Understanding and Improving Optimization in Predictive Coding Networks

Nicholas Alonso[1], **Jeffrey Krichmar**[1,2], **Emre Neftci**[3,4]

[1]Department of Cognitive Science, University of California, Irvine
[2]Department of Computer Science, University of California, Irvine
[3]Electrical Engineering and Information Technology RWTH Aachen, Germany
[4] Peter Grünberg Institute, Forschungszentrum Jülich, Germany
nalonso2@uci.edu, jkrichma@uci.edu, e.neftci@fz-juelich.edu

Abstract

Backpropagation (BP), the standard learning algorithm for artificial neural networks, is often considered biologically implausible. In contrast, the standard learning algorithm for predictive coding (PC) models in neuroscience, known as the inference learning algorithm (IL), is a promising, bio-plausible alternative. However, several challenges and questions hinder IL's application to real-world problems. For example, IL is computationally demanding, and without memory-intensive optimizers like Adam, IL may converge to poor local minima. Moreover, although IL can reduce loss more quickly than BP, the reasons for these speedups or their robustness remains unclear. In this paper, we tackle these challenges by 1) altering the standard implementation of PC circuits to substantially reduce computation, 2) developing a novel optimizer that improves the convergence of IL without increasing memory usage, and 3) establishing theoretical results that help elucidate the conditions under which IL is sensitive to second and higher-order information.

Introduction

Artificial neural networks (ANNs) were originally created to mimic biological neural circuits (McCulloch and Pitts 1943). However, backpropagation (BP) (Rumelhart et al. 1995), the now standard algorithm used to train ANNs, is difficult to reconcile with neurobiology (Crick 1989; Lillicrap et al. 2020). Researchers in computational neuroscience and neuromorphic computing have attempted to alter BP to better fit what is known about the brain (Lillicrap et al. 2016; Liao, Leibo, and Poggio 2016; Guerguiev, Lillicrap, and Richards 2017; Richards and Lillicrap 2019). However, these altered versions of BP typically either fail to overcome the biological implausibilities of BP, such as BP's segregated feed-back signals that do not alter feed-forward activities, or perform significantly worse than standard BP. One promising alternative approach is based on the predictive coding (PC) model in neuroscience (Rao and Ballard 1999). PC is a type of recurrent neural network typically trained with inference learning (IL). IL is a variant of generalized expectation maximization (Millidge et al. 2022b), which works by first performing inference, where an objective known as free energy is reduced w.r.t. neuron activities via the recurrent PC circuits. Then weights are updated to further reduce free energy. Unlike BP, IL does not require a segregated feedback stream. Instead, its recurrent circuitry requires feed-forward and feedback signals to interact with each other. Further, like the brain and in contrast to BP, IL performs Hebbian-like updates that are local in space and time (Whittington and Bogacz 2017).

Previous work has found that networks trained with IL performed competitively with BP on classification and self-supervised tasks that use small and medium sized images (Whittington and Bogacz 2017; Alonso and Neftci 2021; Salvatori et al. 2021, 2022a; Alonso et al. 2022; Song et al. 2022). Recent work has even found some noticeable performance advantages over BP/SGD (e.g., (Alonso et al. 2022; Song et al. 2022)), such as faster loss reduction and convergence with small mini-batches. These results suggest that IL may be useful in certain machine learning and neuromorphic computing applications, such as online learning scenarios which require small mini-batches. However, there are several challenges preventing IL from easily being applied to engineering problems: **1)** IL is much more computationally expensive than BP due primarily to IL's expensive inference phase. **2)** IL has only achieved comparable performance to BP when memory expensive optimizers like Adam are used. Without any optimizer, IL convergence is prone to fall in poor local minima (Alonso et al. 2022), and the reason why this happens is not well understood. **3)** IL sometimes shows speedups in loss reduction and convergence over BP (Alonso et al. 2022; Song et al. 2022), but the robustness of these speedups across models and tasks is unknown. In this paper, we tackle these challenges through the following contributions:

1. We develop and test a non-standard implementation of IL's inference phase, called sequential inference, which propagates errors more quickly through the network allowing for a significant reduction of the computational cost of IL.
2. We show that IL weight updates are drastically smaller in magnitude in early layers, which may lead the network to get caught in poor local minima near the initial parameters. We develop a custom optimizer called **M**atrix Update E**q**ualization (MQ) to address this problem. MQ requires no significant increase in memory or computation compared to IL and our simulations provide

evidence that MQ prevents the convergence issues of IL at least as well as Adam.

3. We provide simulation results showing IL often reduces the loss more quickly than BP/SGD. We establish new theoretical results showing that IL is generally sensitive to second and higher-order information, which could explain this speed up.

In sum, in our simulations below, IL with sequential inference and the MQ optimizer requires no more memory than BP/SGD, requires only slightly more computation than BP/SGD, converges to similar losses as BP/SGD, and is sensitive to higher-order information, which often leads to faster convergence than BP/SGD. *As far as we know, this is the first time an energy-based learning algorithm (as defined in (Scellier and Bengio 2017; Whittington and Bogacz 2019)) has performed as well or better than BP/SGD on natural images in the ways described above without using memory expensive optimizers and without requiring significantly more computation than BP.* This combined with the fact that IL uses Hebbian-like rules local in space and time suggest IL, with the modifications made here, may be a promising learning algorithm for neuromorphic and bio-inspired machine learning communities.

