Learning in the Machine: Random Backpropagation and the Deep Learning Channel (Extended Abstract)*

Pierre Baldi[1†], **Peter Sadowski**[1] and **Zhiqin Lu**[2]
[1]Department of Computer Science, University of California, Irvine
[2]Department of Mathematics, University of California, Irvine
pfbaldi@ics.uci.edu, peter.sadowski@hawaii.edu, zlu@uci.edu

Abstract

Random backpropagation (RBP) is a variant of the backpropagation algorithm for training neural networks, where the transpose of the forward matrices are replaced by fixed random matrices in the calculation of the weight updates. It is remarkable both because of its effectiveness, in spite of using random matrices to communicate error information, and because it completely removes the requirement of maintaining symmetric weights in a physical neural system. To better understand RBP, we compare different algorithms in terms of the information available locally to each neuron. In the process, we derive several alternatives to RBP, including skipped RBP (SRBP), adaptive RBP (ARBP), sparse RBP, and study their behavior through simulations. These simulations show that many variants are also robust deep learning algorithms, but that the derivative of the transfer function is important in the learning rule. Finally, we prove several mathematical results including the convergence to fixed points of linear chains of arbitrary length, the convergence to fixed points of linear autoencoders with decorrelated data, the long-term existence of solutions for linear systems with a single hidden layer and convergence in special cases, and the convergence to fixed points of non-linear chains, when the derivative of the activation functions is included.

1 Introduction

Modern artificial neural networks are optimized using gradient-based algorithms. Gradients can be computed relatively efficiently via the backpropagation algorithm, but the gradient at each weight generally depends on both the data and all other weights in the network. This high degree of interdependence costs energy, both in biological neural systems and in the artificial neural networks simulated using digital computers. Furthermore, the calculation of the gradients in the backpropagation algorithm includes the forward weight matrices, a requirement known as the *weight symmetry problem* that has long been an objection to the hypothesis that biological neurons learn via gradient descent (e.g. [Crick, 1989]). New learning algorithms that do not require full gradient calculations could lead to more efficient neuromorphic hardware and could help explain learning in the brain.

Are gradients really needed for learning in deep neural networks (NNs)? Recent work suggests they are not (e.g. [Jaderberg *et al.*, 2017]). In the random backpropagation algorithm (RBP) [Lillicrap *et al.*, 2016], deep layers of a NN learn useful representations even when the forward weight matrices are replaced with fixed, random matrices in the backpropagation equations. This algorithm differs from greedy *unsupervised* layer-wise approaches [Hinton *et al.*, 2006; Bengio *et al.*, 2007] because the deep weights depend on information about the targets, and it differs from greedy *supervised* layer-wise approaches [Gilmer *et al.*, 2017; Mostafa *et al.*, 2017] because the deep weights depend on the NN output layer, and hence all the other weights.

In this work we connect the RBP algorithm to the notion of the *deep learning channel* that communicates error information from the output layer to the deep hidden layers [Baldi and Sadowski, 2016]. This channel is *necessary* to converge to critical points of the objective, and can be studied using tools from information and complexity theory. We classify learning algorithms by the information that is transmitted along this channel, and our analysis leads to several new learning algorithms, which we analyze through experiments on the MNIST [LeCun *et al.*, 1998], CIFAR-10 [Krizhevsky and Hinton, 2009], and HIGGS [Baldi *et al.*, 2014] benchmark data sets. Furthermore, we prove that these algorithms converge to a global optimum of the objective function for important special cases.

2 Random Backpropagation Algorithms

In this work we consider layered, feed-forward, neural networks in which neurons in layer h are fully-connected to the neurons in the previous layer $h-1$. Layer activity O^h is computed as a function of the preceding layer as

$$\begin{aligned} O^h &\triangleq f^h(S^h) \\ S^h &\triangleq W^h O^{h-1} \quad \text{for} \quad 1 < h \le L \end{aligned} \tag{1}$$

