Backpropagation Free Transformers

Dinko Franceschi*
Max Planck Institute for Software Systems
Saarbrücken, Germany
dfrances@mpi-sws.org

Abstract

This works focuses on evaluating whether the neural network transformer architecture can be learned in a more biologically plausible manner than is currently done using backpropagation. We identify three main problems with the biological plausibility of backpropagation-based learning, the *weight transport problem*, the *global loss problem* and the *asymmetry problem*, and prescribe methods aimed at addressing these issues. We show how these methods can be extended to deep and complex networks like transformers and we evaluate their effectiveness as a function of network depth. Our experiments demonstrate learning for large neural networks in natural language processing, which is a novel application of backpropagation free methods.

1 Introduction

Deep Neural Network (DNN) approaches to machine translation and speech recognition have been shown to be very effective. While DNNs have taken inspiration from the brain, they are incompatible with current understandings in neuroscience and biology. Future advances in deep learning could strongly benefit from biologically plausible learning algorithms [3].

We will begin by providing an overview of the current state of BP-free learning methods. In this analysis we cover Feedback Alignment [7], Difference Target Propagation [6], Local Representation Alignment [8], as well as more recent advances such as [13], [2], [5], [9], [1]. After that we will offer a comparative explanatory overview of the most promising biologically realistic algorithms for learning in deep neural networks. We will finally run experiments which go beyond brain-inspired BP-free learning in simple shallow networks, and to the best of our knowledge, this is the first instance of BP-free learning for large neural networks for natural language processing.

2 The Credit Assignment Problem

The credit assignment problem centers on the notion of how much "blame" or "credit" an individual neuron should receive for a specific output of a network. Once this is known, the weights can be adjusted so that the network performs better in the future. A common approach to the credit assignment problem is to change the weights using the gradient of the objective function. In artificial neural networks, the most prominent method is backpropagation which computes the credit for each synapse. The process begins at the outermost layer of a network and recursively calculates gradients using the chain rule [10]. Backpropagation is very powerful but it mandates that neurons dispatch between one another a large number of synaptic weight information. This forward and backward transport of weights is biologically implausible as synapses in the brain operate unidirectionally.

*Work performed while at Columbia University

Preprint. Under review.

3 Implausibility of Backpropagation

The standard deep learning network has information going in the forward direction (forward path) from the input to the output y_l of a given layer l. On its way from input to output, the signal is multiplied by a matrix of synaptic weights W_l. There is also a backwards path which then sends errors signals e_l in the backwards direction according to the error-backpropagation rule. The aforementioned weight matrix is transposed, W_l^T, in the backwards path as the error signals travel in reverse. While in this artificial neural network construct we have the synaptic weight matrix, W_l, appear in both directions, the synapses in the forward and backwards directions are completely separate and do not communicate in the brain. This is known as the *weight transport problem* and it makes the current deep learning framework involving backpropagation biologically implausible, since biological neurons do not provide weight transport or bidirectional synapses [7]. There are however a number of additional arguments against the plausibility of BP. In particular [6] argues that biological neurons are not likely to be transmitting gradients of single global loss, as occurs in backpropagation, but they are more likely to be transmitting targets that earlier layers will attempt to adapt to. We dub this issue the *global loss problem*. Finally, as is argued by [8], it seems likely that biological forward and backward neurons are making the same type of computations which is not the case in backpropagation. Indeed, BP models forward neurons as linear functions followed by component-wise nonlinearities. Meanwhile it models backward neurons as linear functions followed by products with the pointwise derivatives of the activation functions in the forward pass. We dub this issue the *asymmetry problem*. For illustration, consider an MLP with a single hidden layer, shown in Figure 2. Let an example be given as (z_0, y_2), where z_0 is the input and y_2 is the expected label. $h_1 = W_1 z_0 + b_1, \quad z_1 = \phi_1(h_1) = \mathrm{ReLU}(h_1), \quad h_2 = W_2 z_1 + b_2, \quad z_2 = \phi_2(h_2) = \mathrm{softmax}(h_2), \quad L_2 = H(y_2, z_2) = -y_2^\top \log z_2$