Background and Notation

Notation

Term	Description
W_l	Weight Matrix, pre-synaptic layer l
h_l	Feedforward Activity layer l, $h_l = W_{l-1} f(h_{l-1})$
$\hat{h}_l$	Optimized/Target Activity layer l
p_l	Local Prediction layer l, $p_l = W_{l-1} f(\hat{h}_{l-1})$
e_l	Local Error, layer l, $e_l = \hat{h}_l - p_l$

Table 1: Notation

Notation describing a multi-layered feed-forward (FF) network (MLP) is summarized in table 1. We assume a bias is stored in an extra column of each weight matrix W_l.

Predictive Coding and Inference Learning

Predictive coding (PC) networks are a kind of recurrent neural network typically trained with the inference learning algorithm (IL). IL can be interpreted as a variant of generalized Expectation Maximization algorithm (EM) (Friston 2008; Millidge et al. 2022b). Like EM, IL proceeds in two steps: First, an energy function, known as free energy is minimized w.r.t. neuron activities. PC computations perform this minimization using gradient descent. Then weights are updated to further reduce free energy. Below, we present the energy function and equations for PC and IL weight updates.

Free Energy: The free energy, F, is defined here as follows:

$$F = \mathcal{L}(y, \hat{h}_L) + \sum_{l=1}^{L} \gamma_l \frac{1}{2} \|\hat{h}_l - W_{l-1} f(\hat{h}_{l-1})\|^2 + \sum_{l=1}^{L-1} \gamma_l^{decay} \frac{1}{2} \|f(\hat{h}_l)\|^2, \tag{1}$$

where $\hat{h}$ are layer activities, $\mathcal{L}$ is the global loss, $\frac{1}{2}\|f(\hat{h}_l)\|^2$ is an optional regularization term, f is a non-linearity, and γ are positive scalar weighting terms. The middle term can be interpreted as a summation over squared prediction errors, where prediction $p_l = W_{l-1} f(\hat{h}_{l-1})$ and prediction errors are $e_l = \hat{h}_l - p_l$. If we set the activities at the output layer equal to the prediction target, $y = \hat{h}_L$ and ignore the decay (set $\gamma^{decay} = 0$), as is common in practice, the energy is just the sum of prediction errors: $F = \sum_{l=1}^{L} \gamma_l \frac{1}{2} \|e_l\|^2$.

Inference: During the inference phase, which is similar to the E-step of EM (Millidge et al. 2022b), neuron activities $\hat{h}$ initialized to feed-forward activities h then are updated to reduce F. PC refers to the the process that updates activities iteratively using partial gradients of local prediction errors. For example, PC circuits update $\hat{h}_l$ using gradients from e_l and e_{l+1}. Specifically, a single gradient update over a hidden layer $\hat{h}_l$ is

$$\Delta \hat{h}_l = \epsilon(-\frac{\partial F}{\partial \hat{h}_l}) = \epsilon(\gamma_{l+1} f'(\hat{h}_l) W_l^T e_{l+1} - \gamma_l e_l - \gamma_l^{decay} f'(\hat{h}_l) \hat{h}_l), \tag{2}$$

where ϵ is the step size. The output layer activities, $\hat{h}_L$, are either fully clamped to output targets ($\hat{h}_L = y$) or are softly clamped such that each iteration $\hat{h}_L$ is updated to reduce the loss $\mathcal{L}(\hat{h}_L, y)$ and $\gamma_L e_L$. Here we assume $\mathcal{L}(\hat{h}_L, y) = \frac{1}{2}\|y - p_L\|^2$, which yields a closed form solution that is computed at each iteration:

$$\hat{h}_L = \frac{1}{(1+\gamma_L)} y + \frac{\gamma_L}{(1+\gamma_L)} p_L. \tag{3}$$

The derivation can be found in appendix. In practice, gradient updates are typically performed for 15+ iterations (e.g., (Whittington and Bogacz 2017; Alonso and Neftci 2021; Salvatori et al. 2022a)).

Learning: After the inference phase is completed, weights are updated to further reduce F. This update is typically performed by using the gradient of the local errors, e.g., W_l is updated with the gradient of $\frac{1}{2}\|e_{l+1}\|^2$:

$$\Delta W_l = -\alpha_l \frac{\partial F}{\partial W_l} = \alpha_l e_{l+1} f(\hat{h}_l)^T, \tag{4}$$

where $e_{l+1} = \hat{h}_{l+1} - W_l f(\hat{h}_l)$ and α_l is a layer-wise step size.

Inference Learning Approximates Implicit Gradient Descent

Previous works have analyzed the similarities between backpropagation (BP) and stochastic gradient descent (SGD) and

IL (e.g., (Whittington and Bogacz 2017)). There, it was found that IL approaches BP/SGD as activities $\hat{h}$ approach feed forward activity values h (Whittington and Bogacz 2017). Although insightful, such analyses leave open the question of why IL is able to reduce the loss in a stable manner even when the $\hat{h}$ deviates significantly from h, which often occurs in practice (Rosenbaum 2022). Recently, an alternative theoretical framework for IL has been established that avoids this issue. In particular, it was shown that, in the case of a mini-batch size of one and under specific conditions on the γ terms in the energy equation and learning rates α_l, IL is equivalent to *implicit* stochastic gradient descent. These conditions on the γ and α terms are non-standard but are approximated well in standard implementations (see (Alonso et al. 2022) and below). We emphasize that implicit SGD is not equivalent to standard SGD, which we call explicit SGD here. The difference between the two can stated as follows:

