A Theoretical Framework for Inference Learning

Nick Alonso[1]*, **Beren Millidge**[3], **Jeff Krichmar**[1,2], **Emre Neftci**[1,2,4]

[1]Department of Cognitive Science, UC Irvine
[2]Department of Computer Science, UC Irvine
[3]MRC Brain Network Dynamics Unit, University of Oxford
[4]Electrical Engineering and Information Technology, RWTH Aachen, Germany
and Peter Grünberg Institute, Forschungszentrum Jülich, Germany

Abstract

Backpropagation (BP) is the most successful and widely used algorithm in deep learning. However, the computations required by BP are challenging to reconcile with known neurobiology. This difficulty has stimulated interest in more biologically plausible alternatives to BP. One such algorithm is the inference learning algorithm (IL). IL trains predictive coding models of neural circuits and has achieved equal performance to BP on supervised and auto-associative tasks. In contrast to BP, however, the mathematical foundations of IL are not well-understood. Here, we develop a novel theoretical framework for IL. Our main result is that IL closely approximates an optimization method known as implicit stochastic gradient descent (implicit SGD), which is distinct from the explicit SGD implemented by BP. Our results further show how the standard implementation of IL can be altered to better approximate implicit SGD. Our novel implementation considerably improves the stability of IL across learning rates, which is consistent with our theory, as a key property of implicit SGD is its stability. We provide extensive simulation results that further support our theoretical interpretations and find IL achieves quicker convergence when trained with mini-batch size one while performing competitively with BP for larger mini-batches when combined with Adam.

1 Introduction

Backpropagation (BP) [33], the most successful and ubiquitously used learning algorithm in deep learning, has long been criticized as being incompatible with known neurobiology [10, 21]. For example, unlike the highly interconnected, recurrent circuits in the brain, BP performs credit assignment through a separate feedback stream that does not alter feedforward (FF) signals [10, 21]. Additionally, standard implementations of BP are largely incompatible with energy efficient, neuromorphic hardware, due, in part, to BP's non-local gradient computations. These concerns have motivated neuroscientists and engineers to search for alternatives to BP that use more biologically compatible, local learning rules.

One alternative to BP is the inference learning algorithm (IL) [32, 43, 37]. IL is the standard learning algorithm used to train predictive coding (PC) models of biological neural circuits (e.g. [32, 13, 39, 4, 1, 7]). Within neuroscience, PC models have grown in popularity, where some even propose PC may be a canonical biological circuit [15, 14, 4]. IL works by first minimizing an energy function, known as free energy, w.r.t. neuron activities using the recurrent computations involved in predictive coding. At convergence, IL updates weights to further minimize energy. Unlike standard BP, IL uses local learning rules and recurrent circuits instead of a separate feedback stream to perform

*Correspondence to nalonso2@uci.edu

credit assignment. In addition to evidence the brain does predictive coding (and thus may do IL), recent empirical work has found evidence the brain may learn by first inferring target neural activities through recurrent processing then updating synapses to consolidate those neural activities, similar to IL [38]. IL has also been used to train deep networks for machine learning tasks. IL, for example, achieved comparable classification and recall accuracy to BP trained networks on supervised and auto-associative tasks (e.g, [43, 2, 36, 35], see also below).

Despite IL's wide use in biological models and success on machine learning tasks, a rigorous formal understanding of IL's optimization properties and their differences from BP is lacking. Providing such a characterization could yield several useful contributions: 1) A basis for a novel theory of how optimization and credit assignment may work in the brain, 2) a formal basis for improving, developing, and applying IL to machine learning tasks and neuromorphic hardware.

In this paper, we develop a novel theoretical framework for IL that describes its optimization properties and their similarities and differences from BP. More specifically, 1) we expand beyond current literature by deriving a general form of the IL algorithm, which we call Generalized IL (G-IL). 2) We demonstrate our main result, which is that G-IL closely approximates an optimization method known as *implicit stochastic gradient descent* (implicit SGD), where the approximation is closest in the case of mini-batch size 1. Implicit SGD is distinct from explicit SGD (the standard form of SGD used in machine learning and the sort implemented by BP). 3) We provide theoretical guarantees concerning IL's stability and a connection to Gauss-Newton optimization. 4) We identify a learning rule and variable settings needed in order for IL to equal implicit SGD and use this result to develop a novel implementation of IL called IL-prox. 5) Finally, we present extensive simulation results that support our theoretical findings, and for the first time show certain performance advantages of IL over BP, such as better stability across learning rates and improved convergence speed with biologically realistic online learning tasks. These results collectively suggest that, not only does IL better fit biological constraints, but it is also mathematically justified and has performance advantages over BP in more biologically realistic training scenarios.

2 Related Works

Inference Learning and Backpropagation Whittington and Bogacz [43] proved that parameter updates performed by the IL algorithm approach those of BP as the global loss approaches zero and optimized activities approach FF activities. This proof also implies, however, that non-zero IL updates essentially never equal those of BP, and suggests IL poorly approximates BP early in training when the loss is large and optimized activities deviate significantly from FF activities. This leaves open the question of whether there is a better description of IL's optimization strategy and the question of how IL is able to minimize the loss in a stable manner early in training. Other work has shown IL is formally related to variational expectation maximization [25, 22], which is a learning algorithm with convergence guarantees [27]. This analysis is a step forward though leaves open the question of how IL relates to BP and the broader framework of gradient-based methods that are the backbone of deep learning. The focus of the current work is to develop insights into these open questions. More recently, several works have altered IL to more closely approximate SGD and BP [26, 37]. These variants significantly change the original IL algorithm, and some of the alterations seem hard to reconcile with neurobiology [26, 37]. Alternatively, instead of altering IL to better fit SGD and BP, we show the standard implementation of IL already closely approximates a working optimization method known as implicit SGD, and we identify the variable settings under which the standard implementation of IL better approximates implicit SGD.