*This paper is an extended abstract of the article Baldi, *et al.*, *Artificial intelligence*,260:1–35, 2018.
†Contact Author

where $O^0 = I$ is the input data and f^h is a non-linear activation function. We focus here on supervised learning with typical output activation functions and loss functions, including linear, sigmoid, and softmax output layers, for which the derivative of the loss $\mathcal{E}$ with respect to S^L for a single input-target pair (I, T) is given by

$$\frac{\partial \mathcal{E}}{\partial S^L} = O^L - T. \tag{2}$$

The backpropagation algorithm works by first computing gradients at each *neuron* recursively, then computing the gradients at the weights. These gradients are then used to update the weights:

$$B^h \triangleq -\frac{\partial \mathcal{E}}{\partial S^h} = \begin{cases} (T - O^L), & \text{for } h = L \\ (f^h)' \odot (W^{h+1})^T B^{h+1}, & \text{for } h < L \end{cases}$$
$$\Delta W^h = -\eta \frac{\partial \mathcal{E}}{\partial W^h} = \eta O^{h-1} B^h \tag{BP}$$

where the derivative $(f^h)'$ is evaluated at O^h, and η is the learning rate. In *random* backpropagation, the gradients B^h are replaced with a randomized error signal R^h defined recursively

$$R^h \triangleq \begin{cases} (T - O^L), & \text{for } h = L \\ (f^h)' \odot \boldsymbol{C^h} R^{h+1}, & \text{for } h < L \end{cases}$$
$$\Delta W^h = \eta O^{h-1} R^h \tag{RBP}$$

with constant matrices $\{C^h\}_{1 \le h \le L}$ replacing the transpose of the weight matrices in each layer. In *skip* random backpropagation [Baldi *et al.*, 2018], the randomized error signals are sent directly to the deep layers rather than propagating through each intermediate layer.

$$R^h \triangleq \begin{cases} (T - O^L), & \text{for } h = L \\ (f^h)' \odot \boldsymbol{C^h R^L}, & \text{for } h < L \end{cases}$$
$$\Delta W^h = \eta O^{h-1} R^h \tag{SRBP}$$

where random matrix C^h now connects layer h to the output layer.

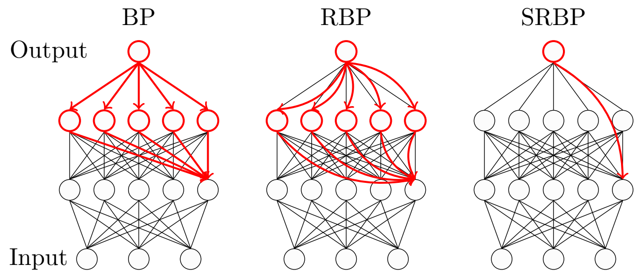

Figure 1: The path of the error signal (red) from an output neuron to a deep, hidden neuron in backpropagation (BP), random backpropagation (RBP), and skip random backpropagation (SRBP).

These learning rules can be compared in terms of the information required to update each weight. RBP solves the weight-symmetry problem by removing the dependency of the update on the forward weights in the backpropagation step; the updates still depend on every other weight in the network, but all that information is subsumed by the error signal at the output, $T - O^L$. In SRBP, we also remove the dependency of ΔW^h on the derivative of the transfer function in the downstream layers, $(f^l)', l > h$. Despite these differences, we show that the these learning algorithms (as well as adaptive variants) *still converge to critical points of the objective function* in network architectures conducive to mathematical analysis, unlike other alternative deep learning algorithms such as greedy, layer-wise "pre-training."

In addition, we introduce the idea of *adaptive* random backpropagation (ARBP), where the backpropagation matrices in the learning channel are initialized randomly, then progressively adapted during learning using the product of the corresponding forward and backward signals, so that

$$\Delta C^h = \eta O^h R^{h+1}.$$

In this case, the forward channel becomes the learning channel for the backward weights. This adaptive behavior can also be used with the skip version (ASRBP).

In all these algorithms, the weight updates in the last layer are equivalent to those of BP, so BP=RBP=SRBP=ARBP=ASRBP in the top layer. They only differ in the way they train the *hidden* layers. In experiments, we also compare to the case where only the top layer is trained, and the hidden layers remain fixed after a random initialization.

3 Results

3.1 Mathematical Results

Through mathematical analysis, we prove that RBP and SRBP converge to a fixed point corresponding to the global optimum of the training set loss for the following neural networks architectures, starting from almost any set of initial weights (except for a set of measure 0). Proofs for the case of ARBP are provided in [Baldi *et al.*, 2017].

- A chain of single linear neurons of arbitrary length ($[1, \ldots, 1]$).
- An expansive architecture of linear neurons $[1, N, 1]$.
- A compressive architecture of linear neurons $[N, 1, N]$.
- A simple $[1, 1, 1]$ architecture, with a power function non-linearity in the hidden neuron of the form $f(x) = x^\mu$. Setting $\mu = 1/3$ for instance gives an S-shaped activation. Furthermore, we show that this system generally does *not* converge for $\mu \neq 1$ when the derivative of the transfer function is omitted from the learning rule.