$$\begin{aligned} &\textbf{Explicit SGD: } \theta^{(b+1)} = \theta^{(b)} - \beta \frac{\partial \mathcal{L}(\theta^{(b)})}{\partial \theta^{(b)}} \\ &\textbf{Implicit SGD: } \theta^{(b+1)} = \theta^{(b)} - \beta \frac{\partial \mathcal{L}(\theta^{(b+1)})}{\partial \theta^{(b+1)}}, \end{aligned} \tag{5}$$

where θ are the model parameters, b is the current training iteration, and β is a 'global' learning rate acting over all parameters. While standard/explicit SGD updates parameters with the gradient of the loss w.r.t. the current parameters, implicit SGD updates parameters with the gradient of the loss w.r.t. the parameters at the next training iteration, $b+1$. This gradient cannot be explicitly computed given known values at iteration b. Hence, this update is computed implicitly. It can be shown that the implicit SGD update is equivalent to the *proximal update* (Parikh and Boyd 2014; Toulis and Airoldi 2014):

$$\theta^{(b+1)} = \text{argmin}_\theta \, \mathcal{L}(\theta) + \frac{1}{2\beta} \|\theta - \theta^{(b)}\|^2. \tag{6}$$

The proximal algorithm sets the new parameters equal to the output of an optimization process that both minimizes the global loss $\mathcal{L}$ and the squared difference between the current and optimized parameters, which keeps the new parameters in the proximity of the old ones. The intuition for why IL approximates the proximal algorithm goes as follows: IL reduces F w.r.t. to neuron activities before updating weights, which does two things: 1) $\hat{h}_L$ is updated to reduce the loss at the output layer (equation 3), which means that when weights are updated the loss is reduced given the same input. 2) Local errors e_l and activity $f(\hat{h}_l)$ magnitudes are minimized, which has the effect of minimizing the magnitude of the weight update since the weight updates are just outer products e_{l+1} and $f(\hat{h}_l)$. Thus, IL's inference phase yields weight updates that both minimize loss and the update norm, just like the proximal update.

Theoretical Results

IL and Implicit SGD

The IL algorithm was shown to be equivalent to the proximal algorithm/implicit SGD under a set of assumptions including that each weight update used a normalized step size:

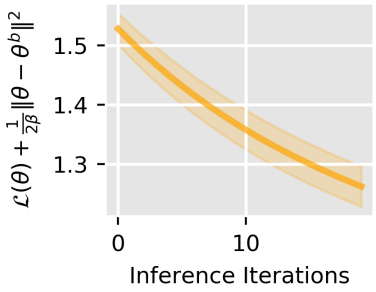

Figure 1: Empirical support for theorem 1. Proximal loss is computed as a function of the number of inference iterations. Consistent with theorem 1, as inference reduces energy F the proximal loss is also reduced in a stable manner in 5 layer MLP.

$\alpha_l = \|\hat{h}_l\|^{-2}$. This normalized step size solves the local prediction problem, such that local errors are minimized to zero given the same input (Alonso et al. 2022) with mini-batch size of 1. The original PC/IL algorithm (Rao and Ballard 1999), however, does not use this normalized step size and instead treats α_l as a static hyper-parameter. Further, below we develop an optimizer that uses static step sizes for each layer. This raises the question of how this class of IL algorithms relate to the proximal algorithm. We show that even when a non-normalized step size is used to update weights, IL approximates implicit SGD under certain settings of the γ terms in the free energy (equation 1) and in a certain limit concerning how much weight updates reduce local errors. In the next sections, we use the implicit SGD interpretation to further describe how IL is distinct from BP/SGD.

Let a 'static' scalar step size, α_l, refer to a scalar that does not change during the inference phase with changes to $\hat{h}_l$ (unlike the normalized step size). Let $\theta_{IL}^{(b+1)}$ be the parameters updated at iteration b by the IL algorithm with static learning rates. Let $\theta_{prox}^{(b+1)}$ be the parameters updated by the proximal algorithm (equation 6). The next theorem states that the IL algorithm with static learning rates is equivalent to the proximal algorithm under specific settings of the γ variables.

Theorem 1. *Consider the IL algorithm at training iteration b with static α_l and mini-batch size one. Assume that at each inference iteration we update the γ terms at hidden layers according to $\gamma_l = \alpha_{l-1}^2 \|f(\hat{h}_{l-1})\|^2$ and $\gamma_l^{decay} = \alpha_l^2 \|e_{l+1}\|^2$, and at the output layer $\gamma_L = \alpha_{L-1}(\frac{1}{\beta} + \|f(\hat{h}_{L-1})\|^2) - 1$. In the limit where $\hat{h}_{L-1}^{(b)} \to h_{L-1}^{(b+1)}$, it is the case that $\theta_{IL}^{(b+1)} = \theta_{prox}^{(b+1)}$.*

The theorem's proof and its extension to larger mini-batches is in appendix. Thus, under these conditions, IL is an implementation of the proximal algorithm. In particular, IL implements the proximal algorithm in an indirect way by defining the parameters, θ, optimized by the proximal update (eq. 6) as a function of hidden/auxiliary variables $\hat{h}$, i.e., $\theta = j(\hat{h})$, which is then optimized through updates to

$\hat{h}$ (see appendix for details). The limit in which this theorem is true is the limit where the last hidden layer activity $\hat{h}_{L-1}^{(b)}$ approaches the feed-forward activity after weights are updated, $h_{L-1}^{(b+1)}$, given the same input $x^{(b)}$. This limit holds in the case where local errors are fully minimized by the weight updates. The proof also depends on particular settings for the γ terms. In practice, these settings are usually not used, but we can see that the γ_l terms will generally be positive scalars and are therefore approximated in practice by weighting each term with some positive scalar (see (Alonso et al. 2022) and figure 1). Further, we can see that $\gamma^{decay} \approx 0$ at most hidden layers, since the magnitude of errors are initially zero and remain small afterward, providing justification for the common practice of neglecting the decay term in F. Finally, the β term, which is the 'global' learning rate in the implicit SGD/proximal update, affects the clamping of the output layer: if $\beta \to 0$ then $\gamma_L \to \infty$ and $\hat{h}_L \to h_L$ (i.e., as β gets smaller the output layer is more softly clamped). If $\beta \to \infty$ then γ_L will get smaller and the output layer will become more strongly clamped.

