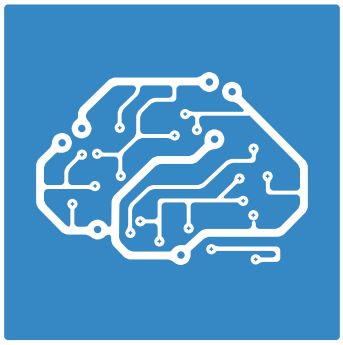

Center for Brains, Minds & Machines

CBMM Memo No. 36 **September 15, 2015**

How Important is Weight Symmetry in Backpropagation?

by

Qianli Liao, Joel Z. Leibo, Tomaso Poggio
Center for Brains, Minds and Machines, McGovern Institute, MIT

Abstract: Gradient backpropagation (BP) requires symmetric feedforward and feedback connections—the same weights must be used for forward and backward passes. This "weight transport problem" [1] is thought to be one of the main reasons of BP's biological implausibility. Using 15 different classification datasets, we systematically study to what extent BP really depends on weight symmetry. In a study that turned out to be surprisingly similar in spirit to Lillicrap et al.'s demonstration [2] but orthogonal in its results, our experiments indicate that: (1) the magnitudes of feedback weights do not matter to performance (2) the signs of feedback weights do matter—the more concordant signs between feedforward and their corresponding feedback connections, the better (3) with feedback weights having random magnitudes and 100% concordant signs, we were able to achieve the same or even better performance than SGD. (4) some normalizations/stabilizations are indispensable for such asymmetric BP to work, namely Batch Normalization (BN) [3] and/or a "Batch Manhattan" (BM) update rule.

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF - 1231216.

1 Introduction

Deep Neural Networks (DNNs) have achieved remarkable performance in many domains [4, 5, 6, 7, 8, 9]. The simple gradient backpropagation (BP) algorithm has been the essential "learning engine" powering most of this work.

Deep neural networks are universal function approximators [10]. Thus it is not surprising that solutions to real-world problems exist within their configuration space. Rather, the real surprise is that such configurations can actually be discovered by gradient backpropagation.

The human brain may also be some form of DNN. Since BP is the most effective known method of adapting DNN parameters to large datasets, it becomes a priority to answer: could the brain somehow be implementing BP? Or some approximation to it?

For most of the past three decades since the invention of BP, it was generally believed that it could not be implemented by the brain [11, 12, 13, 14, 15]. BP seems to have three biologically implausible requirements: (1) feedback weights must be the same as feedforward weights (2) forward and backward passes require different computations, and (3) error gradients must somehow be stored separately from activations.

One biologically plausible way to satisfy requirements (2) and (3) is to posit a distinct "error network" with the same topology as the main (forward) network but used only for backpropagation of error signals. The main problem with such a model is that it makes requirement (1) implausible. There is no known biological way for the error network to know precisely the weights of the original network. This is known as the "weight transport problem" [1]. In this work we call it the "weight symmetry problem". It is arguably the crux of BP's biological implausibility.

In this report, we systematically relax BP's weight symmetry requirement by manipulating the feedback weights. We find that some natural and biologically plausible schemes along these lines lead to exploding or vanishing gradients and render learning impossible. However, useful learning is restored if a simple and indeed *more* biologically plausible rule called Batch Manhattan (BM) is used to compute the weight updates. Another technique, called Batch Normalization (BN) [3], is also shown effective. When combined together, these two techniques seem complementary and significantly improve the performance of our asymmetric version of backpropagation.

The results are somewhat surprising: if the aforementioned BM and/or BN operations are applied, the magnitudes of feedback weights turn out not to be important. A much-relaxed *sign-concordance* property is all that is needed to attain comparable performance to mini-batch SGD on a large number of tasks.

Furthermore, we tried going beyond sign concordant feedback. We systematically reduced the probability of feedforward and feedback weights having the same sign (the *sign concordance probability*). We found that the effectiveness of backpropagation is strongly dependent on high sign concordance probability. That said, completely random and fixed feedback still outperforms chance e.g., as in the recent work of Lillicrap et al. [2].

Our results demonstrate that the perfect forward-backward weight symmetry requirement of backpropagation can be significantly relaxed and strong performance can still be achieved. To summarize, we have the following conclusions:
(I) The magnitudes of feedback weights do not matter to performance. This surprising result suggests that our theoretical understanding of why backpropagation works is probably far from complete.
(II) Magnitudes of the weight updates also do not matter.
(III) Normalization / stabilization methods such as Batch Normalization and Batch Manhattan are necessary for these asymmetric backpropagation algorithms to work. Note that this result was missed by previous work on random feedback weights [2].
(IV) Asymmetric backpropagation algorithms evade the weight transport problem. Thus it is plausible that the brain could implement them.
(V) These results indicate that sign-concordance is very important for achieving strong performance. However, even fixed random feedback weights with Batch Normalization significantly outperforms chance. This is intriguing and motivates further research.
(VI) Additionally, we find Batch Manhattan to be a very simple but useful technique in general. When used with Batch Normalization, it often improves the performance. This is especially true for smaller training sets.