Similar Algorithms Algorithms similar to IL have been proposed. For example, the alternating minimization algorithm [9], which was developed independently of the predictive coding and IL framework from neuroscience, updates weights to minimize local prediction errors similar to IL (see equation 4 below). However, [9] do not connect their algorithm to implicit SGD. A more recent variant of this algorithm by Qiao et al. [31] show some formal links to proximal operators, an optimization process related to implicit SGD [30] (see appendix B.1). However, they do not interpret the weight updates as performing the proximal update or implicit SGD as we do here with IL. Proximal operators have also recently been incorporated into learning algorithms for deep networks. Frerix et al. [12] developed a BP, proximal algorithm hybrid. Lau et al. [17] developed an algorithm that combined block coordinate descent with the proximal operator. Other works use proximal operators to reduce noise or perform other tasks, then BP is used to to update at convergence

[23, 45]. All of these algorithms, however, are distinct from IL, as they use BP or other non-local gradient information to compute weight updates or local targets.

Target Propagation Finally, IL has some relation to target propagation algorithms, in which local target activities are computed and used to compute local error gradients to update weights. Target propagation [5] and difference target propagation [19] utilize approximate Gauss-Newton updates on activities to create local targets [24, 6], while other algorithms compute targets using gradient updates on activities [18, 29, 3, 16]. These algorithms have important differences from IL, e.g., they use different learning rules than IL, which we discuss further below. In sum, previous works have not developed the same mathematical interpretation of IL that we do here and have not developed the same mathematical descriptions of the differences between IL and BP. Further, none of the works mentioned above produced the same empirical findings we do in our simulations below.

3 Background

3.1 Notation

Term	Description
W_n	Weight Matrix, pre-synaptic layer n
h_n	Feedforward Activity layer n, $h_n = W_{n-1}f(h_{n-1})$
$\hat{h}_n$	Optimized/Target Activity layer n
Δh_n	Optimized Activity Change layer n, $\Delta h_n = \hat{h}_n - h_n$
p_n	Local Prediction layer n, $p_n = W_{n-1}f(\hat{h}_{n-1})$
e_n	Local Error, layer n, either $e_n = \hat{h}_n - h_n$ or $e_n = \hat{h}_n - p_n$

Table 1: Notation

Notation describing a multi-layered feed-forward (FF) network (MLP) is summarized in table 1. To simplify notation, we assume that the bias is stored in an extra column on each weight matrix W_n and we assume a 1 is concatenated to the end of the pre-synaptic activity vector (h_n or $\hat{h}_n$ depending). All the following results should hold with and without biases.

3.2 Generalized Backpropagation

Consider a set of parameters $\theta^{(b)}$ at training iteration b with global loss function L. An explicit SGD iteration subtracts the gradient of the loss from the parameters: $\theta^{(b+1)} = \theta^{(b)} - \alpha \frac{\partial L}{\partial \theta^{(b)}}$, with step size α. BP [34] is the common algorithm for implementing SGD in deep networks. Here, to allow for easier comparison to IL, we consider an alternative implementation of SGD in deep networks that involves computing local target activities at hidden and output layers, similar to [18], [3], [29]. Under this regime, each weight matrix W_n is updated using the gradient of local loss $l_{n+1} = \frac{1}{2}\|\hat{h}_{n+1} - h_{n+1}\|^2 = \frac{1}{2}\|e_{n+1}\|^2$. Here $\hat{h}_{n+1}$ is the local target, while h_{n+1} is the FF activity. The target at the output layer is computed $\hat{h}_N = h_N - \frac{\partial L}{\partial h_N}$. At hidden layer n, the local target is computed

$$\hat{h}_n = h_n - \frac{\partial l_{n+1}}{\partial h_n} = h_n - \frac{\partial L}{\partial h_n}. \tag{1}$$

It can be shown $h_n - \frac{\partial l_{n+1}}{\partial h_n} = h_n - \frac{\partial L}{\partial h_n}$ by first computing $\hat{h}_{N-1}$:

$$\begin{aligned}\hat{h}_{N-1} &= h_{N-1} - \frac{\partial l_N}{\partial h_{N-1}} = h_{N-1} - \frac{\partial l_N}{\partial h_N}\frac{\partial h_N}{\partial h_{N-1}} = h_{N-1} + e_N \frac{\partial h_N}{\partial h_{N-1}} \\ &= h_{N-1} + (h_N - \frac{\partial L}{\partial h_N} - h_N)\frac{\partial h_N}{\partial h_{N-1}} = h_{N-1} - \frac{\partial L}{\partial h_{N-1}}\end{aligned} \tag{2}$$

This same process can be applied recursively to show the remaining $\hat{h}_n$ are also global loss gradient steps over h_n. See appendix of [3] for similar proof.

Updates to weights W_n also use the gradient of the local loss such that $\Delta W_n = -\alpha \frac{\partial l_{n+1}}{\partial W_n}$. If we use the result from the above derivation that $-\frac{\partial l_n}{\partial h_n} = -\frac{\partial L}{\partial h_n}$, we can see this update is equivalent to a global loss gradient step over the weights:

$$\Delta W_n = -\alpha \frac{\partial l_{n+1}}{\partial h_{n+1}} \frac{\partial h_{n+1}}{\partial W_n} = -\alpha \frac{\partial L}{\partial h_{n+1}} \frac{\partial h_{n+1}}{\partial W_n} = -\alpha \frac{\partial L}{\partial W_n} = \alpha e_{n+1} f(h_n)^T. \quad (3)$$

This local target formulation of SGD is similar to other target propagation (TP) [5, 19] algorithms, which use essentially the same learning rule but compute targets slightly differently. For example, a well-known variant of TP is difference target propagation (DTP) [19], which computes local targets by approximating a Gauss-Newton update on activities rather than gradient updates [24, 6]: $\hat{h}_n \approx h_n - J^+_{N,n} e_N$, where $J^+_{N,n}$ is the pseudo-inverse of the Jacobian of the forward network from layer n to N and $e_N = y - h_N$. Recent attempts to improve DTP generally do so by making DTP updates more similar to SGD updates (e.g. [24, 11]). Additionally, variants of BP algorithms, like random feedback alignment [20] and sign symmetric feedback alignment [44], approximate SGD. Since all of these methods approximate the (explicit) SGD update over weights and use similar learning rules, we consolidate them all under the heading of generalized BP (G-BP). The target-based formulation of G-BP is shown in algorithm 1.