IL is Sensitive to Second-Order Information

Recent work found that IL, even without optimizers, often reduces loss more quickly than vanilla BP/SGD, especially when small mini-batches are used (Alonso et al. 2022). Relatedly, it was found empirically by multiple works that IL often takes a shorter/more direct path toward local minima than BP/SGD (Alonso et al. 2022; Song et al. 2022). However, it has not been clearly explained mathematically why and how these shorter paths and quicker convergence are achieved. Here we provide some explanation based on the insights of Toulis et al. (Toulis and Airoldi 2014, 2016), who showed that implicit SGD is sensitive to second-order information for small learning rates. Let $\Delta\theta_{IL}^{(b)}$ equal the IL update at iteration b.

Theorem 2. *Consider neural network parameters $\theta^{(b)}$ at iteration b with mini-batch size 1. Assume weight-wise step sizes α_l and the γ variables are set so that $\Delta\theta_{IL}^{(b)} = -\beta\frac{\partial\mathcal{L}(\theta^{(b+1)})}{\partial\theta^{(b+1)}}$. With these assumptions we have $\Delta\theta_{IL}^{(b+1)} = -\beta\frac{\partial\mathcal{L}(\theta^{(b+1)})}{\partial\theta^{(b+1)}} \approx -(I+\beta H)^{-1}\beta\frac{\partial\mathcal{L}(\theta^{(b)})}{\partial\theta^{(b)}}$ with error $\mathcal{O}(\beta^2)$, where $H = \frac{\partial^2\mathcal{L}(\theta^{(b)})}{\partial\theta^{(b)2}}$.*

For proof see appendix. This theorem implies that implicit SGD, and IL when it approximates implicit SGD closely, is sensitive to second-order information. Specifically, IL/implicit SGD approximates a shrinked version of the gradient, where the shrinkage is determined by the inverse Hessian. This use of second-order information, about the curvature of the loss landscape, can help account for why IL often converges more quickly and takes more direct paths toward minima than standard SGD. This theorem also implies that when β is small ($0 < \beta < 1$), IL closely approximates the Newton-Raphson algorithm with 'damping' (scaling by a small learning rate) and Levenburg-Marquardt regularization (Nesterov and Polyak 2006), which is: $\theta^{(b+1)} = \theta^{(b)} - \alpha(\mathbf{I}\lambda + H^{-1})\frac{\partial\mathcal{L}(\theta^{(b)})}{\partial\theta^{(b)}}$, where λ is a scalar weighting term and α the learning rate/damping term. Thus, for small but non-zero β and when IL approximates implicit SGD closely, IL approximates a regularized and damped version of the Newton-Raphson method with small error, $\mathcal{O}(\beta^2)$. Note that as $\beta \to \infty$ the approximation to Newton-Raphson becomes worse because IL becomes increasingly sensitive to even higher-order information in the Taylor-expansion. We find empirically that our implementations of IL reduces the proximal objective well for both large and some small values of β (appendix, supp. fig.), suggesting IL is generally sensitive to second and higher order information.

Reinterpreting IL's Relation to BP and SGD

Previous work found that in the limit where $\hat{h}_l \to h_l$, IL approaches BP/SGD (Whittington and Bogacz 2017, 2019; Millidge et al. 2022a; Rosenbaum 2022). This limit is equivalent, in our notation, to the limit where the $\beta \to 0$ since in this case the output layer $\hat{h}_L$ approaches its initial value h_L, resulting in $\hat{h}_l \to h_l$. Notice that this is exactly what theorem 2 implies: as $\beta \to 0$ the Hessian term βH and the error, $\mathcal{O}(\beta^2)$, go to zero, and we are left with a standard SGD update. *Thus, consistent with these previous results, the implicit SGD interpretation predicts that IL should approach SGD as $\beta \to 0$ and $\hat{h} \to h$. However, theorems 1 and 2 imply that standard/explicit SGD is not the best interpretation of IL when $\beta > 0$, since the SGD approximation has a larger error, $\mathcal{O}(\beta)$, than approximations that take into account higher order terms, which have at most error of $\mathcal{O}(\beta^2)$.*

The Matrix Update Equalization Optimizer

We present a novel optimizer, called Matrix Update Equalization (MQ), made specifically for the IL algorithm. This optimizer is motivated in part by our theorem 2. Previous work (e.g. (Alonso et al. 2022), also see below) observed that IL often converges to poor local minima when it does not use optimizers with parameter-wise adaptive learning rates, like Adam. Parameter-wise adaptive learning rates were designed to allow BP, a first-order method, to gain sensitivity to second-order information (Kingma and Ba 2014). However, theorem 2 shows IL algorithms that use only matrix-wise scalar learning rates are sensitive to second and higher-order information, suggesting parameter-wise learning rates are not needed in IL to gain this sensitivity. Second, IL updates, without the use of Adam, tend to be very small[1], especially at early layers (figure 2), which likely explains why IL gets caught in shallow local minima nearer its initial values. Adam makes updates larger and more equal across matrices (figure 2), suggesting the main benefit of using Adam with IL is that it prevents weight updates at early layers from becoming too small, instead of making it sensitive to second-order information.