2 Asymmetric Backpropagations

A schematic representation of backpropagation is shown in Fig. 1. Let E be the objective function. Let W and V denote the feedforward and feedback weight matrices respectively. Let X denote the inputs and Y the outputs. W_{ij} and V_{ij} are the feedforward and feedback connections between the j-th output Y_j and the i-th input X_i, respectively. $f(.)$ and $f'(.)$ are the transfer function and its derivative. Let the derivative of the i-th input with respect to the objective function be $\frac{\partial E}{\partial X_i}$, the formulations of forward and back propagation are as follows:

$$Y_j = f(N_j), where\ N_j = \sum_i W_{ij} X_i \tag{1}$$

$$\frac{\partial E}{\partial X_i} = \sum_j V_{ij} f'(N_j) \frac{\partial E}{\partial Y_j} \tag{2}$$

The standard BP algorithm requires $V = W$. We call that case *symmetric backpropagation*. In this work we systematically explore the case of *asymmetric backpropagation* where $V \neq W$.

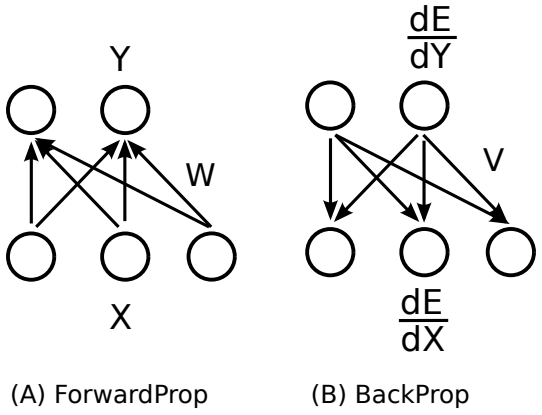

Figure 1: A simple illustration of backpropagation

By varying V, one can test various asymmetric BPs. Let $sign()$ denote the function that takes the sign (-1 or 1) of each element. Let $\circ$ indicate element-wise multiplication. M, S are matrices of the same size as W. M is a matrix of uniform random numbers $\in [0, 1]$ and S_p is a matrix where each element is either 1 with probability $1 - p$ or -1 with probability p. We explored the following choices of feedback weights V in this paper:

1. **Uniform Sign-concordant Feedbacks (uSF)**:
$V = sign(W)$
2. **Batchwise Random Magnitude Sign-concordant Feedbacks (brSF)**:
$V = M \circ sign(W)$, where M is redrawn after each update of W (i.e., each mini-batch).
3. **Fixed Random Magnitude Sign-concordant Feedbacks (frSF)**:
$V = M \circ sign(W)$, where M is initialized once and fixed throughout each experiment.
4. **Batchwise Random Magnitude p-percent-sign-concordant Feedbacks (brSF-p)**:
$V = M \circ sign(W) \circ S_p$, where M and S_p is redrawn after each update of W (i.e., each mini-batch).
5. **Fixed Random Magnitude p-percent-sign-concordant Feedbacks (frSF-p)**:
$V = M \circ sign(W) \circ S_p$, where M and S_p is initialized once and fixed throughout each experiment.
6. **Fixed Random Feedbacks (RndF)**:
Each feedback weight is drawn from a zero-mean gaussian distribution and fixed throughout each experiment: $V \sim \mathcal{N}(0, \sigma^2)$, where σ was chosen to be 0.05 in all experiments.

The results are summarized in the Section 5. The performances of 1, 2 and 3, which we call **strict sign-concordance** cases, are shown in Experiment A. The performances of 4 and 5 with different choices of p, which we call **partial sign-concordance** cases, are shown in Experiment B. The performances and control experiments about setting 6, which we call **no concordance** cases, are shown in Experiments C1 and C2.

3 Normalizations/stabilizations are necessary for "asymmetric" backpropagations

Batch Normalization (BN)
Batch Normalization (BN) is a recent technique proposed by [3] to reduce "internal covariate shift" [3]. The technique consists of element-wise normalization to zero mean and unit standard deviation. Means and standard deviations are separately computed for each batch. Note that in [3], the authors proposed the use of additional learnable parameters after the whitening. We found the effect of this operation to be negligible in most cases. Except for the "BN" baseline condition (e.g., in Table 2), we did not use the learnable parameters of BN.

Batch Manhattan (BM)
We were first motivated by looking at how BP could tolerate noisy operations that could be seen as more easily implementable by the brain. We tried relaxing the weight updates by discarding the magnitudes of the gradients. Let the weight at time t be $w(t)$, the update rule is:

$$w(t+1) = w(t) + \eta * \tau(t) \tag{3}$$

where η is the learning rate.

We tested several settings of $\tau(t)$ as follows:
Setting 0 (SGD): $\tau(t) = -\sum_b \frac{\partial E}{\partial w} + m * \tau(t-1) - d * w(t)$
Setting 1: $\tau(t) = -sign(\sum_b \frac{\partial E}{\partial w}) + m * \tau(t-1) - d * w(t)$
Setting 2: $\tau(t) = sign(-sign(\sum_b \frac{\partial E}{\partial w}) + m * \tau(t-1) - d * w(t))$
Setting 3: $\tau(t) = sign(\kappa(t))$
where $\kappa(t) = -sign(\sum_b \frac{\partial E}{\partial w}) + m * \kappa(t-1) - d * w(t)$

where m and d are momentum and weight decay rates respectively. $sign()$ means taking the sign (-1 or 1), E is the objective function, and b denotes the indices of samples in the mini-batch. Setting 0 is the SGD algorithm (by "SGD" in this paper, we always refer to the mini-batch version with momentum and weight decay). Setting 1 is same as 0 but rounding the accumulated gradients in a batch to its sign. Setting 2 takes an extra final sign after adding the gradient term with momentum and weight decay terms. Setting 3 is something in between 1 and 2, where an final sign is taken, but not accumulated in the momentum term.