3.3 Generalized Inference Learning

We now present a general description of the IL algorithm, which we call Generalized Inference Learning (G-IL). A summary of the algorithm is presented in algorithm 2. At each layer n of an MLP, G-IL stores two variables: $\hat{h}_n$ and p_n. At initialization these variables are set to feed-forward activity values: $\hat{h}_n = p_n = h_n$. Then $\hat{h}_n$ are altered over time to minimize the energy function F:

$$F = L(y, \hat{h}_N) + \sum_{n=1}^{N} \gamma_n \frac{1}{2} \|\hat{h}_n - p_n\|^2 + \sum_{n=1}^{N-1} \gamma_n^{decay} \frac{1}{2} \|\hat{h}_n\|^2, \quad (4)$$

where L is the global loss, $\frac{1}{2}\|\hat{h}_n - p_n\|^2$ are local losses, $\gamma_n^{decay}\frac{1}{2}\|\hat{h}_n\|^2$ is an optional regularization term, and γ are positive scalar weighting terms. The prediction p_n is computed as $p_n = W_{n-1} f(\hat{h}_{n-1})$. (Note p_n may also be computed as $p_n = f(W_{n-1}\hat{h}_{n-1})$. Our theoretical results use the first formulation, but we find little difference in performance between the two in practice. See simulations below.) G-IL first minimizes F w.r.t. to activities $\hat{h}$ (inference phase). While optimizing $\hat{h}$, predictions also change since $p_{n+1} = W_n f(\hat{h}_n)$. After $\text{argmin}_{\hat{h}} F$ is estimated, F is again minimized, now w.r.t. the weight, i.e. $\text{argmin}_W F$. Typical IL algorithms use SGD to perform the inference phase with either the LMS rule (below) or Adam optimizers to update weights (e.g. [32, 43]). G-IL, on the other hand, is general w.r.t. the optimization processes that approximate $\text{argmin}_W F$ and $\text{argmin}_{\hat{h}} F$. We consider two weight updates in our analysis and simulations below. First, is a local error gradient update, which is equivalent to the least-mean squares (LMS) rule:

$$\text{argmin}_{W_n} F \approx W_n + \Delta W_n = W_n - \alpha \frac{\partial l_{n+1}}{\partial W_n} = W_n + \alpha e_{n+1} f(\hat{h}_n)^T, \quad (5)$$

where $e_{n+1} = \hat{h}_{n+1} - p_{n+1}$ and local loss $l_{n+1} = \frac{1}{2}\|e_{n+1}\|^2$. The LMS rule does not solve $\text{argmin}_W F$ but only approximates it. The next rule, which is equivalent to the normalized least mean squared rule (NLMS) [28] is

$$\text{argmin}_{W_n} F = W_n + \Delta W_n = W_n + \|f(\hat{h}_n)\|^{-2} e_{n+1} f(\hat{h}_n)^T. \quad (6)$$

The NLMS rule performs a gradient update but with an inverse squared ℓ_2 norm of the pre-synaptic activities as its step size instead of a hyper-parameter. The NLMS rule is the minimum-norm solution to $argmin_W F$ in the case where we use mini-batch size 1 (see proposition B.1). The LMS rule closely approximates the NLMS update in the sense the two are proportional. Comparing these rules to the update rule for G-BP (equation 3) we see the main difference is the pre-synaptic term used in the rule: BP uses FF activity h_n, while IL uses optimized/target activity $\hat{h}_n$. This difference leads to significantly different stability properties between the algorithms, as we note below.

Algorithm 1: Generalized BP

```
begin
    // Feedforward Pass
    $h_0 \leftarrow x^{(b)}$
    for $n = 0$ to $N-1$ do
        $h_{n+1} \leftarrow W_n f(h_n)$
    end
    $\hat{h}_N \leftarrow y^{(b)}$
    // Compute Local Targets
    for $n = 1$ to $N$ do
        $\hat{h}_n \approx h_n - \frac{\partial L}{\partial h_n}$
    end
    // Update Weight Matrices
    Eqn. 3
end
```

Algorithm 2: Generalized IL

```
begin
    // Feedforward Pass
    $\hat{h}_0 \leftarrow x^{(b)}$
    for $n = 0$ to $N-1$ do
        $p_{n+1}, \hat{h}_{n+1} \leftarrow W_n f(h_n)$
    end
    // Compute Local Targets
    $(\hat{h}_1, ...\hat{h}_N) \approx \text{argmin}_{\hat{h}} F$
    // Update Weight Matrices
    $(W_0, ...W_{N-1}) \approx \text{argmin}_W F$, Eqn. 4,5
end
```

4 Theoretical Results

4.1 Implicit Gradient Descent

Let the set of MLP weight parameters $\theta^{(b)} = [W_0^{(b)}, ...W_{N-1}^{(b)}]$, where the input data is $x^{(b)}$, the output target is $y^{(b)}$, and b is the current training iteration. The explicit SGD update uses the loss gradient produced by the current parameters:

$$\theta^{(b+1)} = \theta^{(b)} - \alpha \frac{\partial L(\theta^{(b)}, x^{(b)}, y^{(b)})}{\partial \theta^b}, \tag{7}$$

where α is step size and L is the loss measure. This update is explicit because the gradient can be readily computed given $\theta^{(b)}, x^{(b)}, y^{(b)}$. The *implicit* SGD update uses the loss gradient of the parameters at the next training iteration:

$$\theta^{(b+1)} = \theta^{(b)} - \alpha \frac{\partial L(\theta^{(b+1)}, x^{(b)}, y^{(b)})}{\partial \theta^{(b+1)}}. \tag{8}$$

This is an implicit update because $\theta^{(b+1)}$ shows up on both sides of the equation. Unlike explicit SGD, $\theta^{(b+1)}$ cannot be readily computed using available quantities. However, the implicit gradient update is equivalent to the solution of the following optimization problem (see appendix equation 14):

$$\theta^{(b+1)} = \text{argmin}_\theta \left(L(\theta, x^{(b)}, y^{(b)}) + \frac{1}{2\alpha} \|\theta - \theta^{(b)}\|^2 \right). \tag{9}$$

This update is equivalent to the proximal operator/update [30]. This proximal update changes parameters $\theta^{(b)}$ in a way that minimizes the loss L and the magnitude of the update, which helps keep $\theta^{(b+1)}$ in the proximity of $\theta^{(b)}$. (For more details on the proximal update and its relation to implicit SGD see appendix B.1).