Motivated by these observations, MQ uses only a single scalar learning rate per layer that adapts to prevent overly

[1]It is worth noting that IL's tendency for small update magnitudes is consistent with the proximal interpretation, and similar observations about decaying errors have been observed previously by Song et al. (Song et al. 2020).

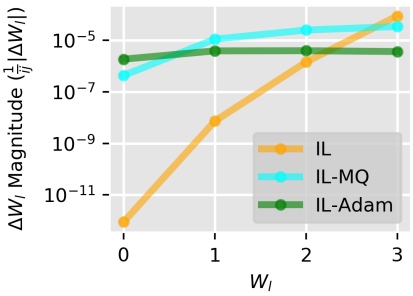

Figure 2: Average update magnitude of each weight matrix in MLP with three hidden layers, T=15.

small update magnitudes and to better equalize the update magnitudes between matrices. By only using a single scalar learning rate we drastically reduce memory without removing sensitivity to higher-order information, while still gaining the improved convergence properties of Adam. MQ stores several scalars: a constant matrix-wise learning rate, α_l, for each matrix W_l, a constant global minimum learning rate, α_{min}, and matrix wise scalar v_l, which is updated continuously to adapt to the moving average of the update magnitude for matrix W_l. The MQ weight update for weight matrix W_l is

$$W_l^{(b+1)} = W_l^{(b)} - (\frac{\alpha_l}{v_l + r} + \alpha_{min})\frac{\partial F}{\partial W_l^{(b)}}, \qquad (7)$$

where r is a constant that prevents division by zero and stabilizes learning. The minimum learning rate α_{min} is used to prevent learning from decreasing too much in any one matrix. The variable v_l is updated each training iteration b as follows:

$$v_l^{(b+1)} = (1-\rho)v_l^{(b)} - \rho\frac{1}{ij}|\frac{\partial F}{\partial W_l^{(b)}}|, \qquad (8)$$

where i and j refer to the row and column sizes, respectively, of the gradient $\frac{\partial F}{\partial W_l}$ and $|\frac{\partial F}{\partial W_l^{(b)}}|$ is the L-1 norm of the gradient. Finally, ρ is a global hyper-parameter (scalar). Note, although ρ is a hyper-parameter, we use a method for reducing bias early in training, which is explained in appendix. Thus, v_l first averages the update over elements of the absolute value of the gradient (right term), then it takes a weighted average of this with the previous value of v_l. Figure 2 shows MQ reduces the differences in update magnitude between weights matrices nearly as well as Adam and prevents updates at early layers form becoming too small.

Reducing Computation with Sequential Inference

The inference phase accounts for most of the computational cost of the IL algorithm. Typically, activities are updated for $T = 15$ or more iterations (e.g., (Whittington and Bogacz 2017; Alonso and Neftci 2021; Salvatori et al. 2022a)), which requires around $T(2L-1)$ matrix multiplications. The obvious way to reduce computation is to truncate the inference phase (reduce T). The difficulty with this approach is that the standard implementation of the inference phase requires at least $T \geq (L-2)$ in order for non-zero errors to form at each hidden layer (see supp. figures), and typically more iterations to attain the best performance (see experiments section), e.g., a network with four hidden layers typically requires at least $T \geq 4$, and usually much more than 4 to perform well. In the standard implementation requires large T because there is a delay in error propagating through the network (see supp. fig. 6). Each inference iteration, t, the standard method computes and stores errors at each layer, then these stored errors are used to update the activities according to equation 2. This process simulates activities being updated simultaneously in time, since the update at one layer does not alter the updates or errors at other layers until the next iteration/time step. Instead of updating activities simultaneously, we propose updating them one at a time starting from the output layer and moving back in a sequence. We call this method *sequential inference*. Sequential inference removes the delay and progagates non-zero errors to every layer in a single iteration, allowing for smaller T. [2]

Experiments

In this section, we test the performance of sequential inference and the MQ optimizer. The MQ optimizer works well in all models using following hyper-parameters: $\alpha_{min} = .001$, $r = .000001$, and $\rho = .9999$ for fully connected and $\rho = .999$ for convolutional networks. Grid searches are used to find learning rates, α_l. Mini-batches size 64 are used in all simulations.

Comparing Sequential and Simultaneous IL The standard/simultaneous inference method creates a delayed error propagation through the network, as illustrated in figure 4. We test how this delay effects performance at very small values of T on a classification of CIFAR-10 images. An MLP with four hidden layers, dimension 3072-4x1024-10, was trained on CIFAR-10 using both standard/simultaneous inference with Adam optimization (IL-Adam) and sequential inference with Adam optimization (SeqIL-Adam) for different values of T. Grid searches were used to find the learning rate and step size, ϵ, for activity updates. Models are trained over 45 epochs, about the amount of time needed for learning to near convergence. Results shown in figure 4. SeqIL-Adam performs approximately the same for all values of T (1-5), while IL-Adam's performance is highly sensitive to the value of T, with performance degrading significantly for smaller values of T.

Classification with Natural Images Next, we test the MQ optimizer combined with SeqIL on classifications tasks with natural image data sets: SVHN, CIFAR-10, and Tiny Imagenet. We train fully connected MLPs dimension 3072-3x1024-10 on SVHN and CIFAR-10, and small convolutional networks on SVHN, CIFAR-10, and Tiny Imagenet.