We found these techniques to be surprisingly powerful in the sense that they did not lower performance in most cases (as long as learning rates were reasonable). In fact, sometimes they improved performance. This was especially true for smaller training sets. Recall that asymmetric BPs tend to have exploding/vanishing gradients, these techniques are immune to such problems since the magnitudes of gradients are discarded.

We also found that the performance of this technique was proportional to batch size. In the cases of very small batch sizes, discarding the magnitudes of the weight updates was detrimental to performance (see Appendix).

This class of update rule is very similar to a technique called the Manhattan update rule [16]. We suggest calling it "Batch Manhattan" (BM) to distinguish it from the stochastic version [17]. By default, we used setting 1 for BM throughout the Experiments A, B, C1 and C2. The "miscellaneous experiment" at the end of the Section 5.3 demonstrates that settings 1, 2 and 3 give similar performances, so the conclusions we draw broadly apply to all of them.

4 Related Work

Since the invention of backpropagation (BP) [18], its biological plausibility has been a long-standing controversy. Several authors have argued that BP is not biologically plausible [11, 12, 13, 14, 15]. Various biologically plausible modifications have been proposed. Most involve bidirectional connections e.g. Restricted Boltzmann Machines [19, 20] and so-called recirculation algorithms [21, 13] which despite their name provided, in the case of an autoencoder, an elegant early demonstration that adaptive backwards weight can work without being identical to the forward ones. Recently, there have also been BP-free auto-encoders [22] based on "target propagation" [23].

The most relevant work to ours is a recent paper by Lillicrap et al. [2] of which we became aware after most of this work was done. Lillicrap et al. showed that fixed random feedback weights can support the learning of good representations for several simple tasks: (i) approximating a linear function, (ii) digit recognition on MNIST and (iii) approximating the outputs of a random 3 or 4 layer nonlinear neural network. Our work is very similar in spirit but rather different and perhaps complementary in its results, since we conclude that signs must be concordant between feedforward and corresponding feedback connections for consistent good performance, whereas the magnitudes do not matter, unlike Lilicrap et al. experiments in which both signs and magnitudes were random (but fixed). To explain the difference in our conclusions it is useful to consider the following points:
1. We systematically explored performance of the algorithms using 15 different datasets because simple tasks like MNIST by themselves do not always reveal differences between algorithms.
2. We tested deeper networks, since the accuracy of asymmetric BP's credit assignment may critically attenuate with depth (for task (i) and (ii) Lillicrap et al. used a 3-layer (1 hidden layer) fully-connected network, and for task (iii) they used a 3 or 4 layer fully-connected network, whereas in most of our experiments, we use deeper and larger CNNs as shown in Table 1).
3. We found that local normalizations/stabilizations is critical for making asymmetric BP algorithms work. As shown by our results in Table 5, the random feedbacks scheme (i.e. the "RndF" column) suggested by Lillicrap et al. seem to work well only on one or two tasks, permorning close to chance on most of them. Only when combined with Batch Normalization ("RndF+BN" or "RndF+BN+BM" in Table 5), it appears to become competitive.
4. We investigated several variants of asymmetric BPs such as sign-concordance (Table 2, 3 and 4), batchwise-random vs. fixed-random feedbacks (Table 3 and 4) and learning with clamped layers (Table 5 Exp. C2).

5 Experiments

We were interested in relative differences between algorithms, not absolute performance. Thus we used common values for most parameters across all datasets to facilitate comparison.

Method: Key to our approach was the development of software allowing us to easily evaluate the "cartesian product" of models (experimental conditions) and datasets (tasks). Each experiment was a $\{$model,dataset$\}$ pair, which was run 5 times using different learning rates (reporting the best performance).

5.1 Datasets

We extensively test our algorithms on 15 datasets of 5 Categories, including popular datasets of machine learning, object recognition, fine-grained recognition, face recognition, scene recognition and speech recognition. No data augmentation (e.g., cropping, flip, etc.) is used in any of the experiments.

The datasets are briefly described below:

5.1.1 Machine Learning

1. MNIST[24]: handwritten digits recognition; 10 classes of digits 0 to 9; Inputs are 28x28 binary images; 60000 training and 10000 testing samples in total.
2. CIFAR-10[25]: classification of 10 object classes; Tiny 32x32 color natural images; 5000 training and 100 testing samples per class.
3. CIFAR-100[25]: similar to CIFAR-10, but with 100 classes. Training and testing samples per class decrease by 10 fold.
4. SVHN[26]: Street View House Numbers Recognition; Classification of 10 classes, one for each digit; 32x32 color images; Closely cropped version is used; 73257 digits for training, 26032 digits for testing; Extra data NOT used.
5. STL10[27]: classification of 10 object classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck; 96x96 color natural images; 500 training and 800 testing samples per class; There are, in addition, 100,000 unlabeled images, but NOT used in our experiments.

5.1.2 Fine-grained Recognition

1. Flowers17[28]: Classification of 17 categories of flowers. 60 training and 20 testing samples per class.
2. Flowers102[29]: Classification of 102 categories of flowers. Each class consists of between 40 and 258 images.