4.2 Main Results

We now present our main result, which is to show G-IL is equivalent to implicit SGD in certain limits that are approximated well in practice. We focus on the case where single data-points (mini-batch size 1) are presented each training iteration. This scenario is more biologically realistic than mini-batching and is the case where IL best approximates implicit SGD. We discuss the case of mini-batching in appendix B.5. We first define a kind of IL algorithm that minimizes the proximal loss (equation 9) w.r.t. neuron activities and show G-IL approaches this algorithm in certain limits.

Definition 4.1. *Proximal Inference Learning (IL-prox). An algorithm identical to G-IL (algorithm 2), except activities are optimized according to* $\text{argmin}_{\hat{h}}\, prox = \text{argmin}_{\hat{h}}(L(\theta^*) + \frac{1}{2\alpha}\|\theta^* - \theta^{(b)}\|^2)$ *and the NLMS rule is used to update weights.*

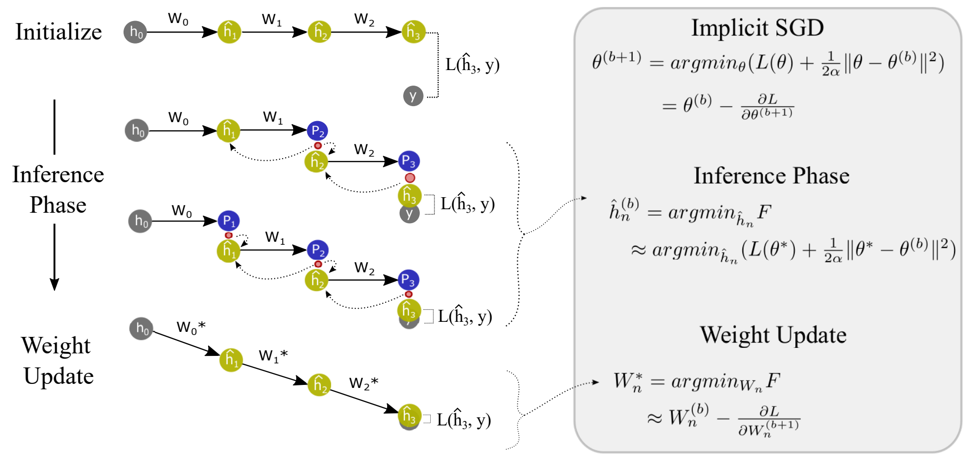
Initialize
W_0
W_1
W_2
$L(\hat{h}_3, y)$
Inference Phase
$W_0{*}$
$W_1{*}$
$W_2{*}$
Weight Update
Implicit SGD
$\theta^{(b+1)} = argmin_{\theta}(L(\theta) + \frac{1}{2\alpha}\|\theta - \theta^{(b)}\|^2)$
$= \theta^{(b)} - \frac{\partial L}{\partial \theta^{(b+1)}}$
Inference Phase
$\hat{h}_n^{(b)} = argmin_{\hat{h}_n} F$
$\approx argmin_{\hat{h}_n}(L(\theta^*) + \frac{1}{2\alpha}\|\theta^* - \theta^{(b)}\|^2)$
Weight Update
$W_n^* = argmin_{W_n} F$
$\approx W_n^{(b)} - \frac{\partial L}{\partial W_n^{(b+1)}}$

Figure 1: A depiction of IL. Target activities $\hat{h}$ are initialized to FF activities h then updated to minimize energy F during the inference phase. Afterward, weights are updated to further minimize F. Weight updates 'align' with $\hat{h}$ activities so $\hat{h}$ become the new FF activities given the same data point. $\hat{h}$ are activities that will both improve the loss L and minimize $\|\Delta\theta\|^2$. This process is equivalent to minimizing the proximal objective and approximates implicit SGD.

Here b is the current training iteration and θ^* are the parameters updated with the NLMS rule. IL-prox is the same as G-IL except during the inference phase, it minimizes the proximal loss w.r.t. activities $\hat{h}$ instead of the energy F. Intuitively, minimizing the proximal loss w.r.t. activities will result in $\hat{h}$ that yields a small loss after weights are updated and small weight update norms. Weight update norm $\frac{1}{2}\|\theta^* - \theta^{(b)}\|^2$ is known during the inference phase because the NLMS learning rule is used and known explicitly:

$$\frac{1}{2}\|\theta^* - \theta^{(b)}\|^2 = \frac{1}{2}\sum_{n}^{N}\|\Delta W_n\|^2 = \frac{1}{2}\sum_{n}^{N}\|\alpha_n e_{n+1} f(\hat{h}_n)^T\|^2,$$

where $\alpha_n = \|f(\hat{h}_n)\|^{-2}$. Thus, it can be minimized w.r.t. $\hat{h}_n$. As we explain in the appendix (see lemma B.1 and proposition B.1), the loss $L(\theta^*)$ is also explicitly known during the inference phase and can be optimized w.r.t. $\hat{h}_n$.

Now, let α be a 'global' learning rate hyper-parameter, and α_n be layer-wise learning rates used in the actual weight update ΔW_n.