[2] Though this is a simple alteration, we could only find one work where an implementation similar to sequential inference is used (Rosenbaum 2022), and in that work this implementation was not discussed in the paper, analyzed, or used as a method for reducing computation. Instead, it seemed to be an arbitrary implementation decision.

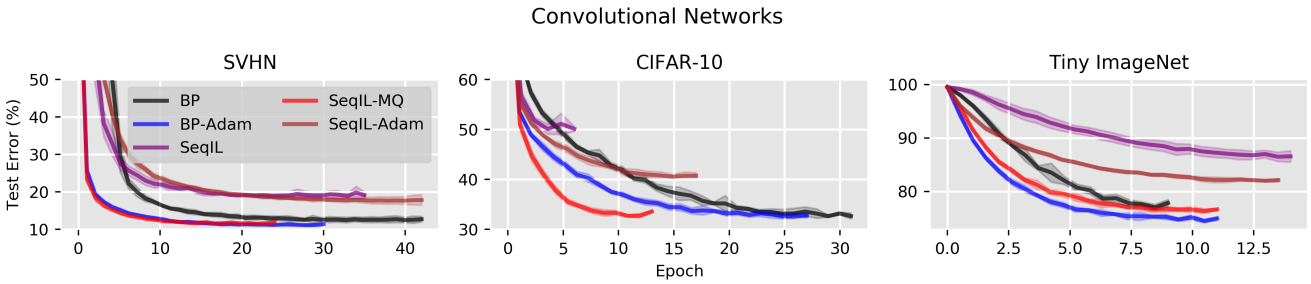

Figure 3: Small convolutional networks trained on classification tasks. Training runs are averaged over 5 seeds. Shaded error bars show standard deviation across seeds. Each training run is shown up until convergence (the point of lowest test error +1 epoch).

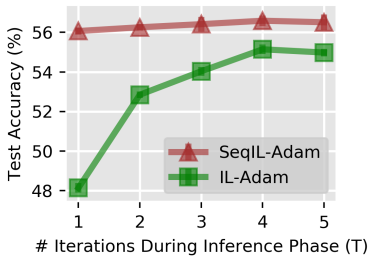

Figure 4: MLPs trained on CIFAR-10 for 45 epochs, using either sequential inference with Adam (SeqIL-Adam) or simultaneous inference with Adam (SimIL-Adam). X-axis specifies the number of inference iterations used during training.

All of IL algorithms use a highly truncated inference phases of T=3. In addition to sequential IL with MQ (SeqIL-MQ), we test SeqIL with no optimizer (SeqIL) and SeqIL with Adam (SeqIL-Adam). We compare to the performance of BP-SGD and BP with Adam (BP-Adam). Grid searches over learning rates were performed to find the learning rate that yielded the best test performance at convergence. SeqIL-MQ test accuracies were never significantly worse than BP and slightly better than BP on Tiny ImageNet, while consistently performing comparably to BP-Adam (table). Further, SeqIL-MQ reduced loss more quickly early in training than BP-SGD in all simulations, and typically converged much more quickly (figure 3). SeqIL-MQ learning curves looked more similar to BP-Adam, despite the fact SeqIL-MQ uses about one third the memory of Adam per parameter. SeqIL-Adam matched the performance of SeqIL-MQ in fully connected networks but struggled in convolutional networks. We believe this is because in convolutional networks SeqIL-Adam had very small weight updates, which could not be increased by simply increasing learning rate due to instability (see appendix). Consistent with previous work on IL, we find that SeqIL without optimizers often reduces loss more quickly than BP early in training in MLPs, but struggled to converge to good minima.

Autoencoders We trained fully connected autoencoders (layer sizes 3072-1024-256-20-256-1024-3072) with ReLU at hidden layers and sigmoid at the output layer. Networks were trained on SVHN and CIFAR-10. A grid search was used to find a learning rate that produces the lowest test loss at convergence or after 100 epochs. Testing auto-encoders is important because, unlike classifiers, their output layers are large, which may make the task of propagating errors through the network easier for IL algorithms. IL models use T=6 inference iterations. We find that all IL models reduce the loss significantly faster than both BP and BP-Adam early in training (figure 5, right two plots). IL-MQ and IL-Adam converge to similar reconstruction losses as the BP algorithms.

Related Work

Theoretical work on IL Several recent works develop theoretical analyses of IL. Song et al. (Song et al. 2022) argue IL and similar energy-based algorithms, like Constrastive Hebbian Learning (Movellan 1991), work differently than BP. Unlike BP, these algorithms first compute the desired neural activities, then update parameters to consolidate these neural activities (i.e., make them more probable). They call this approach 'prospective configuration'. They propose the brain likely learns in a way similar to prospective configuration and provide empirical support. Millidge et al. (Millidge et al. 2022b) provide a formal analysis showing the neural activities computed by prospective configuration algorithms are similar to regularized Gauss-Newton targets, and IL is a variant EM algorithm known as maximum a posteriori learning. Alonso et al. (Alonso et al. 2022) developed an alternative, though compatible, analyses of IL that showed its approximation to the proximal algorithm/implicit SGD, and further examined the differences between IL and BP such as differences in stability across learning rates. The theoretical work here expands on these previous findings by showing 1) the results of Alonso et al. extend approximately to the more general case where a static learning rate is used at each layer (instead of a dynamic normalized learning rate) and 2) for the first time shows that, under the assumption IL is closely approximating implicit SGD, describes how IL is sensitive to higher order information. We believe our results provide a deeper understanding of how IL differs from BP/SGD than these previous works.

