR10 fully connected networks. Data points are grouped by sparsity, and averaged over 10 different simulations.

TABLE III: Test Accuracy (in %) for Fully Connected Networks

Benchmark	BP	DFA	SSDFA
MNIST	98.2	97.8	97.5
CIFAR10	59.9	58.9	58.6
CIFAR100	33.1	29.4	28.8

C. SDFA and SSDFA for Convolutional Neural Networks

Earlier work [26] show that direct feedback alignment can be used in convolutional neural networks (CNNs). However, we note, similar to [5], as we scale up the network and the problems complexity increases, DFA fails to match the accuracy of BP. For the convolutional networks we show our results in table IV. Consistent with previous work, the CNNs we use for MNIST and CIFAR10 show similar performance for both BP, DFA as well as SSDFA. However as the problem and network complexities scale to CIFAR100 and ImageNet, the gap between BP and DFA grows to where their performance is no long comparable. In all these cases, SDFA performs similar to DFA.

We attribute this problem to the convolutional layers in the network. The convolutional architecture introduces extra constraints on the weight space, and as such, results in stronger constraints on the feedback matrix which can be used. This can be seen in figure 7, where we show the filter weights for the first convolutional layer resulting from training Alexnet on ImageNet using BP and DFA. The resulting filters show completely different patterns. The BP filters show a well defined spatial structure while