5.1.3 Basic-level Recognition

1. Caltech101[30]: classification of 102 classes (101 objects + background class). We use 30 training and 10 testing samples per class.
2. Caltech256-101[31]: to achieve a resonable performance on this dataset (and also interesting to compare with Caltech101), we train/test on a subset of randomly sampled 102 classes. 30 training and 10 testing samples are used per class.
3. iCubWorld dataset[32]: We follow the "categorization" protocol; 10 class categorization task; 600 training and 200 testing samples per category.

5.1.4 Face Identification

We perform face identification tasks on three face datasets — Pubfig83, SUFR-W and Labeled Faces in the Wild. The testing protocol is somewhat unconventional, since the above datasets are usually used for face verification (predicting if two images depict the same person). In our experiments, we randomly sample 80 individuals from each dataset and perform a 80-way classification task.

1. Pubfig83-ID[33]: 80-individual identification task. 85 training and 15 testing samples per individual
2. SUFR-W-ID[34]: 80-individual identification task. 10 training and 5 testing samples per individual
3. LFW-ID[35]: 80-individual identification task. 10 training and 5 testing samples per individual

5.1.5 Scene Recognition

We tested our algorithms on one scene dataset.

1. MIT-indoor67[36]: classification of 67 indoor categories; 80 training and 20 testing samples per category

5.1.6 Non-visual tasks

For completeness, we also tested running a fully-connected neural network on a non-visual task: phoneme recognition on TIMIT dataset.

1. TIMIT-80[37]: the development set of TIMIT is used. For each frame, 123 dimensional MFCC features are extracted. We stack each frame with 7 frames before and after it, forming a 15x123 matrix, which is flattened to 1845x1 and fed into a 3-layer feed-forward neural network. We train/test by classifying each frame into 80 subphonetic structures (instead of the standard set of 183). There are 400 training and 100 testing samples per class.

	All others	MNIST	CIFAR10&100	SVHN	TIMIT-80
InputSize	119x119x3	28x28x1	32x32x3	32x32x3	1845x1x1
1	Conv 9x9x48/2	Conv 5x5x20/1	Conv 5x5x32/1	Conv 5x5x20/1	FC 512
2	Max-Pool 2x2/2	Max-Pool 2x2/2	Max-Pool 3x3/2	Max-Pool 2x2/2	FC 256
3	Conv 5x5x128/1	Conv 5x5x50/1	Conv 5x5x64/1	Conv 7x7x512/1	FC 80
4	Max-Pool 2x2/2	Max-Pool 2x2/2	Avg-Pool 3x3/2	Max-Pool 2x2/2	
5	FC max(256,#Class*3)	FC 500	Conv 5x5x64/1	FC 40	
6	FC #Class*2	FC 10	Avg-Pool 3x3/2	FC 10	
7	FC #Class		FC 128		
8			FC 10/100		

Table 1: Network architectures used in the experiments: AxBxC/D means C feature maps of size AxB, with stride D. The CIFAR10&100 architecture has a 2 units zero-padding for every convolution layer and 1 unit right-bottom zero-padding for every pooling layer. The other models do not have paddings. "FC X" denotes Fully Connected layer of X feature maps. In the first model, the number of hidden units in FC layers are chosen according to the number of classes (denoted by "#Class") in the classification task. " max(256,#Class*3)" denotes 256 or #Class*3, whichever is larger. Rectified linear units (ReLU) are used as nonlinearities for all models.

5.2 Training Details

The network architectures for various experiments are listed in Table 1. The input sizes of networks are shown in the second row of the table. All images are resized to fit the network if necessary.

Momentum was used with hyperparameter 0.9 (a conventional setting). All experiments were run for 65 epochs. The base learning rate: 1 to 50 epochs $5 * 10^{-4}$, 51 to 60 epochs $5 * 10^{-5}$, and 61 to 65 epochs $5 * 10^{-6}$. All models were run 5 times on each dataset with base learning rate multiplied by 100, 10, 1, 0.1, 0.01 respectively. This is because different learning algorithms favor different magnitudes of learning rates. The best validation error among all epochs of 5 runs was recorded as each model's final performance. The batch sizes were all set to 100 unless stated otherwise. All experiments used a softmax for classification and the cross-entropy loss function.

Results from all dataset / model pairs are presented. There are a few outliers with worse-than-usual performance since these models were not specifically tuned for each dataset. For example, it seems that Batch Normalization does not work well with the iCub, Flowers and TIMIT dataset, etc. Such outliers do not obscure the general pattern of these results.

5.3 Results

Experiment A: sign-concordant Feedback

In this experiment, we show the performances of setting 1, 2 and 3 in Section 2, which we call **strict sign-concordance** cases: while keeping the signs of feedbacks the same as feedforward ones, the magnitudes of feedbacks are either randomized or set to uniform. The results are shown in Table 2 : Plus sign (+) denotes combination of methods. For example, uSF+BM means Batch Manhattan with uniform sign-concordant feedback. **SGD:** Stochastic gradient descent, the baseline algorithm. **BM:** SGD + Batch Manhattan. **BN:** SGD + Batch Normalization. **BN+BM:** SGD + Batch Normalization + Batch Manhattan. **uSF:** Uniform sign-concordant feedback. This condition often had exploding gradients. **NuSF:** same as uSF, but with feedback weights normalized by dividing the number of inputs of the feedforward filter (filter width * filter height * input feature number). This scheme avoids the exploding gradients but still suffers from vanishing gradients. **uSF+BM:** this setting is somewhat unstable for small batch sizes. Two training procedures were explored: (1) batchsize 100 for all epochs (2) batchsize 100 for 3 epochs and then batchsize 500 for the remaining epochs. The best performance was reported. While this gives a little advantage to this model since more settings were tried, we believe it is informative to isolate the stability issue and show what can be achieved if the model converges well. Note that uSF+BM is the only entry with slightly different training procedures. All other models share exactly the same training procedure & parameters. **uSF+BN, uSF+BN+BM, brSF+BN+BM, frSF+BN+BM:** These are some combinations of uSF, brSF, frSF, BN and BM. The last three are the most robust, well-performing ,and biologically-plausible algorithms.