Algorithmic Work on IL The IL algorithm has recently been altered in several ways. One line of work altered IL in

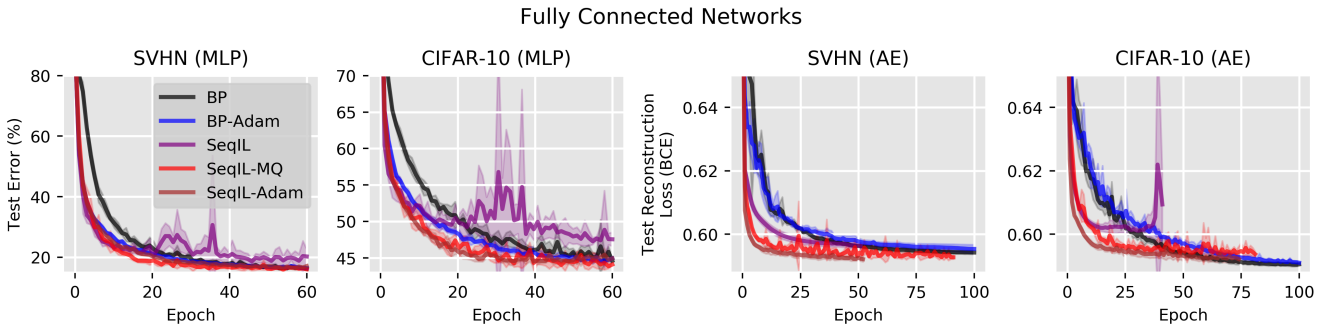

Figure 5: Fully connected networks trained for classification tasks (MLP) and for autoencoder tasks (AE). Training runs are averaged over 5 seeds. Shaded error bars show standard deviation across seeds. The first 60 epochs of training shown for MLPs. For AEs, training runs are shown to convergence (lowest loss +1 epoch) or the max 100 epochs.

Classification Test Accuracies						
	SVHN (FC)	SVHN (Conv)	CIFAR (FC)	CIFAR (Conv)	TinyImNet Top 1	Top 5
BP	**84.40**$^{(\pm.12)}$	87.82$^{(\pm.67)}$	56.55$^{(\pm.08)}$	**67.76**$^{(\pm.43)}$	23.14$^{(\pm.42)}$	47.05$^{(\pm.46)}$
BP-Adam	84.08$^{(\pm.13)}$	**89.02**$^{(\pm.19)}$	56.15$^{(\pm.21)}$	**67.51**$^{(\pm.45)}$	**25.61**$^{(\pm.22)}$	**50.04**$^{(\pm.41)}$
SeqIL	82.19$^{(\pm.70)}$	81.55$^{(\pm.59)}$	54.24$^{(\pm.18)}$	50.67$^{(\pm.76)}$	13.88$^{(\pm.73)}$	33.42$^{(\pm1.16)}$
SeqIL-MQ	**84.51**$^{(\pm.17)}$	88.67$^{(\pm.18)}$	**57.70**$^{(\pm.21)}$	**67.50**$^{(\pm.19)}$	23.91$^{(\pm.31)}$	47.53$^{(\pm.48)}$
SeqIL-Adam	84.13$^{(\pm.15)}$	82.76$^{(\pm.87)}$	56.21$^{(\pm.20)}$	59.72$^{(\pm.48)}$	18.22$^{(\pm.29)}$	39.58$^{(\pm.46)}$

Table 2: Best test accuracies averaged across 5 seeds (mean $\pm$ standard dev.). Top scores and scores not significantly different from top score (according to two-sample t-test) are bolded.

order to make its weight updates more similar to SGD (e.g., (Song et al. 2020; Millidge, Tschantz, and Buckley 2020)). Alonso et al. altered IL to make its weight updates better approximate the proximal algorithm/implicit SGD. Our alterations to IL differ from these previous works, as they do not seek to alter IL to make it more similar to SGD or implicit SGD. Salvatori et al. (Salvatori et al. 2022b) developed a variant of IL based on incremental EM (Neal and Hinton 1998), called incremental PC (iPC), which works by updating weights each inference iteration instead of at the end of the inference phase. iPC has some benefits, but still suffers from the delayed error propagation problem and is implemented in (Salvatori et al. 2022b) with Adam. It is unclear whether iPC avoids convergence issues when Adam is not used. iPC also requires more matrix multiplications each training iteration than SeqIL, given the same number of inference iterations (T). Our goal was to reduce computation of IL while improving convergence, so we do not test incremental PC here. (See appendix for further comparison).[3]

Limitations

We provide comments on relation between the MQ optimizer and probabilistic interpretations of IL in appendix, but further analysis is needed. We did not analyze the parallelizability of seq-IL compared to IL. We think seq-IL may be less parallelizable than standard IL, which could be a drawback on certain hardware.