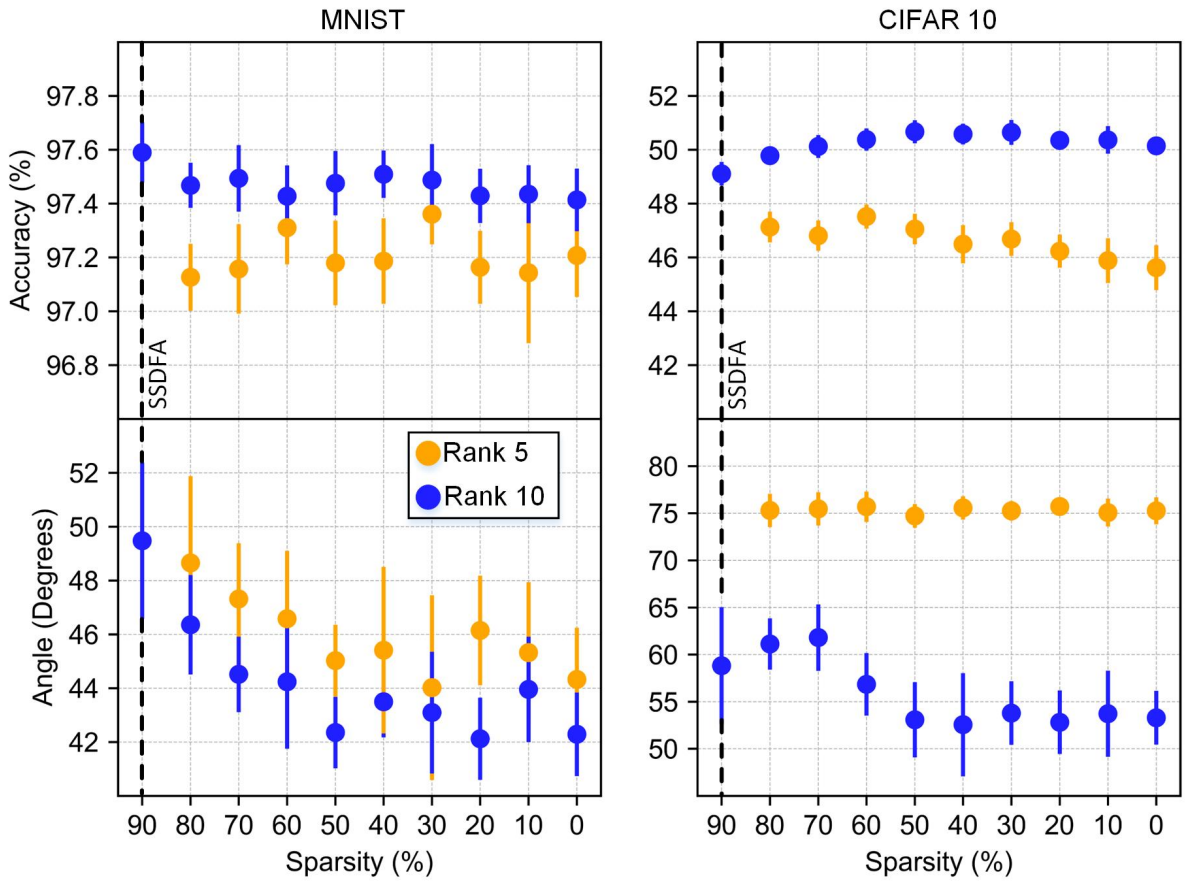

Fig. 6: Accuracy and angle versus sparsity. Results from rank 5 and rank 10 are shown with bars showing the standard deviation for 10 different simulations.

TABLE IV: Test Accuracy (in %) for the Convolutional Neural Networks. Accuracy results are slightly less for ImageNet than originals because minimal pre-processing and data augmentation techniques were used.

Benchmark	BP	DFA	SSDFA
MNIST	99.1	98.9	98.8
CIFAR10	79.6	72.3	73.1
CIFAR100	51.0	43.7	41.8
ImageNet (Alexnet)	53.6	6.2	2.8

the filters from DFA are random. This shows that DFA cannot learn convolutional filters as efficiently as BP for large and complex networks.

One way to alleviate this problem is to use transfer learning where the convolutional filters are reused from BP and only the fully connected layers are trained. We show that by reusing these convolutional filters and training the fully connected layers using DFA and SSDFA we can achieve the same performance that BP achieves. In table V we show the results for five different benchmarks of different sizes. In all of our benchmarks the performance of DFA and SSDFA is nearly identical and in some cases better than BP.

D. Computational Advantage

The primary advantage of using SSDFA is that it greatly reduces data movement in the backwards pass. Further, SSDFA also reduces the number of multiply-and-accumulate (MACs), memory-reads, and memory-writes. Computationally, this is