Note that z_2 is a probability distribution and the loss L_2 is the cross-entropy loss between z_2 and y_2, where y_2 is a one-hot vector indicating the expected label. The gradients of the parameters with the respect to the global loss L_2 are then computed by the computational graph shown in Figure 3 as

$$e_2 = \frac{\partial L_2}{\partial z_2} = -\frac{y_2}{z_2}, \quad \delta_2 = \frac{\partial L_2}{\partial h_2} = \phi_2'(h_2) \odot e_2, \quad e_1 = \frac{\partial L_2}{\partial z_1} = W_2^\top \delta_2, \quad \delta_1 = \frac{\partial L_2}{\partial h_1} = \phi_1'(h_1) \odot e_1$$

4 Solving the Weight Transport Problem

We examine three algorithms which do not exhibit the weight transport problem: Feedback Alignment (FA), Weight Mirror (WM) and Kolen-Pollack (KP). These algorithms all introduce a new set of parameters to be used while computing training signals, as shown for example in Figure 4. The additional set of weights E_2 is then incentivized to be similar to $W_2^\top$ during training. This property causes learning to mimic backpropagation and thus can achieve similar results to backpropagation even in deep networks.

4.1 Feedback Alignment

Feedback Alignment takes the radical approach of setting the parameters of the backward neurons to be a fixed random matrix. So the matrix $W_2^\top$ is replaced by a fixed random matrix of weights B. It bypasses the transport of weight information in the backward direction [7]. Instead, it involves multiplying the error of a network with random synaptic weights. The forward weights align to the fixed random weights to send appropriate learning signals to inner layers of the network. Surprisingly, this is found to work well for MLPs when the number of layers is limited. In our experiments we found that Feedback Alignment works better than chance up to two transfomer layers, but fails to converge entirely for any deeper networks. Notably, while Feedback Alignment was shown to perform well in shallow networks, it obtains disappointing results when compared to BP in deep neural networks [7].

4.2 Weight Mirror

Weight Mirror is a neural circuit that is able to incrementally tweak the initially random matrix of synaptic weights, B, so that it ultimately becomes equivalent to the transpose of W without the transport of weights information. So it is said that B mirrors W^T. The networks alternates between

two modes: engaged mode and mirror mode. In the engaged mode it takes in inputs and adjusts the forward weights. This is done by the cross-projections sending across feedback signals to forward neurons. In the mirror mode, forward neurons fire noisily, and forward cells drive feedback cells so that $\delta_l = y_l$ and $\delta_{l+1} = y_{l+1}$. This is done by the Hebbian learning rule $\Delta E_{l+1} = \eta \delta_l \delta_{l+1}^\top - \lambda E_{l+1}$, where δ are the signals in the feedback path. In the example shown in Figure 4, the weight mirror algorithm modifies matrix E_2 with the update $\Delta E_2 = \eta z_1 z_2^\top - \lambda E_2$, and the reason this works is essentially that $E[z_1 z_2^\top] = E[z_1 \phi_2(W_2 z_1 + b_2)^\top] \approx \alpha W_2^\top$.