Experiment A	SGD	BM	BN	BN+BM	uSF	NuSF	uSF +BM	uSF +BN	uSF +BN +BM	brSF +BN +BM	frSF +BN +BM
MNIST	0.67	0.99	0.46	0.80	**88.65**	0.60	0.95	0.65	0.87	0.83	0.97
CIFAR	22.73	23.98	**17.36**	**17.87**	**90.00**	**40.60**	**26.25**	19.88	**18.99**	**19.34**	**19.41**
CIFAR100	55.15	**58.44**	**50.38**	**49.91**	**99.00**	**71.51**	**65.28**	56.99	53.88	**51.35**	52.48
SVHN	9.06	10.77	7.91	10.11	**80.41**	**14.55**	9.78	9.14	10.47	10.51	10.63
STL10	48.01	**44.14**	45.64	**40.34**	**90.00**	**56.53**	46.41	49.53	**41.40**	**41.35**	**41.89**
Cal101	74.08	**66.70**	71.23	**63.33**	**98.95**	**70.50**	75.24	**68.28**	**65.12**	**63.54**	**64.59**
Cal256-101	87.06	**83.43**	85.98	84.31	**99.02**	85.98	86.37	85.59	**83.43**	84.51	**83.82**
iCub	57.62	55.57	**86.91**	**87.31**	**89.96**	**66.57**	**70.61**	**86.26**	**86.46**	**86.51**	**86.91**
Flowers17	35.29	**31.76**	**54.71**	**56.47**	**94.12**	**42.65**	38.24	**57.94**	**51.47**	**55.88**	**52.35**
Flowers102	77.30	77.57	**92.97**	**90.18**	**99.67**	77.92	79.25	**91.17**	**90.70**	**90.76**	**91.20**
PubFig83	63.25	**54.42**	63.83	**54.67**	**98.75**	**78.58**	65.83	**67.92**	**53.67**	**55.25**	**55.92**
SUFR-W-ID	80.00	**74.25**	77.75	**65.50**	**98.75**	**83.50**	79.50	**74.25**	**65.00**	**65.25**	**67.50**
LFW-ID	79.25	**74.25**	**74.75**	**56.75**	**98.75**	**85.75**	80.75	**75.50**	**57.75**	**60.50**	**60.50**
Scene67	87.16	85.37	85.60	**80.75**	**98.51**	88.21	87.09	86.04	**80.90**	**80.22**	**81.72**
TIMIT80-FC	23.04	25.92	**94.28**	**94.00**	23.60	**29.28**	25.84	**92.88**	**93.72**	**93.92**	**93.68**

Table 2: **Experiment A:** The magnitudes of feedbacks do not matter. Sign concordant feedbacks can produce strong performance. iCub, Flowers and TIMIT datasets have low performance whenever Batch Normalization is used — the reason is unclear. Numbers are error rates (%). **Yellow**: performances worse than baseline(SGD) by 3% or more. **Blue**: performances better than baseline(SGD) by 3% or more.

Experiment B: Violating Sign-Concordance with probability p
In this experiment, we test the effect of **partial sign-concordance**. That is, we test settings 4 and 5 as described in Section 2. In these cases, the feedback weight magnitudes were all random. Strict sign-concordance was relaxed by manipulating the probability p of concordance between feedforward and feedback weight signs. Feedback weights $V = M \circ sign(W) \circ S_p$ depend on the matrix S_p as defined in Section 2. Table 3 reports results from setting 4, the case where a new M and S_p is sampled for each batch. Table 4 reports results of setting 5, the case where M and S_p are fixed.

Experiment C1: Fixed Random Feedback
Results are shown in Table 5: **RndF:** fixed random feedbacks. **RndF+BN, RndF+BN+BM:** some combinations of RndF, BN and BM. **uSF+BN+BM:** one of our best algorithms, for reference. The "RndF" setting is the same as the one proposed by [2]. Apparently it does not perform well on most datasets. However, combining it with Batch Normalization makes it significantly better. Although it remains somewhat worse than its sign concordant counterpart. Another observation is that random feedback does not work well with BM alone (but better with BN+BM).

Experiment C2: Control experiments for Experiment C1
The fact that random feedback weights can learn meaningful representations is somewhat surprising. We explore this phenomenon by running some control experiments, where we run two control models for each model of interest: **1.** (.)*Bottom*: The model's last layer is initialized randomly and clamped/frozen. All learning happens in the layers before the last layer. **2.** (.)*Top*: The model's layers before the last layer are initialized randomly and clamped/frozen. All learning happens in the last layer.