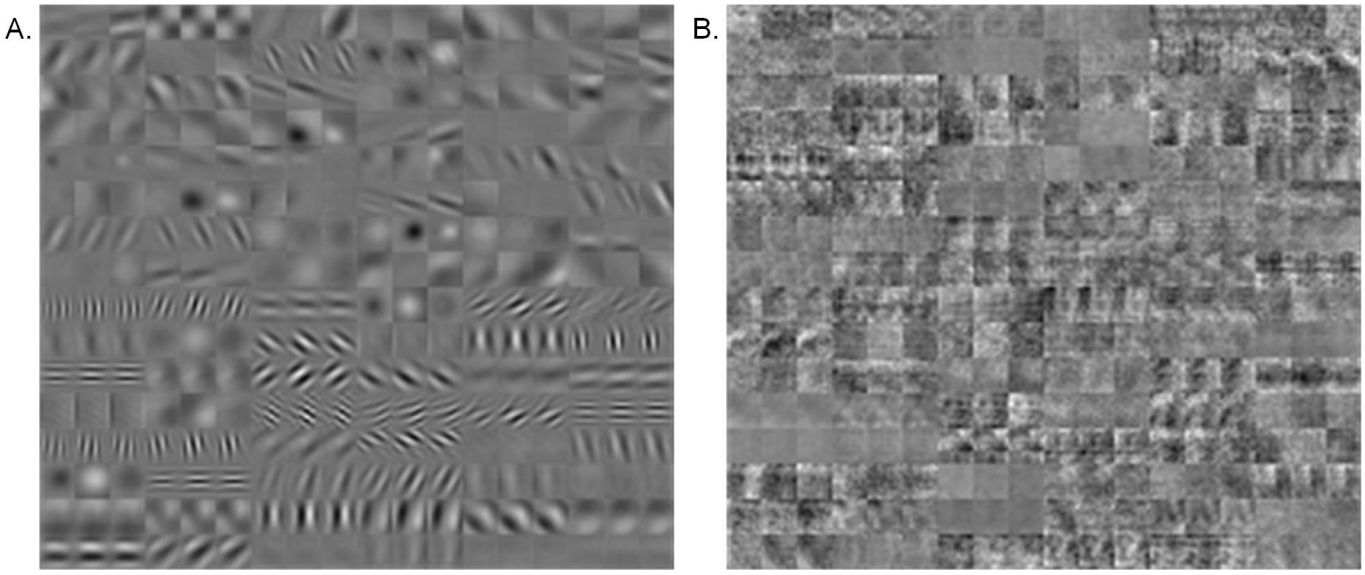

Fig. 7: Filters acquired from training Alexnet with BP (A) and DFA (B). Filters for BP show shape and spatial structure, while filters from DFA are random.

TABLE V: Test Accuracy (in %) for after Transfer Learning of the Convolutional Filters followed by SSDFA for the Fully Connected Layers. Accuracy results are slightly less for ImageNet than originals because minimal pre-processing and data augmentation techniques were used.

Benchmark	BP	DFA	SSDFA
MNIST	99.1	99.1	99.0
CIFAR10	77.1	77.8	78.0
CIFAR100	48.2	49.0	48.2
ImageNet (Alexnet)	49.0	48.8	46.3
ImageNet (VGG)	65.8	65.3	64.5

motivated by biology where memory and compute are interleaved and global movement of data is minimal. New neuromorphic architectures [1], [2] seek to apply this constraint to avoid the communication overhead. However, as we have noted earlier, BP violates this constraint. Rather than requiring local information, it requires information from the weights of the deeper layers in the network. SSDFA minimizes data movement so that each neuron requires only a single error and a single random feedback weight. Using our benchmarks we have shown that for fully connected networks, SSDFA performs similarly to BP. On CIFAR100 and ImageNet, we show the MAC and data movement savings when implemented in a spatial-array architecture similar to [9], as shown in figure 8. We define data-movement as any data that must be sent outside the neuron. In the backwards pass this is the error information sent to each neuron in SSDFA and DFA. For BP implemented on a von Neumann architecture, this is equivalent to all the data that must be transported to and from the main memory. In 8.A we show the number of MACs across SSDFA, DFA, and BP for ImageNet.

The reduction in the number of MACs from BP to SSDFA is nearly a factor of two. In table I we show that the number of MACs reduces from $2|W|$ to $|W| + |B|$. The number of MACs required to compute the error at a hidden layer reduces from $|W|$ to $|b|$ but the number of MACs to compute the partial error at each weight remains $|W|$. If the feedforward matrix, W, was also sparse then this bottleneck could be reduced and allow for a larger reduction in MAC operations. The same algorithms and benchmark are shown in figure 8.B for data movement. From BP to SSDFA, the data movement reduces by several orders of magnitude. Given that each error requires only a single error signal, the communication becomes far more manageable than compared with DFA and BP. We show the same results except for the CIFAR10 benchmark in figure 8.C and 8.D. Because CIFAR10 is a smaller benchmark with less errors that need to be propagated, the advantage of using SSDFA is less. However, as we scale the networks and problems to real world applications the need for reduced data movement becomes more important.