Theorem 4.1. *Let $\alpha_n = \|f(\hat{h}_n)\|^{-2}$ and assume mini-batch size 1. In the limit where $\gamma_n^{decay} \rightarrow \|e_{n+1}\|^2(1 - \frac{2}{\alpha_n^6})$, $\gamma_N \rightarrow \frac{\alpha_{N-1}}{\alpha}$, and $\gamma_n \rightarrow \alpha_{n-1}^{-1}$ for all $n < N$, it is the case that* $\text{argmin}_{\hat{h}} F = \text{argmin}_{\hat{h}} L(\theta^*) + \frac{1}{2\alpha}\|\Delta\theta\|^2$. *Hence, in these limits G-IL is equivalent to IL-prox.*

The proof can be found in supplementary materials theorem B.1. This theorem states that when using the NLMS rule, G-IL is increasingly similar to IL-prox, as the γ weighting terms in the free energy (equation 4) approach the scalar values specified in the above theorem. In these limits, $\text{argmin}_{\hat{h}}\, prox = \text{argmin}_{\hat{h}} F$ and G-IL is equivalent to IL-prox. The intuitive explanation of this theorem goes as follows: minimizing the proximal loss (equation 9) requires minimizing L w.r.t. θ^* while also minimizing the update norm $\frac{1}{2\alpha}\|\theta^* - \theta^{(b)}\|^2$. The inference phase of IL minimizes L w.r.t. θ^* by initializing output layer activity $\hat{h}_N$ to FF output h_N then shifting it toward global target y. Upon a weight update this will yield a θ^* that produces a smaller loss L. See figure 4.1 for visualization. The magnitude of a weight update $\|e_{n+1}f(\hat{h}_n)^T\|^2$ intuitively depends on the magnitude of local prediction error e_{n+1}. F is a sum of the magnitudes of errors (see equation 4), so minimizing F minimizes the magnitudes of errors and consequently weight update norms $\frac{1}{2\alpha}\|\theta^* - \theta^{(b)}\|^2$. Importantly, the γ settings under which theorem 4.1 holds can be computed exactly in practice and are of reasonable magnitude that is easily approximated in practice (e.g., they do not approach ∞ or 0 and typically approach positive scalars). This is opposed to the limits under which

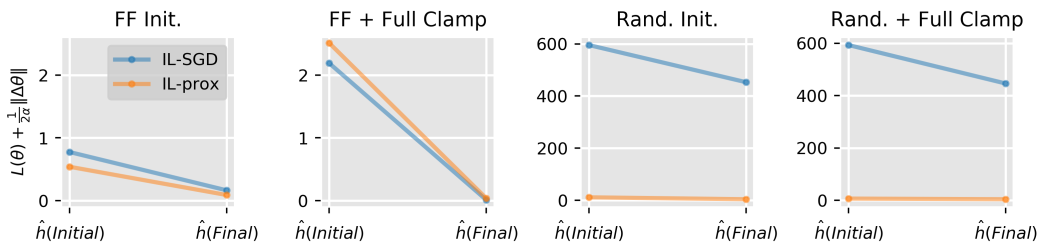

Figure 2: We measure the proximal objective during the inference phase, where F is minimized w.r.t. activities $\hat{h}$, in a two hidden layer MLP with ReLU activations trained on MNIST. We test under four different initializations of $\hat{h}$. First, all $\hat{h}$ are initialized to h (FF Init.). Second, hidden layer $\hat{h}$ are initialized to h, while output layer $\hat{h}_N = y$ (FF + Full Clamp). For both of these initializations, the proximal loss is reduced to near zero for both the IL-prox and IL-SGD variants of IL (defined below). We next initialized $\hat{h}$ randomly (Rand Init. and Rand + Full Clamp). Both networks reduce the proximal loss, but nowhere near the global minima, which may help explain why the $\hat{h}$ needs to be initialized to FF activities when F is minimized with SGD. These results support theorem 4.1, which states that minimizing F w.r.t. $\hat{h}$ also minimizes the proximal objective.

IL approximates BP/explicit SGD, which is the limit where $\hat{h}_n \to h_n$ and thus $\Delta W_n \to 0$ [43], a condition that does not generally occur in practice.

Theorem 4.2. *Let $\theta^{(b)}$ be a set of MLP parameters at training iteration b. Let $\theta^{(b+1)}_{prox} = argmin_\theta L(\theta) + \frac{1}{2\alpha}\|\theta - \theta^{(b)}\|^2$. Let $\theta^{(b+1)}_{IL-prox}$ be the parameters updated by IL-prox (see def. 4.1) and $\theta^{(b+1)}_{IL}$ the parameters updated by G-IL under γ values in theorem 4.1. Assume mini-batch size 1. Under this assumption, it is the case $\theta^{(b+1)}_{prox} = \theta^{(b+1)}_{IL-prox} = \theta^{(b+1)}_{IL}$.*

This theorem states that the IL-prox update is equivalent to the proximal update (equation 9), and thus implicit SGD. It follows that, under the γ settings in theorem 4.1 the G-IL update is also equivalent to the proximal update/implicit gradient update.

Further theoretical results are developed in the appendix. A novel closed form description of IL targets is developed in C.5. The closed form description shows local targets approximate a regularized Gauss-Newton update on FF activities (see section C). We call this closed form approximation IL-GN. We study the stability of IL-GN when using the LMS learning rule, and compare to an analogous BP network, which uses Gauss-Newton updates (BP-GN). We show that IL-GN weight updates push output layer activities down their loss gradients *for any positive learning rate*, i.e., for any positive learning rate $\hat{h}^{(b)}_N - \hat{h}^{(b+1)}_N = j\frac{\partial L}{\partial \hat{h}^{(b)}_N}$ where j is a positive scalar (theorem D.1). We show BP-GN only has this property for a small range of learning rates (theorem D.2). We develop a partial explanation of why IL is more stable than BP that is based on differences in their learning rule (for details see section E). These results suggest that the differences in the way IL and BP compute and use local targets in their weight updates can explain much of their differences in stability across learning rate. The IL learning rule and target computations are advantageous over that of G-BP in this sense.