There are some observations: **(i)** When only the last layer was allowed to adapt, all models behaved similarly. This was expected since the models only differed in the way they backpropagate errors. **(ii)** With the last layer is clamped, random feedback cannot learn meaningful representations. **(iii)** In contrast, the models with sign concordant feedback can learn surprisingly good representations with the last layer locked—almost as if the last layer was not locked. We can draw the following conclusions from such observations: sign concordant feedback ensures that meaningful error signals reach lower layers by itself, while random feedback is not sufficient. If all layers can learn, random feedback seems to work via a "mysterious co-adaptation" between the last layer and the layers before it. This "mysterious co-adaptation" was first observed by [2], where it was called "feedback alignment" and some analyses were given. Note that our Experiment C shows that the effect is more significant if Batch Normalization is applied.

Miscellaneous Experiment: different settings of Batch Manhattan We show that the 3 settings of BM (as described in Section 3) produce similar performances. This is the case for both symmetric and asymmetric BPs. Results are in Table 6.

Experiment B (1)	Baseline	Random M and S every batch				
Datasets	SGD	100%	75%	50%	25%	Same Sign (brSF+BN+BM)
MNIST	0.67	**7.15**	**8.16**	**8.12**	1.11	0.83
CIFAR	22.73	**70.16**	**75.04**	**70.63**	20.41	**19.34**
CIFAR100	55.15	**94.87**	**95.44**	**94.80**	**69.44**	**51.35**
SVHN	9.06	**73.04**	**78.03**	**75.57**	11.39	10.51
STL10	48.01	**68.92**	**77.14**	**72.92**	45.10	**41.35**
Cal101	74.08	**94.42**	**94.20**	**93.68**	75.13	**63.54**
Cal256-101	87.06	**94.71**	**95.00**	**94.71**	86.76	84.51
iCub	57.62	**86.81**	**86.21**	**85.86**	**86.26**	**86.51**
Flowers17	35.29	**80.88**	**85.00**	**80.59**	**65.88**	**55.88**
Flowers102	77.30	**96.99**	**96.98**	**96.81**	**94.13**	**90.76**
PubFig83	63.25	**96.00**	**96.50**	**95.25**	**75.42**	**55.25**
SUFR-W-ID	80.00	**94.50**	**95.50**	**94.50**	**74.00**	**65.25**
LFW-ID	79.25	**94.75**	**96.25**	**94.75**	**71.25**	**60.50**
Scene67	87.16	**93.58**	**94.85**	**94.10**	**82.91**	**80.22**
TIMIT80	23.04	**94.68**	**94.72**	**94.80**	**94.76**	**93.92**

Table 3: **Experiment B:** Feedbacks have random magnitudes, varing probability of having different signs (percentages in second row, column 3-7) from the feedforward ones. The M and S redrawn in each mini-batch. Numbers are error rates (%). **Yellow**: performances worse than baseline(SGD) by 3% or more. **Blue**: performances better than baseline(SGD) by 3% or more.

Experiment B (2)	Baseline	Fixed random M and S				
Datasets	SGD	100%	75%	50%	25%	Same Sign (frSF+BN+BM)
MNIST	0.67	**4.71**	**4.06**	1.62	0.98	0.97
CIFAR	22.73	**71.42**	**69.88**	**31.35**	21.36	**19.41**
CIFAR100	55.15	**94.66**	**94.89**	**72.02**	56.52	52.48
SVHN	9.06	**43.00**	**31.68**	**15.66**	11.36	10.63
STL10	48.01	**73.94**	**66.09**	50.92	**44.24**	**41.89**
Cal101	74.08	**95.79**	**88.62**	74.18	**68.07**	**64.59**
Cal256-101	87.06	**95.98**	**93.43**	87.84	85.20	**83.82**
iCub	57.62	**87.46**	**87.26**	**86.81**	**86.61**	**86.91**
Flowers17	35.29	**83.24**	**77.94**	**67.94**	**55.88**	**52.35**
Flowers102	77.30	**97.69**	**96.75**	**93.09**	**91.87**	**91.20**
PubFig83	63.25	**94.67**	**88.92**	**73.33**	**60.00**	**55.92**
SUFR-W-ID	80.00	**96.00**	**94.50**	78.25	**68.50**	**67.50**
LFW-ID	79.25	**96.50**	**94.50**	79.00	**63.00**	**60.50**
Scene67	87.16	**94.85**	**92.99**	88.43	84.25	**81.72**
TIMIT80	23.04	**94.72**	**94.24**	**94.24**	**93.40**	**93.68**

Table 4: **Experiment B:** Same as Table 3, but The M and S were fixed throughout each experiment.