4.3 Kolen-Pollack

In the Kolen-Pollack algorithm instead of having matrices of weights W and B we consider individual synapses w and b. This is because in the brain, changes to different synapses are calculated within the synapses themselves. This algorithm allows for learning without transporting weights or weight changes. The central idea is that the only variables transmitted between neurons are vectors y_l and δ_{l+1}, and each synapse calculates its own adjustments locally [1]. In essence, unlike in the weight mirror, here there is a single mode of operation. Feedback signals δ are conveyed to forward path neurons so that they can adjust forward weights w using learning rule: $\Delta \mathbf{w}_{l+1} = -\eta_w \boldsymbol{\delta}_{l+1} \mathbf{y}_l^T - \lambda \mathbf{w}_{l+1}$. Also forward signals y are conveyed to feedback cells so that they can adjust feedback weights b using learning rule: $\Delta \mathbf{b}_{l+1} = -\eta_w \mathbf{y}_l \boldsymbol{\delta}_{l+1}^T - \lambda \mathbf{b}_{l+1}$ [1]. Synapses in the feedback path adjust themselves based on their own inputs and individual neuron-specific teaching signals from the forward path. At each time step, the individual synapses go through the same adjustments and have the same weight-decay factor λ. In the example shown in Figure 4, this results in the simple update rule: $\Delta E_2 = -\eta z_1 \delta_2^\top - \lambda E_2$.

5 Solving the Global Loss Problem: Target Propagation

The global loss problem is frequently tackled by defining local losses that only adjacent layers are responsible to minimize. An approach which we elaborate on below stresses the importance of target value rather than a loss gradient. It suggests computing a good target which essentially provides a layer-local training criterion which can then be defined to update each layer separately. Figure 5 illustrates how local targets are computed for a simple two-layer MLP. In Target Propagation the target y_1 is computed as: $y_1 = \bar{\phi}_2(E_2 z_2)$, while in Difference Target Propagation it is computed as: $y_1 = z_1 + (\bar{\phi}_2(E_2 y_2) - \bar{\phi}_2(E_2 z_2))$. In both cases the local losses are: $L_1 = \frac{1}{2}||y_1 - z_1||^2, \quad L_2^{\text{inv}} = \frac{1}{2}||\bar{\phi}_2(E_2 z_2 + \epsilon) - (y_1 + \epsilon)||^2, \quad \epsilon \sim \mathcal{N}(0, \sigma)$. The central idea is to compute at each layer targets as opposed to that of gradients in traditional backpropagation. In Difference Target Propagation (DTP), the computation of the local target is corrected in an attempt to stabilize training [3]. The correction consists in using the backward mapping to compute the displacement of the current state as opposed to the target directly.

6 Solving the Asymmetry Problem: LRA

Although the asymmetry problem in backpropagation seems quite evident, we found that only the LRA algorithm provides a solution to it, while also generating local targets and avoiding weight transport. Local Representation Alignment (LRA) is also a biological motivated algorithm [8]. Notably, its target computation and error unit mechanism is agnostic to underlying feedforward model so it allows for extensions to models such as residual networks and other more exotic architectures. The LRA algorithm centers around the general process of coordinated local learning rules. Computing targets with these kinds of rules should not require an actual pathway, as in back-propagation, and instead make use of top-down and bottom-up signals to generate targets. LRA has 2 parts. Part 1 involves computing error units and targets, and part 2 involves computing weight updates. At any given layer z_l, we calculate the target for the layer below z_{l-1} by multiplying the error unit values at l by a set of synaptic error weights E_l. This is then subtracted from the initially found pre-activation of the layer below h_{l-1}. Once the targets for each layer have been found, we can then use the local loss $L_l(y_l, z_l)$ to compute updates to the weights W_l and its corresponding error weights E_l.

7 Application to Transformers

Transformers were first introduced by [11] and were used to obtain dramatic improvements in a number of NLP tasks in [4]. We could find no existing work applying any backpropagation free method to transformers. The architecture of transformers is rather complex, however it does fit the model of a sequence of linear transformations of the input data, each followed by a fixed non-linearity. The version of [4] in particular is a stack L layers, each of the form:

$$z_{4l+l} = \text{self-attention}\left(\begin{bmatrix} W_K \\ W_Q \\ W_V \end{bmatrix} z_{4l} + \begin{bmatrix} b_K \\ b_Q \\ b_V \end{bmatrix}\right), \quad z_{4l+2} = \text{layer-norm}(\text{dropout}(W_n z_{4l+1} + b_n) + z_1),$$

$$z_{4l+3} = \text{GELU}(W_i z_{4l+2} + b_i), \quad z_{4l+4} = \text{layer-norm}(\text{dropout}(W_o z_{4l+3} + b_o) + z_{4l+2}), \quad \ldots$$

where l is the layer index in $0, \ldots, L-1$. Because each layer of a transformer can be expressed in this "MLP" form, we are able to apply the BP-free algorithms described earlier as if a transformer was a simple MLP, with the only caveat that some of the non-linearities are not component-wise.