For the linear architectures, under a set of standard assumptions, we can derive a set of polynomial, autonomous, ordinary differential equations (ODEs) for the average time evolution of the weights under the different learning algorithms. As soon as there is more than one variable and the system is non-linear, there is no general theory to understand the corresponding behavior. In fact, even in two dimensions, the problem of understanding the upper bound on the number and relative position of the limit cycles of a system of the form $dx/dt = P(x, y)$ and $dy/dt = Q(x, y)$, where P and Q are polynomials of degree n is open–in fact this is Hilbert's 16-th problem in the field of dynamical systems [Smale, 1998; Ilyashenko, 2002]. Thus, for the general $N_0, \ldots, N_L$ network architectures used in practiced, we turn to experiments.

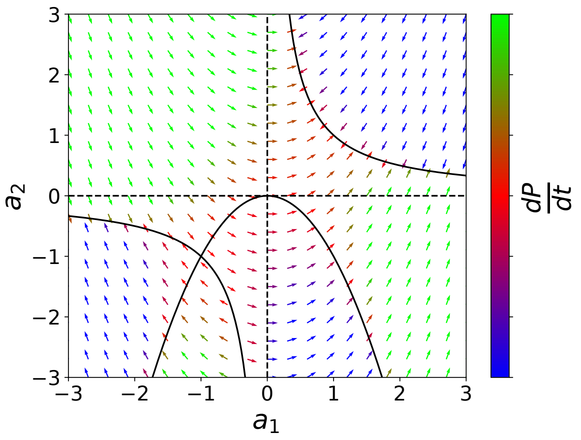

Figure 2: Vector field of RBP dynamics in the $[1, 1, 1]$ linear case with two network weights, a_1 and a_2, plotted on the x and y axis, and a fixed random backprop weight $c_1 = 1$. The critical points of the learning equations correspond to the two hyperbolas, and all critical points are fixed points and global minima of the error functions. Arrows are colored according to the value of $dP/dt = d(a_1a_2)/dt$, showing how the critical points inside the parabola $a_2 = -a_1^2/c_1$ are unstable. All other critical points are attractors. Reversing the sign of c_1 leads to a reflection across both the a_1 and a_2 axes.

3.2 Empirical Results

We performed extensive experiments on variants of random backpropagation algorithms. We tested the following variations through experiments on MNIST, CIFAR-10, and HIGGS benchmark classification data sets, using architectures with 5-10 layers of tanh or relu neurons. The overall conclusion is that RBP and its variants are surprisingly robust to these variations.

- **Activations:** Random backpropagation algorithms work with different transfer functions including linear, logistic, tanh, and relu units.
- **Derivatives:** The derivative of the activation function, f' appears to be an important factor in the learning rule. Our mathematical analysis found a case where RBP only converges if this term is included, and in experiments with real data we observe similar results (Figure 3).
- **Adaptation:** ARBP and ASRBP, the adaptive versions, are both able to learn. along with a variant reminiscent of Spike-Time Dependent Plasticity (STDP).

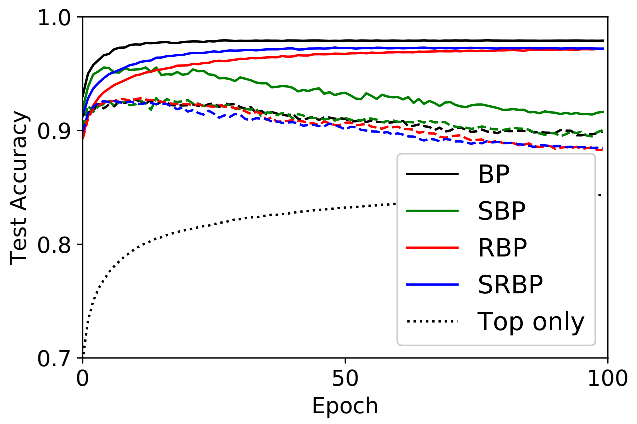

Figure 3: MNIST test accuracy for backpropagation (BP), random backpropagation (RBP), skip random backpropagation (SRBP), and a skipped variant of backpropagation (SBP) in a 5-layer architecture. BP, RBP, and SRBP converge to high accuracy while training the top layer only (Top only) does not. Variants without transfer derivatives in the learning rule (dashed lines) also fail.