	Experiment C1								Experiment C2							
	SGD	RndF	NuSF	BN	RndF +BN	RndF +BM	RndF +BN +BM	uSF +BN +BM	SGD *Bottom*	SGD *Top*	RndF *Bottom*	RndF *Top*	RndF +BN +BM *Bottom*	RndF +BN +BM *Top*	uSF +BN +BM *Bottom*	uSF +BN +BM *Top*
MNIST	0.67	1.81	0.60	0.46	0.88	1.89	1.41	0.87	0.65	**3.85**	**85.50**	**3.81**	**73.74**	**3.85**	0.64	**4.39**
CIFAR	22.73	**42.69**	**40.60**	**17.36**	25.07	**62.71**	**26.33**	**18.99**	23.12	**56.80**	**89.71**	**56.77**	**75.38**	**59.85**	**17.33**	**58.60**
CIFAR100	55.15	**90.88**	**71.51**	**50.38**	**62.75**	**97.11**	**65.47**	53.88	**59.49**	**80.71**	**98.79**	**80.65**	**98.31**	**85.37**	**60.22**	**84.56**
SVHN	9.06	**12.35**	**14.55**	7.91	**13.69**	**13.63**	**13.25**	10.47	8.31	**75.22**	**82.86**	**75.12**	**86.44**	**72.30**	11.27	**69.33**
STL10	48.01	**57.80**	**56.53**	45.64	**52.98**	**80.39**	47.71	**41.40**	49.96	**74.69**	**88.36**	**72.44**	**82.06**	**76.74**	**55.70**	**76.53**
Cal101	74.08	**88.51**	**70.50**	71.23	75.76	**98.42**	73.02	**65.12**	71.97	**82.72**	**98.63**	**79.14**	**98.00**	**84.30**	**68.60**	**83.46**
Cal256-101	87.06	**94.12**	85.98	85.98	87.75	**98.14**	86.96	**83.43**	86.08	89.71	**98.43**	88.92	**97.94**	**90.20**	84.71	89.80
iCub	57.62	**67.87**	**66.57**	**86.91**	**88.06**	**84.41**	**87.46**	**86.46**	**46.73**	**83.96**	**87.56**	**83.26**	**87.26**	**87.66**	**85.91**	**88.91**
Flowers17	35.29	**69.41**	**42.65**	**54.71**	**60.00**	**91.18**	**54.71**	**51.47**	38.24	**70.59**	**92.35**	**70.00**	**90.59**	**80.00**	**62.06**	**76.47**
Flowers102	77.30	**92.31**	77.92	**92.97**	**93.61**	**96.13**	**92.63**	**90.70**	78.99	**86.57**	**97.89**	**86.84**	**99.01**	**95.06**	**91.84**	**94.96**
PubFig83	63.25	**95.42**	**78.58**	63.83	**75.17**	**97.67**	**67.00**	**53.67**	**66.75**	**89.58**	**97.67**	**89.58**	**98.25**	**90.50**	**54.67**	**92.08**
SUFRW-ID	80.00	**94.75**	**83.50**	77.75	78.75	**97.25**	**71.75**	**65.00**	80.50	**90.50**	**97.50**	**90.50**	**97.25**	**89.25**	**70.25**	**88.25**
LFW-ID	79.25	**95.75**	**85.75**	**74.75**	81.25	**97.75**	**68.50**	**57.75**	79.75	**92.50**	**98.25**	**93.00**	**96.75**	**90.00**	**66.75**	**91.00**
Scene67	87.16	**95.75**	88.21	85.60	88.06	**97.69**	87.09	**80.90**	88.73	**91.57**	**97.84**	**91.49**	**97.69**	**90.82**	85.37	**91.57**
TIMIT80	23.04	**26.76**	**29.28**	**94.28**	**94.44**	**33.12**	**93.96**	**93.72**	23.52	**46.20**	**95.00**	**46.20**	**98.44**	**95.40**	**93.04**	**94.64**

Table 5: **Experiment C1 (left):** fixed random feedbacks. **Experiment C2 (right):** (.)*Bottom*: The model's last layer is initialized randomly and clamped/frozen. All learning happens in the layers before the last layer. (.)*Top*: The model's layers before the last layer are initialized randomly and clamped/frozen. All learning happens in the last layer. Numbers are error rates (%). **Yellow**: performances worse than baseline(SGD) by 3% or more. **Blue**: performances better than baseline(SGD) by 3% or more.

	SGD	BM1	BM2	BM3	uSF +BN	uSF +BN +BM1	uSF +BN +BM2	uSF +BN +BM3
MNIST	0.67	0.99	1.30	1.09	0.65	0.87	0.77	0.67
CIFAR	22.73	23.98	23.09	20.47	19.88	**18.99**	**19.20**	**18.82**
CIFAR100	55.15	**58.44**	**58.81**	52.82	56.99	53.88	53.35	54.80
SVHN	9.06	10.77	**12.31**	**12.23**	9.14	10.47	11.11	9.68
STL10	48.01	**44.14**	**44.74**	45.23	49.53	**41.40**	45.81	**43.59**
Cal101	74.08	**66.70**	**65.96**	**70.28**	**68.28**	**65.12**	**67.12**	**64.81**

Table 6: Different settings of Batch Manhattan seem to give similar performances. SGD: setting 0, BM1: setting 1, BM2: setting 2, BM3: setting 3. The interaction of BM with sign concordant feedback weights (uSF) and Batch Normalization are shown in "uSF+BN+(.)" entries. Numbers are error rates (%). **Yellow**: performances worse than baseline (SGD) by 3% or more. **Blue**: performances better than baseline(SGD) by 3% or more.

6 Discussion

This work aims to establish that there exist variants of the gradient backpropagation algorithm that could plausibly be implemented in the brain. To that end we considered the question: how important is weight symmetry to backpropagation? Through a series of experiments with increasingly asymmetric backpropagation algorithms, our work complements a recent demonstration[2] that perfect weight symmetry can be significantly relaxed while still retaining strong performance.

These results show that Batch Normalization and/or Batch Manhattan are crucial for asymmetric backpropagation to work. Furthermore, they are complementary operations that are better used together than individually. These results highlight the importance of sign-concordance to asymmetric backpropagation by systematically exploring how performance declines with its relaxation.