Discussion

Local learning algorithms have historically faced at least two central challenges: 1) difficulty converging to test losses as good as BP/SGD when scaled to natural images (Bartunov et al. 2018) and/or 2) requiring far more computation than BP/SGD. A common method for dealing with one or both of these issues is to make one's bio-plausible algorithm of choice more similar to BP/SGD, e.g., (Scellier and Bengio 2017; Song et al. 2020; Millidge, Tschantz, and Buckley 2020; Laborieux et al. 2021; Ernoult et al. 2022). In this paper, we took a different strategy. We took a biologically constrained, local algorithm, IL, and showed its differences from BP/SGD are not undesirable properties that ought to be removed. Instead, IL differs from BP/SGD because it approximates a different optimization method, which has useful sensitivity to higher-order information. On the basis of our analysis, we altered the way its recurrent processing worked to significantly reduce computation and created a custom optimizer made specifically to address the convergence issue of IL, similar to the way many standard optimizers, e.g., Adam, were specifically created to address shortcomings of BP/SGD. The result is an algorithm that, in our simulations, matched BP/SGD in terms of final test accuracy/loss, memory, and training time (in seconds) (see appendix), while reducing loss and converging more quickly than BP/SGD. These results suggest IL is a promising alternative for bio-inspired machine learning.

[3]The paper with full appendix can be found at https://arxiv.org/abs/2305.13562

References

Alonso, N.; Millidge, B.; Krichmar, J.; and Neftci, E. O. 2022. A Theoretical Framework for Inference Learning. *Advances in Neural Information Processing Systems*, 35: 37335–37348.

Alonso, N.; and Neftci, E. 2021. Tightening the Biological Constraints on Gradient-Based Predictive Coding. In *International Conference on Neuromorphic Systems 2021*, 1–9.

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

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

Ernoult, M.; Normandin, F.; Moudgil, A.; Spinney, S.; Belilovsky, E.; Rish, I.; Richards, B.; and Bengio, Y. 2022. Towards Scaling Difference Target Propagation by Learning Backprop Targets. *arXiv preprint arXiv:2201.13415*.

Friston, K. 2008. Hierarchical models in the brain. *PLoS computational biology*, 4(11): e1000211.

Guerguiev, J.; Lillicrap, T. P.; and Richards, B. A. 2017. Towards deep learning with segregated dendrites. *Elife*, 6: e22901.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Laborieux, A.; Ernoult, M.; Scellier, B.; Bengio, Y.; Grollier, J.; and Querlioz, D. 2021. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15: 129.

Liao, Q.; Leibo, J.; and Poggio, T. 2016. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lillicrap, T. P.; Cownden, D.; Tweed, D. B.; and Akerman, C. J. 2016. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1): 1–10.

Lillicrap, T. P.; Santoro, A.; Marris, L.; Akerman, C. J.; and Hinton, G. 2020. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6): 335–346.

McCulloch, W. S.; and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5: 115–133.

Millidge, B.; Song, Y.; Salvatori, T.; Lukasiewicz, T.; and Bogacz, R. 2022a. Backpropagation at the Infinitesimal Inference Limit of Energy-Based Models: Unifying Predictive Coding, Equilibrium Propagation, and Contrastive Hebbian Learning. *arXiv preprint arXiv:2206.02629*.

Millidge, B.; Song, Y.; Salvatori, T.; Lukasiewicz, T.; and Bogacz, R. 2022b. A Theoretical Framework for Inference and Learning in Predictive Coding Networks. *arXiv preprint arXiv:2207.12316*.

Millidge, B.; Tschantz, A.; and Buckley, C. L. 2020. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*.

Movellan, J. R. 1991. Contrastive Hebbian learning in the continuous Hopfield model. In *Connectionist models*, 10–17. Elsevier.

Neal, R. M.; and Hinton, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 355–368.

Nesterov, Y.; and Polyak, B. T. 2006. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1): 177–205.

Parikh, N.; and Boyd, S. 2014. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 127–239.

Rao, R. P.; and Ballard, D. H. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1): 79–87.

Richards, B. A.; and Lillicrap, T. P. 2019. Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54: 28–36.

Rosenbaum, R. 2022. On the relationship between predictive coding and backpropagation. *Plos one*, 17(3): e0266102.

Rumelhart, D. E.; Durbin, R.; Golden, R.; and Chauvin, Y. 1995. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, 1–34.

Salvatori, T.; Pinchetti, L.; Millidge, B.; Song, Y.; Bogacz, R.; and Lukasiewicz, T. 2022a. Learning on Arbitrary Graph Topologies via Predictive Coding. *arXiv preprint arXiv:2201.13180*.

Salvatori, T.; Song, Y.; Hong, Y.; Sha, L.; Frieder, S.; Xu, Z.; Bogacz, R.; and Lukasiewicz, T. 2021. Associative Memories via Predictive Coding. *Advances in Neural Information Processing Systems*, 34.

Salvatori, T.; Song, Y.; Millidge, B.; Xu, Z.; Sha, L.; Emde, C.; Bogacz, R.; and Lukasiewicz, T. 2022b. Incremental Predictive Coding: A Parallel and Fully Automatic Learning Algorithm. *arXiv preprint arXiv:2212.00720*.

Scellier, B.; and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11: 24.

Song, Y.; Lukasiewicz, T.; Xu, Z.; and Bogacz, R. 2020. Can the brain do backpropagation? exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33: 22566.

Song, Y.; Millidge, B. G.; Salvatori, T.; Lukasiewicz, T.; Xu, Z.; and Bogacz, R. 2022. Inferring Neural Activity Before Plasticity: A Foundation for Learning Beyond Backpropagation. *bioRxiv*.

Toulis, P.; and Airoldi, E. M. 2014. Implicit stochastic gradient descent for principled estimation with large datasets. *ArXiv e-prints*.

Toulis, P.; and Airoldi, E. M. 2016. Stochastic Gradient Methods for Principled Estimation with Large Datasets.

Whittington, J. C.; and Bogacz, R. 2017. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5): 1229–1262.

Whittington, J. C.; and Bogacz, R. 2019. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3): 235–250.