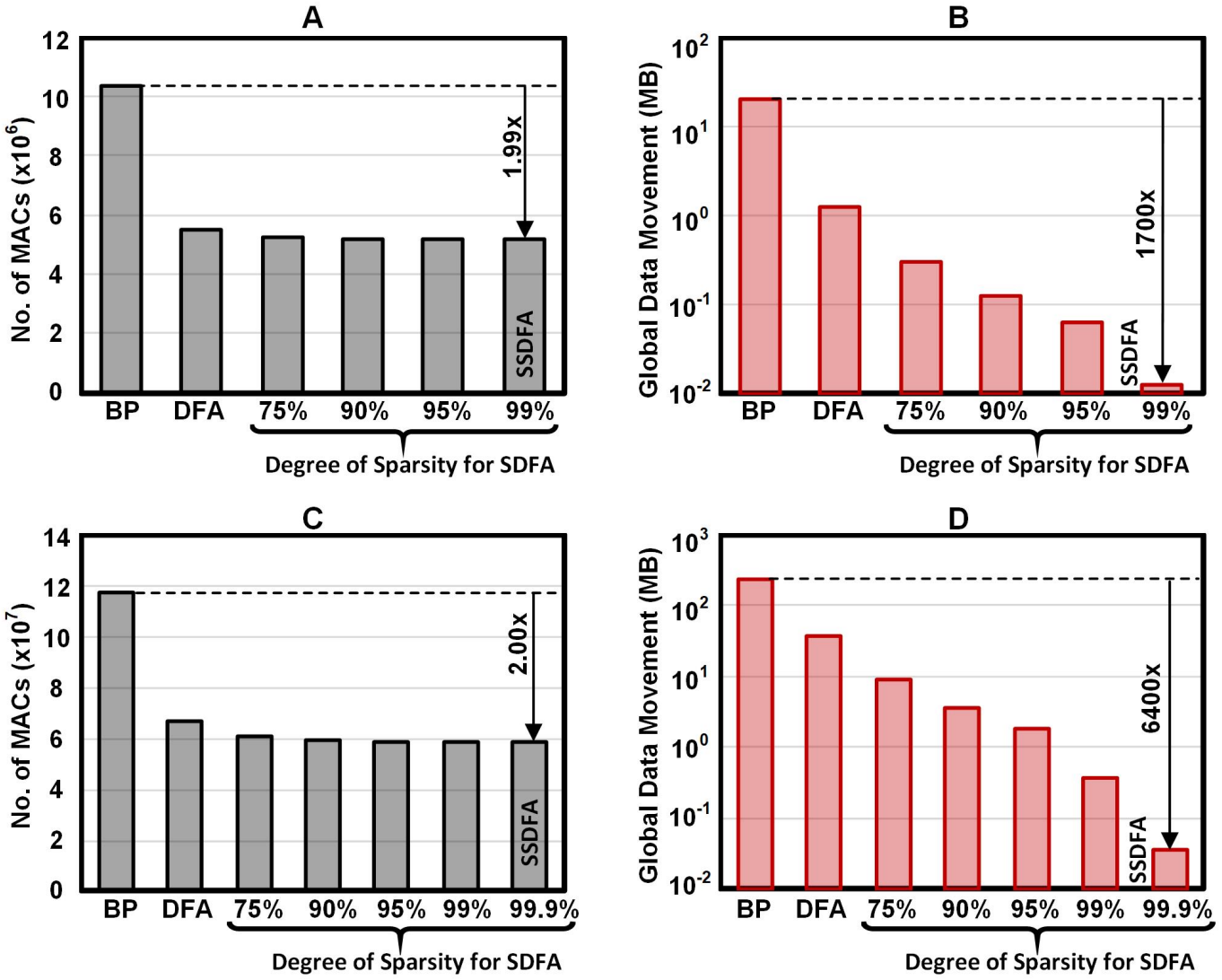

Fig. 8: Total number of MACs and Data Movement (in MB) for a single training example on CIFAR100 (**A, B**) and ImageNet (**C, D**) across BP, DFA, and SDFA

V. Discussion

Feedback Alignment and Direct Feedback Alignment have been proposed to address the weight transport problem in BP. We propose Sparse DFA where the fixed feedback matrix used is constrained to be sparse. Such a sparse feedback helps in a simpler physical implementation of communicating errors, without sacrificing performance. To justify our arguments, we studied the training performance of networks MNIST, CIFAR10, CIFAR100, and ImageNet using feedback matrices with different constraints, such as rank and sparsity. We observe that rank of the feedback matrix has much stronger impact on accuracy, and making the feedback connections sparse has negligible effect on performance. Further, an extremely sparse version of SDFA, where only a single error is fed back for weight update, we observe comparable performance while enabling local learning.