Finally, let us return to our original motivation. How does all this relate to the brain? Based on current neuroscientific understanding of cortical feedback, we cannot make any claim about whether such asymmetric BP algorithms are actually implemented by the brain. Nevertheless, this work shows that asymmetric BPs, while having less constraints, are not computationally inferior to standard BP. So if the brain were to implement one of them, it could obtain most of the benefits of the standard algorithm.

This work suggests a hypothesis that could be checked by empirical neuroscience research: if the brain does indeed implement

an asymmetric BP algorithm, then there is likely to be a high degree of sign-concordance in cortical forward-backward connections.

These empirical observations concerning Batch Manhattan updating may shed light on the general issue of how synaptic plasticity may implement learning algorithms. They show that changes of synaptic strength can be rather noisy. That is, the *sign* of a long term accumulation of synaptic potentiation or depression, rather than precise magnitude, is what is important. This scheme seems biologically implementable.

Acknowledgements We thank G. Hinton for useful comments. This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF 1231216.

References

[1] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive science*, vol. 11, no. 1, pp. 23–63, 1987.

[2] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random feedback weights support learning in deep neural networks," *arXiv preprint arXiv:1411.0247*, 2014.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, (Lake Tahoe, CA), 2012.

[5] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4277–4280, 2012.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3111–3119, 2013.

[8] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1701–1708, IEEE, 2014.

[9] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[11] F. Crick, "The recent excitement about neural networks.," *Nature*, vol. 337, no. 6203, pp. 129–132, 1989.

[12] P. Mazzoni, R. A. Andersen, and M. I. Jordan, "A more biologically plausible learning rule for neural networks.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 10, pp. 4433–4437, 1991.

[13] R. C. O'Reilly, "Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm," *Neural computation*, vol. 8, no. 5, pp. 895–938, 1996.

[14] L. V. Chinta and D. B. Tweed, "Adaptive optimal control without weight transport," *Neural computation*, vol. 24, no. 6, pp. 1487–1518, 2012.

[15] Y. Bengio, D.-H. Lee, J. Bornschein, and Z. Lin, "Towards biologically plausible deep learning," *arXiv preprint arXiv:1502.04156*, 2015.

[16] J. Heaton, "Manhattan propagation, encog." `http://www.heatonresearch.com/javadoc/encog-3.0/org/encog/neural/networks/training/propagation/manhattan/ManhattanPropagation.html`.

[17] E. Zamanidoost, F. M. Bayat, D. Strukov, and I. Kataeva, "Manhattan rule training for memristive crossbar circuit pattern classifiers," 2015.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, 1988.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," 1986.

[21] G. E. Hinton and J. L. McClelland, "Learning representations by recirculation," in *Neural information processing systems*, pp. 358–366, New York: American Institute of Physics, 1988.

[22] Y. Bengio, "How auto-encoders could provide credit assignment in deep networks via target propagation," *arXiv preprint arXiv:1407.7906*, 2014.

[23] Y. Le Cun, "Learning process in an asymmetric threshold network," in *Disordered systems and biological organization*, pp. 233–240, Springer, 1986.

[24] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database: http://www.hollywoodreporter.com/news/earthquake-twitter-users-learned-tremors-226481."

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, p. 5, 2011.

[27] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International conference on artificial intelligence and statistics*, pp. 215–223, 2011.

[28] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, IEEE, 2006.

[29] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, IEEE, 2008.

[30] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[31] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[32] S. R. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone, "icub world: Friendly robots help building good vision data-sets," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pp. 700–705, IEEE, 2013.

[33] N. Pinto, Z. Stone, T. Zickler, and D. Cox, "Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 35–42, IEEE, 2011.

[34] J. Z. Leibo, Q. Liao, and T. Poggio, "Subtasks of Unconstrained Face Recognition," in *International Joint Conference on Computer Vision, Imaging and Computer Graphics, VISIGRAPP*, (Lisbon, Portugal), 2014.

[35] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," in *Workshop on faces in real-life images: Detection, alignment and recognition (ECCV)*, (Marseille, Fr), 2008.

[36] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 413–420, IEEE, 2009.

[37] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus. https://catalog.ldc.upenn.edu/ldc93s1."

A The influence of batch size on Batch Manhattan

We show the influence of batch size on Batch Manhattan (setting 1) in Table 7.

	SGD-100	SGD-30	SGD-10	SGD-5	BM-100	BM-30	BM-10	BM-5
MNIST	0.69	0.66	0.52	0.67	1.00	1.74	2.72	2.66
CIFAR	22.74	22.09	22.28	22.69	23.56	24.53	**31.64**	**35.53**
CIFAR100	54.94	55.81	55.16	56.56	57.81	56.17	**78.49**	**83.14**
SVHN	9.02	8.45	8.54	8.47	**12.83**	**17.06**	**30.60**	**51.19**

Table 7: SGD-x denotes Stochastic Gradient Descent with batchsize x. BM-x denotes Batch Manhattan (setting 1) with batchsize x. Numbers are error rates (%). **Yellow**: performances worse than baseline(SGD-100) by 3% or more. **Blue**: performances better than baseline(SGD-100) by 3% or more. The performance of Batch Manhattan is proportional to the batch size. It seems that some "consensus" is drawn among the samples of each batch such that the weight updates become more meaningful when the batch size gets larger. This effect does not present in SGD.