7.1 Experiments: MNLI

MNLI (Multi-Genre Natural Language Inference) is a large-scale, crowdsourced entailment classification task [12] with 433k sentences pairs. Given a sentence pair, the model is asked to determine whether the first sentence entails the second, whether it contradicts the second, or neither. Since there are only three choices per example, picking a label randomly will give an accuracy of about 33%.

7.2 Results

Our experimental results are shown in Figure 1. We trained a transformer from scratch with no pretraining for 1 epoch on all of the MNLI training set and then evaluate on the MNLI development set. We found that the Kolen-Pollack algorithm indeed tracks the performance of backpropagation closely, regardless of the depth for transformers, showing that backpropagation free learning is indeed quite achievable even in deep complex networks and with realistic NLP tasks.

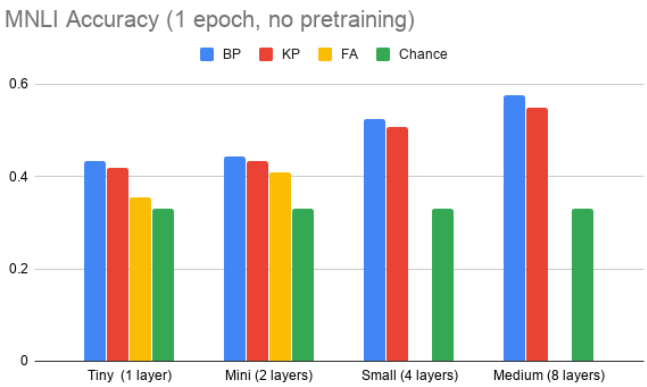

Figure 1: Classification accuracy on the MNLI task for transformers with different number of layers, for different learning algorithms.

8 Conclusion

Deep neural networks have had a tremendous impact on the advancement of language processing and computer vision and they are the basis of the current rise of AI technologies. This work contributes to the long-term quest of developing a biologically plausible ANN learning mechanism which more closely resembles learning in the brain. We provide a comparative study of promising biologically plausible algorithms for learning which nearly match the power of backpropagation, and we demonstrate learning for large neural networks in natural language processing, which is a novel application of backpropagation free methods.

References

[1] Mohamed Akrout, Collin Wilson, Peter C Humphreys, Timothy Lillicrap, and Douglas Tweed. Using weight mirrors to improve feedback alignment. *arXiv preprint arXiv:1904.05391*, 2019.

[2] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378, 2018.

[3] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Daniel Kunin, Jonathan M Bloom, Aleksandrina Goeva, and Cotton Seed. Loss landscapes of regularized linear autoencoders. *arXiv preprint arXiv:1901.08168*, 2019.

[6] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[7] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

[8] Alexander G Ororbia and Ankur Mali. Biologically motivated algorithms for propagating local target representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4651–4658, 2019.

[9] B Richards, T Lillicrap, P Beaudoin, A Saxe, and R Bogacz. A deep learning framework for neuroscience. *Nature Neuroscience*, 2019.

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[12] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[13] Will Xiao, Honglin Chen, Qianli Liao, and Tomaso Poggio. Biologically-plausible learning algorithms can scale to large datasets. *arXiv preprint arXiv:1811.03567*, 2018.