- **Convolution:** RBP and SRBP both can be used to train convolutional neural network architectures on CIFAR-10. (Confirmed also by [Nøkland, 2016].)
- **Precision:** We show that reducing the precision of the error signals $\{R^h\}_{1 \le h \le L}$ leads to gracious degradation, as opposed to catastrophic failure, as the error signals are quantized down to a single bit (Figure 4). Similar results are observed when we limit the precision of the weight updates ΔW^h rather than that of R^h.
- **Sparsity:** RBP and SRBP work when the random learning channel matrices $\{C^h\}_{1 \le h \le L}$ are sparse, even if each element is a single bit (Figure 5).
- **Rank:** The performance of both RBP and SRBP degrades in as the rank of the matrices decrease to zero (Figure 6).
- **Dropout:** The dropout algorithm can be used in the learning channel in both RBP and SRBP. This does not appear to have a large impact on learning, and it is unclear whether it has the same regularizing effect.
- **Stochasticity:** If the matrices of the learning channel are randomly sampled from a Normal or Uniform distribution with mean zero at each stochastic mini-batch update, then performance is poor and similar to training only the top layer.
- **Sign-Concordance:** If each element of C^h is constrained to have the same sign as the corresponding forward weight in RBP, then the system learns. This is the sign-concordance algorithm explored by Liao, et al. [Liao *et al.*, 2016].
- **Initialization:** If the elements of the matrices of the learning channel in RBP or SRBP are sampled from a uniform or normal distribution with non-zero mean, performance is unchanged. This is also consistent with the

sparsity experiments above, where the means of the sampling distributions are not zero.

- **Sign:** If we remove the sign information from the weight updates, keeping only the absolute value, the system does not learn.
- **Weight Cooperation:** If a different random backward weight is used to send an error signal to each individual *weight*, rather than to a hidden neuron which then updates all it's incoming weights using the same signal, the system does not learn.
- **Fixing Layers:** If we fix the top layer(s) of a network (those closest to the output) and only train the lower layers, the network does not learn. This is different from gradient descent.

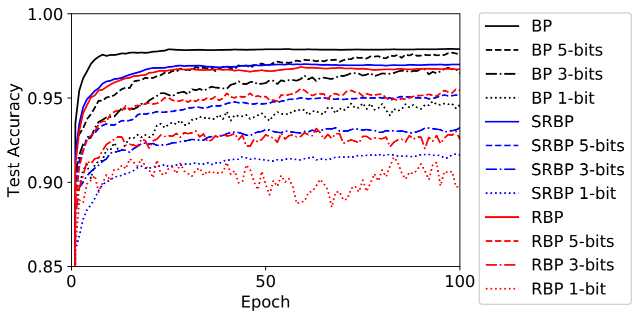

Figure 4: MNIST test accuracy with low-precision error signals (gradients in the case of BP). The error signals B^h and R^h are quantized on a log scale symmetric around 0. Performance is close to full precision at 7-bit quantization, and progressively degrades as the number of bits is decreased.

4 Conclusion

Here we have derived several variants of RBP and studied them through simulations and mathematical analyses. The emerging picture is that many of the variants lead to robust deep learning even without the computation of gradients. These algorithms often lead to slower learning when compared to backpropagation on networks of a fixed size, but they should be useful in the future both to better understand biological neural systems, and to implement new neural physical systems in silicon or other substrates.

Additional variants are studied in two followup papers. [Baldi *et al.*, 2017] considers symmetry issues such as having a learning channel with an architecture that is not a symmetric version of the forward architecture, or having non-linear units in the learning channel that are similar to the non-linear units of the forward architecture. [Baldi and Sadowski, 2018] connects RBP to the recirculation learning algorithm for autoencoder networks [Hinton and McClelland, 1988], showing that they use the same learning principle. Recirculation is noteworthy because it removes a second major objection to the biological-plausibility of backpropagation: the use of fundamentally different types of signals in the forward and backward propagation steps.