5 Experiments

In this section, we compare the performance of BP based algorithms to IL algorithms. BP algorithms compute and use explicit gradients to update weights, while IL algorithms compute and use approximate implicit gradients to perform updates. We test BP models that either perform simple gradient descent (**BP-SGD**) or use Adam optimizers (**BP-Adam**). **IL-SGD** is a simple implementation of IL that is similar to implementations of [32, 43]. It uses the LMS rule (local error gradient update) to update weights and optimizes $\hat{h}$ using SGD to near convergence (25 update steps). **IL-Prox** is our novel algorithm that is based on definition 4.1. IL-prox uses the NLMS rule to update weights. Following theorem 4.1, the learning rate is only used to determine how much the output layer activities $\hat{h}_N$ are pushed toward the target y. As $\alpha \to \infty$, $\hat{h}_N \to y$ and as $\alpha \to 0$, $\hat{h}_N \to h_N$. IL-Prox

Reconstruction BCE (mean±std.)					
Data	BP-SGD	BP-Adam	IL-SGD	IL-prox	IL-prox fast
F-MNIST	.303	**.282**	.306	**.286**	**.286**
CIFAR-10	.631	.622	.652	**.611**	**.611**

Table 3: Best test reconstruction loss after one epoch of training, mini-batch size 1. BCE was averaged across pixels and data-points. Top two scores highlighted in bold. All standard deviations $\leq .002$.

optimizes $\hat{h}$ using SGD to near convergence (25 update steps). **IL-Prox Fast** is the same as the IL-Prox algorithm except it truncates the optimization of $\hat{h}$ to only 12 iterations. **IL-Adam** computes the weight gradients using IL-SGD, then uses Adam optimizers to update weights. **IL-prox Adam** computes the normalized weight gradient using IL-prox and uses Adam optimizers to update weights. All simulations average results over at least 5 seeds of each model type. All models use ReLU activations at hidden layers. Softmax is used at output layer for classification, while sigmoid is used on the autoencoder task.

Mini-batch Training Previous work has shown that IL algorithms can achieve the same accuracy as BP on small scale data sets (e.g., MNIST and Fashion-MNIST), in the case where Adam optimizers and medium sized mini-batches (e.g, 64) are used ([43, 2]. Here we train our BP and IL algorithms on CIFAR-10 with mini-batch size 64. Results are summarized in table 2. Consistent with previous results, IL algorithms, including our novel IL-prox, achieve near equal accuracy as BP-SGD and BP-Adam when using Adam optimizers. However, unlike previous studies, we also test IL-SGD and IL-prox algorithms without Adam, and find they do not perform as well as BP algorithms in this scenario. IL algorithms without Adam tend to converge more quickly but to shallower local minima. Adam, with its momentum and adaptive learning rate, may help prevent IL algorithms from getting caught in shallow local minima (see discussion).

CIFAR-10 Test Accuracy (mean±std.)		
Model	m-batch size=1	m-batch size=64
BP-SGD	36.834(±.478)	**57.044** (±.144)
IL-SGD	**43.894** (±.371)	46.924(±.248)
IL-prox	42.060(±.412)	49.844(±.359)
IL-prox Fast	**42.984** (±.530)	46.094(±.228)
BP-Adam	42.102(±.770)	**56.466** (±.228)
IL-Adam	40.724(±.160)	54.066(±.190)
IL-prox + Adam	42.802(±.390)	55.572(±.172)

Table 2: Best test accuracy on CIFAR-10 supervised task with mini-batch size 1 after 50,000 iterations (1 epoch) and with mini-batch size 64 with 77000 iterations.

Online Learning We find IL algorithms tend to converge significantly quicker and to more similar loss/accuracies as BP in a more biologically realistic, online learning scenario, where a single data-point is presented each training iteration and each data point is seen only once. This is especially apparent with CIFAR-10, shown in tables 2 and 3 and figure 3. IL-SGD and IL-prox algorithms perform similar to and even slightly better than BP-Adam, despite not using momentum or stored, parameter-wise adaptive learning rates like Adam. As far as we know, this is a novel result. Less significant speedups were also seen on MNIST and Fashion-MNIST classification tasks (see figure 6). On autoencoder tasks, IL-prox algorithms, in particular, decreased loss significantly quicker than BP-SGD and even slightly quicker than BP-Adam (figure 3).

Stability Test: CIFAR-10 Test Accuracy						
Model	lr=.01	lr=.1	lr=1	lr=2.5	lr=10	lr=100
BP-SGD	35.48(±1.65)	–	–	–	–	–
IL-SGD	41.66(±1.65)	36.268(±.271)	–	–	–	–
BP-prox	38.39(±2.91)	23.978(±1.57)	12.89(±1.1)	12.10(±1.61)	12.21(±1.61)	–
IL-prox	34.97(±.67)	33.47(±2.56)	37.38(±1.46)	37.12(±1.81)	37.01(±1.01)	37.64(±.74)

Table 4: Accuracies after 50,000 training iterations on CIFAR-10, mini-batch size 1. Fully connected networks with layer sizes 3072-3x1024-10.