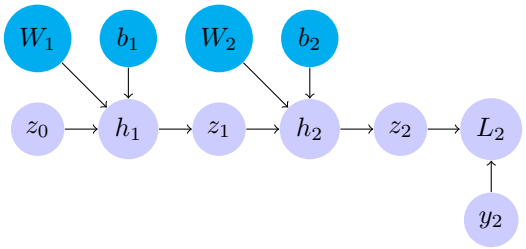

Figure 2: A simple two-layer MLP, expressed in computation graph notation. W_i and b_i are learnable parameters, z_0 is the model's input, h_i and z_i are preactivations and postactivations respectively, y_2 is the expected label expressed as a one-hot vector, z_2 is a probability distribution over output labels, L_2 is the cross entropy $H(y_2, z_2)$.

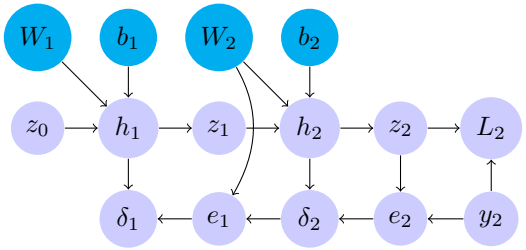

Figure 3: Computational graph for backpropagation for a two-layer MLP. δ_i and e_i are the derivatives of the global loss L_2 with respect to the preactivations and the postactivations of layer i respectively. The fact that W_2 is needed to compute e_2 exemplifies the weight transport problem. The fact that only one loss is defined for the entire network is the global loss problem. The fact that δ_i and e_i are computed in a completely different way compared to h_i and z_i is the asymmetry problem.

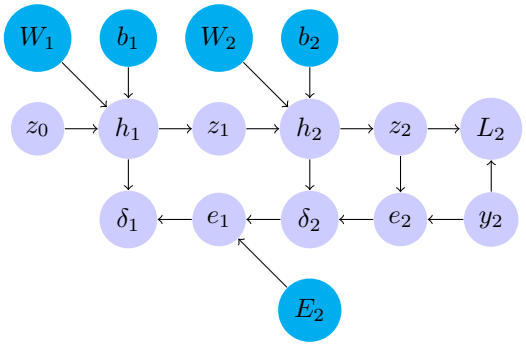

Figure 4: A solution to the weight transport problem in a two-layer MLP, in the Feedback Alignmnet, Weight Mirror and Kolen Pollack algorithms introduce an additional set of weights E_2 and then incentivize E_2 to be similar to $W_2^\top$ during training. This property causes learning to mimic backpropagation and thus can achieve similar results to backpropagation even in deep networks.

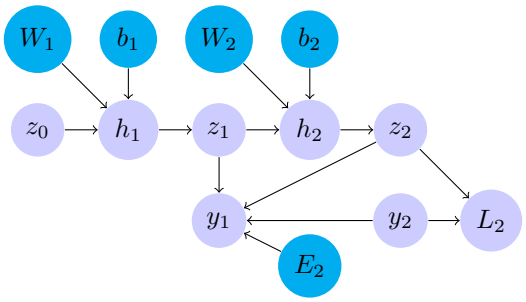

Figure 5: For a two-layer MLP the additional target y_1 is constructed and some discrepancy between z_1 and y_1 is utilized as local loss.

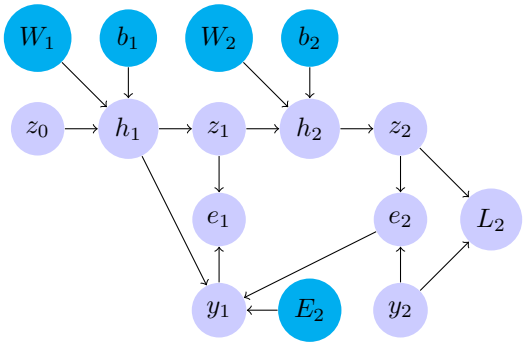

Figure 6: A solution to the asymmetry problem is proposed in the Local Representation Alignment algorithm, where a linear layer parametrized by E_2 is used to predict the preactivation diaplacement on h_1 and then same linearity as in the forward neurons is used to compute the target y_1.