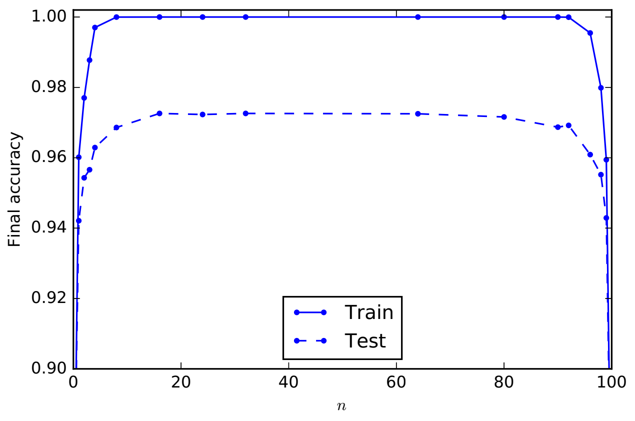

Figure 5: MNIST accuracy after training with sparse SRBP, plotted against the sparsity in the random weight matrices $\{C^h\}_{1\leq h\leq 5}$. The random backpropagation matrix connecting any two layers is created by sampling each entry using a (0,1) Bernoulli distribution, where each element is 1 with probability $p = n/(\text{fan} - \text{in})$ and 0 otherwise. For extreme values of n, sparse SRBP fails: for $n = 0$, all the backward weights are set to zero and no error signals are sent; for $n = 100$ all the backward weights are set to 1, and all the neurons in a given layer receive the same error signal. The performance of the algorithm is surprisingly robust in between these extremes.

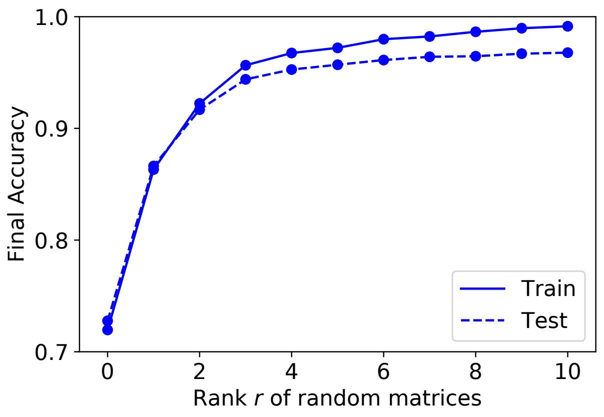

Figure 6: MNIST accuracy after training with SRBP, plotted against the rank of the random weight matrices in network with seven hidden layers of 100 tanh units each. Random weight matrices were initialized using a Glorot-scaled uniform distribution, then a rank-r approximation to the matrix was computed using the truncated singular value decomposition. Both SRBP and RBP (not shown) perform best with full-rank matrices in all layers.

The robustness and other properties of these algorithms cry for explanations and more general principles. We have provided both intuitive and formal explanations for several of these properties. On the mathematical side, polynomial learning rules in linear networks lead to systems of polynomial differential equations. We have shown in several cases that the corresponding ODEs converge to an optimal solution. However these polynomial systems of ODEs rapidly become complex and, while the results provided are useful, they are not yet complete, thus providing directions for future research.

References

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

[Baldi and Sadowski, 2018] Pierre Baldi and Peter Sadowski. Learning in the machine: Recirculation is random backpropagation. *Neural Networks*, 108:479–494, 2018.

[Baldi *et al.*, 2014] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5, 2014.

[Baldi *et al.*, 2017] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the machine: The symmetries of the deep learning channel. *Neural Networks*, 95:110–133, 2017.

[Baldi *et al.*, 2018] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the machine: Random backpropagation and the deep learning channel. *Artificial intelligence*, 260:1–35, 2018.

[Bengio *et al.*, 2007] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, Université De Montréal, and Montréal Québec. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

[Crick, 1989] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

[Gilmer *et al.*, 2017] Justin Gilmer, Colin Raffel, Samuel S. Schoenholz, Maithra Raghu, and Jascha Sohl-Dickstein. Explaining the learning dynamics of direct feedback alignment. 2017.

[Hinton and McClelland, 1988] Geoffrey E Hinton and James L McClelland. Learning representations by recirculation. In *Neural information processing systems*, pages 358–366. New York: American Institute of Physics, 1988.

[Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.

[Ilyashenko, 2002] Yu Ilyashenko. Centennial history of hilbert's 16th problem. *Bulletin of the American Mathematical Society*, 39(3):301–354, 2002.

[Jaderberg *et al.*, 2017] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635. JMLR. org, 2017.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[LeCun *et al.*, 1998] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Liao *et al.*, 2016] Qianli Liao, Joel Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1837–1844, 2016.

[Lillicrap *et al.*, 2016] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, November 2016.

[Mostafa *et al.*, 2017] Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *arXiv preprint arXiv:1711.06756*, 2017.

[Nøkland, 2016] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1037–1045, 2016.

[Smale, 1998] Steve Smale. Mathematical problems for the next century. *The Mathematical Intelligencer*, 20(2):7–15, 1998.