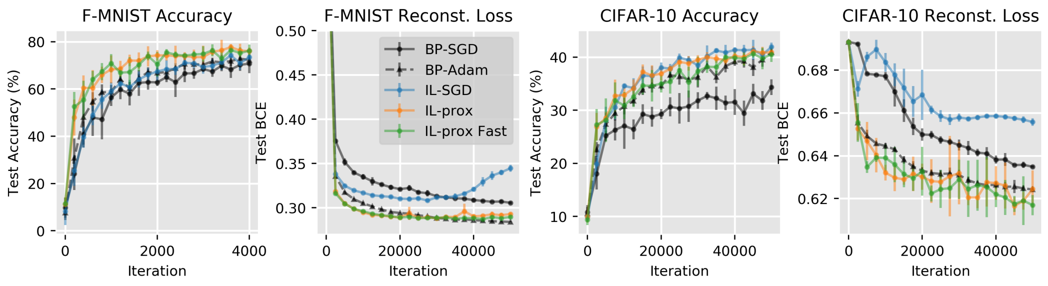

Figure 3: Training runs for mini-batch size 1 (online) scenario, for classification and autoencoder tasks. MLP size 784-2x500-10 and autoencoder dimension 784-256-100-256-784 are trained on F-MNIST for 50000+ iterations. MLP size 3072-3x1024-10 and autoencoder 3072-1024-500-100-500-1024-3072 are trained on CIFAR-10 for 50000+ iterations. The first 4000 training iterations shown for F-MNIST on left. Full training run are shown for other simulations in right three plots.

Learning Under Data Constraints We also find that IL algorithms can converge to better accuracy than BP in highly data constrained scenarios. We trained IL algorithms using on only 10, 100, or 500 data points from each category on F-MNIST and CIFAR-10, with mini-batch size 100. IL algorithms achieved better accuracies than BP-SGD in the 10 datapoint scenario on F-MNIST and CIFAR, and better accuracy on the 100 data point scenario on CIFAR, while performances evened out in the 500 data point scenario (see appendix table 8 for details). IL-prox also tended to converge significantly quicker than other algorithms on Cifar-10 (figure 4).

Stability Test Explicit SGD is well-known to be highly sensitive to learning rate. Implicit SGD, on the other hand, is highly insensitive to learning rate and is unconditionally stable, i.e., it is able to discount the loss in a stable manner for nearly any positive learning rate [41]. We compare performance of BP-SGD, IL-SGD, and IL-prox on MNIST (table 5) and CIFAR-10 (table 4) classification task across different learning rates. As a control, we train a hybrid of BP and IL-prox (BP-prox). BP-prox, like IL-prox, uses the learning rate only to adjust the target at the output layer and uses the NLMS rule to update weights. Unlike IL-prox, BP-prox computes local targets using the G-BP algorithm rather than the IL algorithm. Results can be found in table 5 and 4. Blank entries are those with accuracies below 12%. IL-SGD is more stable than BP-SGD, and IL-prox is more stable than BP-prox. This suggests that the way IL computes and uses targets contributes to stability. Additionally, IL-prox and BP-prox algorithms are more stable than SGD algorithms, showing that using the normalized gradient to update weights and α to adjust the output layer target (rather than scale weight updates) also contributes to stability.

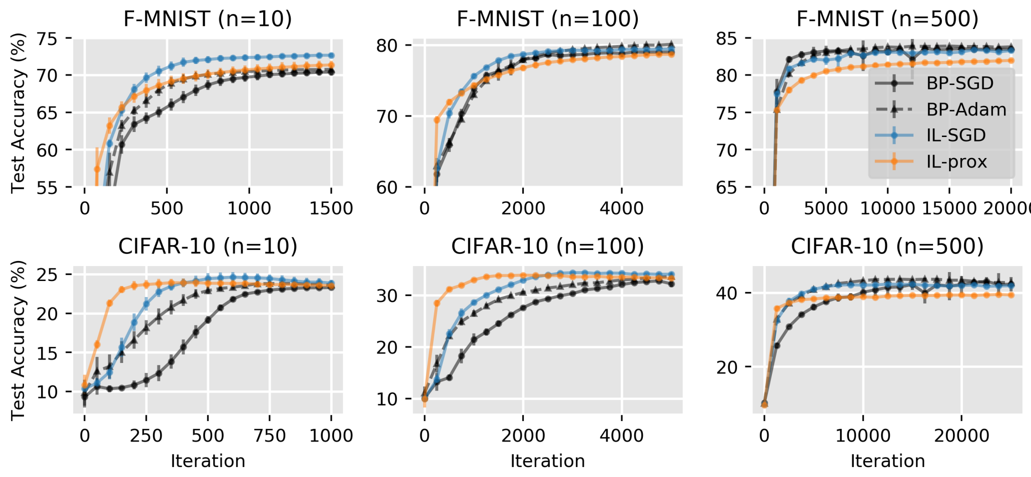

Figure 4: Training runs for a subset of the algorithms on the data constrained scenarios. Each model trained on n data points from each category.

6 Limitations

Our theoretical results show that IL algorithms closely approximate implicit SGD in the case where only a single data-point (i.e., mini-batch size 1) is presented each training iteration. This approximation is looser in the case where large mini-batches are used (see appendix B.5). This is no problem from a biological point of view, since the brain does not train with large mini-batches. Additionally, when Adam optimizers are used IL still performs similarly to BP. However, it may still be desirable to further study the optimization properties of IL with mini-batches to improve performance on machine learning tasks where mini-batches are typically used.

7 Discussion

In this paper, we found that IL closely approximates implicit SGD, which is distinct from the explicit SGD performed by BP. This theoretical result, along with results that help to characterize how IL and BP behave differently, suggest a hypothesis, which as far was we know is novel: learning in the brain is more similar to implicit gradient descent than explicit gradient descent. This hypothesis would imply the learning characteristics of the brain behave differently from BP and explicit SGD, more generally. More empirical work will be needed to test this theory. However, recent evidence directly supports the idea the brain does something similar to IL [38]. Evidence that neural circuits perform predictive coding provides further, though more indirect, evidence for the idea.

We also found that IL often converged more quickly than BP and achieved similar test losses and accuracies in online and data constrained scenarios. However, in large mini-batch scenarios without data constraints and without the use of Adam optimizers, IL sometimes converged to worse accuracies than BP in classification tasks. With the use of Adam, IL's performance more closely matched that of BP. One possible explanation of this empirical finding may be related to the fact that IL, and implicit SGD more generally, tend to take smaller steps over parameters and a more direct path toward local minima than BP (e.g., see figures 7 and table 9). More direct paths to minima often lead to faster convergence. Smaller update magnitudes may also act as a kind of built-in regularization that aids in difficult, noisy learning scenarios, such as small mini-batch and data constrained scenarios. In cases where regularization is less advantageous (e.g., no data constraints and large mini-batches), IL's small update magnitudes may provide less of an advantage, and its more direct optimization paths may push parameters into shallow, nearby minima. Adam optimizers, with their momentum and adaptive learning rates, may help push parameters out of shallow minima. More work will be needed to fully characterize and explain these behavioral differences, but we believe the work done here provides a good formal basis for doing so.