As was claimed in [5], Feedback Alignment, and as a result, the proposed Sparse Feedback Alignment, do not scale to large networks. While our results are similar, we show that this is specifically true for convolutional networks, due to the additional architectural constraints of repeated filters which they incorporate. We show that by fixing the convolutional filters of the network to those trained using BP, and learning only the non-convolutional part using DFA, we observe that the network's performance is close to that of BP. Our approach greatly simplifies the task of propagating the errors from the deeper end of the network to the shallow layers, for learning the model weights.

VI. Methods

We construct our networks in Tensorflow and create a new layer type so that we can feedback error directly to each hidden layer. To get optimal results we perform a hyper parameter search and also test different gradient descent optimizers and activation functions. We sweep learning rate and learning rate decay to find the optimal set. For weight initialization we used a uniform distribution in the range $[-1/\sqrt{fanout}, 1/\sqrt{fanout}]$. For feedback matrix initialization we used $[-1/\sqrt{fanin}, 1/\sqrt{fanin}]$ where the input dimension was layer size. For sparse matrices we found $[-1/\sqrt{\frac{fanin \cdot N}{fanout}}, 1/\sqrt{\frac{fanin \cdot N}{fanout}}]$ (where N is the number of sparse connections) to work well.

References

[1] A. Amaravati, S. B. Nasir, J. Ting, I. Yoon, and A. Raychowdhury. A 55-nm, 1.0-0.4 v, 1.25-pj/mac time-domain mixed-signal neuromorphic accelerator with stochastic synapses for reinforcement learning in autonomous mobile robots. *IEEE Journal of Solid-State Circuits*, 2018.

[2] A. Amravati, S. B. Nasir, S. Thangadurai, I. Yoon, and A. Raychowdhury. A 55nm time-domain mixed-signal neuromorphic accelerator with stochastic synapses and embedded reinforcement learning for autonomous micro-robots. In *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*, pages 124–126. IEEE, 2018.

[3] P. Baldi and P. Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks*, 83:51–74, 2016.

[4] P. Baldi, P. Sadowski, and Z. Lu. Learning in the machine: Random backpropagation and the deep learning channel. *Artificial intelligence*, 260:1–35, 2018.

[5] S. Bartunov, A. Santoro, B. A. Richards, G. E. Hinton, and T. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *arXiv preprint arXiv:1807.04587*, 2018.

[6] Y.-H. Chen, J. Emer, and V. Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016.

[7] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.

[8] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 27–39. IEEE Press, 2016.

[9] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[11] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[12] P. U. Diehl and M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[13] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[14] J. Hsu. Ibm's new brain [news]. *IEEE spectrum*, 51(10):17–19, 2014.

[15] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] H. Kwon, M. Pellauer, and T. Krishna. Maestro: An open-source infrastructure for modeling dataflows within deep learning accelerators. *arXiv preprint arXiv:1805.02566*, 2018.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[19] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[20] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[21] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nature Communications*, 9(1):2385, 2018.

[22] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7:13276, 2016.

[23] 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, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[24] H. Mostafa, V. Ramesh, and G. Cauwenberghs. Deep supervised learning using local errors. *Frontiers in neuroscience*, 12:608, 2018.

[25] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

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

[27] I. Present. Cramming more components onto integrated circuits. *Readings in computer architecture*, 56, 2000.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.