These results further suggest IL may have advantages over BP in more biologically realistic machine learning tasks, such as lifelong, online learning scenarios, where streams of data are received by the neural network, a single datapoint at a time, and the model must adapt quickly to changes in the environment and minimize the magnitude of weight updates to prevent catastrophic forgetting. This possibility is especially interesting given that, unlike BP, IL's weight updates are both local in space and time and may thus be more compatible with energy efficient neuromorphic hardware, which is well suited for autonomous, embedded systems that run on batteries and receive data streams through sensors, similar to the brain.

8 Broader Impacts

This work is largely theoretical. As such, we do not see any potential negative societal impacts.

9 Acknowledgements

This work was partially supported by the National Science Foundation (NSF) under grant 1652159.

References

[1] Rick A Adams, Stewart Shipp, and Karl J Friston. Predictions not commands: active inference in the motor system. *Brain Structure and Function*, 218(3):611–643, 2013.

[2] Nicholas Alonso and Emre Neftci. Tightening the biological constraints on gradient-based predictive coding. In *International Conference on Neuromorphic Systems 2021*, pages 1–9, 2021.

[3] Sergey Bartunov, Adam Santoro, Blake A Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *arXiv preprint arXiv:1807.04587*, 2018.

[4] Andre M Bastos, W Martin Usrey, Rick A Adams, George R Mangun, Pascal Fries, and Karl J Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

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

[6] Yoshua Bengio. Deriving differential target propagation from iterating approximate inverses. *arXiv preprint arXiv:2007.15139*, 2020.

[7] Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211, 2017.

[8] Hyeonwoo Cho, Chang Woo Lee, and Sang Woo Kim. Derivation of a new normalized least mean squares algorithm with modified minimization criterion. *Signal Processing*, 89(4):692–695, 2009.

[9] Anna Choromanska, Benjamin Cowen, Sadhana Kumaravel, Ronny Luss, Mattia Rigotti, Irina Rish, Paolo Diachille, Viatcheslav Gurev, Brian Kingsbury, Ravi Tejwani, et al. Beyond backprop: Online alternating minimization with auxiliary variables. In *International Conference on Machine Learning*, pages 1193–1202. PMLR, 2019.

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

[11] Maxence Ernoult, Fabrice Normandin, Abhinav Moudgil, Sean Spinney, Eugene Belilovsky, Irina Rish, Blake Richards, and Yoshua Bengio. Towards scaling difference target propagation by learning backprop targets. *arXiv preprint arXiv:2201.13415*, 2022.

[12] Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *arXiv preprint arXiv:1706.04638*, 2017.

[13] Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

[14] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[15] Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*, 364(1521):1211–1221, 2009.

[16] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

[17] Tim Tsz-Kit Lau, Jinshan Zeng, Baoyuan Wu, and Yuan Yao. A proximal block coordinate descent algorithm for deep neural network training. *arXiv preprint arXiv:1803.09082*, 2018.

[18] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[19] 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.

[20] 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(1):1–10, 2016.

[21] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[22] Joseph Marino. Predictive coding, variational autoencoders, and biological connections. *Neural Computation*, 34(1):1–44, 2022.

[23] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1781–1790, 2017.

[24] Alexander Meulemans, Francesco S Carzaniga, Johan AK Suykens, João Sacramento, and Benjamin F Grewe. A theoretical framework for target propagation. *arXiv preprint arXiv:2006.14331*, 2020.

[25] Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021.

[26] Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*, 2020.

[27] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[28] Jin-Ichi Nagumo and Atsuhiko Noda. A learning method for system identification. *IEEE Transactions on Automatic Control*, 12(3):282–287, 1967.

[29] 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.

[30] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[31] Linbo Qiao, Tao Sun, Hengyue Pan, and Dongsheng Li. Inertial proximal deep learning alternating minimization for efficient neutral network training. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3895–3899. IEEE, 2021.

[32] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[33] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34, 1995.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[35] Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *arXiv preprint arXiv:2201.13180*, 2022.

[36] Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34, 2021.

[37] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation? exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566, 2020.

[38] Yuhang Song, Beren Gray Millidge, Tommaso Salvatori, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Inferring neural activity before plasticity: A foundation for learning beyond backpropagation. *bioRxiv*, 2022.

[39] Michael W Spratling. Predictive coding as a model of biased competition in visual attention. *Vision research*, 48(12):1391–1408, 2008.

[40] Panagiotis Toulis, Edoardo Airoldi, and Jason Rennie. Statistical analysis of stochastic gradient methods for generalized linear models. In *International Conference on Machine Learning*, pages 667–675. PMLR, 2014.

[41] Panos Toulis and Edoardo M Airoldi. Implicit stochastic gradient descent for principled estimation with large datasets. *ArXiv e-prints*, 2014.

[42] Panos Toulis and Edoardo M Airoldi. Asymptotic and finite-sample properties of estimators based on stochastic gradients. *The Annals of Statistics*, 45(4):1694–1727, 2017.

[43] James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

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

[45] Dong Yang and Jian Sun. Proximal dehaze-net: A prior learning-based deep network for single image dehazing. In *Proceedings of the european conference on computer vision (ECCV)*, pages 702–717, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 8
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] For all theorems, propositions, and lemmas assumptions were stated in the statement.
   (b) Did you include complete proofs of all theoretical results? [Yes] All proofs are in appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] This information is in supplementary material and in link to code
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] This information is reported in supplementary material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Standard deviations over multiple seeds are shown in tables and in plots using error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Is included in supplementary material
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]