A Theoretical Framework for Inference Learning

Anonymous Author(s)
Affiliation
Address
email

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 has close connections to neurobiological models of cortical function and has achieved equal performance to BP on supervised learning 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 also demonstrate IL achieves quicker convergence when trained with small mini-batches while matching the performance of BP for large mini-batches.

1 Introduction

Backpropagation (BP) [37], the most successful and ubiquitously used learning algorithm in deep learning, has long been criticized as being incompatible with known neurobiology [10, 23]. This criticism has inspired researchers to search for alternatives to BP that may better fit biological constraints. One alternative to BP is the inference learning algorithm (IL) [36, 46, 42]. IL is an energy based algorithm [41, 47], which first minimizes an energy function, known as free energy, w.r.t. neuron activities, then at convergence updates weights to further minimize energy. Interest in IL has grown in recent years because IL arguably better fits biological constraints than BP, while matching BP's performance on many tasks. For example, BP is sometimes said to compute weight updates using information non-local to the synapse, whereas synapses in the brain depend on local information having to do with the activity of pre-synaptic and post-synaptic neurons [10, 23]. IL, on the other hand, uses local learning rules that are similar to Hebbian models of synaptic plasticity in the brain (e.g., [46, 47]). Another criticism is that error feedback propagated by BP does not alter feed forward (FF) activities, which is hard to reconcile with the highly interconnected circuitry of FF and feedback signals in the brain [22, 25]. Feedback used in IL, on the other hand, necessarily alters FF activity in order to minimize free energy. Further, IL has long been used to train predictive coding (PC) models of circuits in the cortex (e.g. [36, 13, 43, 4, 1, 8]). PC and IL have even been proposed as canonical computations used throughout the brain [15, 14, 4]. More recently, IL has achieved comparable accuracy to BP on supervised learning tasks with small and medium scale data-sets (e.g, [46, 2, 39], see also below). IL has also been used to train PC networks to perform auto-associative memory tasks, and on some tests has achieved a higher recall accuracy than both modern Hopfield networks [17, 35] and denoising auto-encoders trained with BP [40, 39].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

IL's biological connections motivates further analysis into its optimization properties given that such an analysis may lead to a better understanding of the principles behind synaptic plasticity and optimization in the brain. Additionally, IL's competitive performance with BP further motivates an investigation into the question of *why* IL is competitive with BP. Similar questions, such as *How does IL behave differently than BP? What advantages/disadvantages does IL have as compared to BP?* can also aid in understanding which applications, if any, are better suited for IL rather than BP and vice versa. The optimization properties of IL and their relations to BP, however, are still not fully understood. This gap in our understanding is the main motivation for this paper.

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). Implicit SGD is distinct from explicit SGD (the standard form of SGD used in machine learning and the sort executed by BP). 3) We provide theoretical guarantees concerning IL's stability and a connection to Gauss-Newton optimization and BP. 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 streaming data and small mini-batches. These results collectively suggest that, not only does IL better fit biological constraints but is mathematically justified and has performance advantages over BP in more biologically realistic training scenarios.

2 Related Works

Whittington and Bogacz [46] proved that parameter updates performed by the IL algorithm approach those of BP as the global loss approaches zero and optimized activities approach feed-forward activities. However, this same proof implies that non-zero IL updates essentially never equal those of BP. Additionally, this analysis leaves open the question of how IL is able to minimize the loss in a stable manner early in training when the loss is large and IL updates diverge significantly from those of BP. Other work has shown IL is formally related to variational expectation maximization [28, 24], which is a learning algorithm with convergence guarantees [30]. This analysis still 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 [29, 42]. These variants significantly change the original IL algorithm, and some the alterations seem hard to reconcile with neurobiology [29, 42]. 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 show the variable settings under which the standard implementation of IL better approximates implicit SGD.

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 3 below). However, [9] do not connect their algorithm to implicit SGD. A more recent variant of this algorithm by Qiao et al. [34] show some formal links to proximal operators, an optimization process related to implicit SGD [33] (see appendix A.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. [19] developed an algorithm that combined and algorithm known as 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 [26, 49]. 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.

Finally, IL has similarities 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 [21] utilize approximate Gauss-Newton updates on activities to create local targets [27, 6], while other algorithms compute targets using gradient updates on activities

[20, 32, 3, 18]. These algorithms have important differences from IL, e.g., they use a 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 and Notation

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 FF network (MLP) is summarized in table 1. To simplify notation, we assume 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 [38] is a 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 [20], [3], [32]. Specifically, the local target at layer n, $\hat{h}_n$, is computed as follows

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

where local loss $l_{n+1} = \frac{1}{2}\|\hat{h}_{n+1} - h_{n+1}\|^2 = \frac{1}{2}\|e_{n+1}\|^2$. The target at the output layer $\hat{h}_N = h_N - \frac{\partial L}{\partial h_N}$. Notice that $e_{n+1} = \hat{h}_{n+1} - h_{n+1} = h_{n+1} + \frac{\partial l_{n+2}}{\partial h_{n+1}} - h_{n+1} = \frac{\partial l_{n+2}}{\partial h_{n+1}} = -\frac{\partial L}{\partial h_{n+1}}$. As noted in the above equation, this local target update is equivalent to subtracting the global loss gradient from the FF activity h_n (for proof see appendix [3]). To update weight W_n one uses the gradient of the local loss:

$$\Delta W_n = -\alpha \frac{\partial l_{n+1}}{\partial W_n} = \alpha \frac{\partial L}{\partial W_n} = -\alpha e_{n+1} f(h_n)^T \tag{2}$$

This update uses only local information and is equivalent to subtracting the global loss gradient from the weights (see [3]). This local target formulation of SGD is similar to other target propagation (TP) [5, 21] algorithms, which use a similar learning rule but compute targets slightly differently. For example, a well-known variant of TP is difference target propagation (DTP) [21], which computes local targets by approximating a Gauss-Newton update on activities rather than gradient updates [27, 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 $e_N = y - h_N$. Gauss-Newton updates approximate (are within $90°$) of SGD updates [27], and recent attempts to improve DTP generally do so by making DTP updates more similar to SGD updates (e.g. [27, 11]). Additionally, variants of BP algorithms like random feedback alignment [22] and sign symmetric feedback alignment [48] approximate SGD. Since all of these methods approximate the (explicit) SGD update over weights, 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:

$$F = L(y, \hat{h}_N) + \sum_{n=1}^{N} \gamma_n \frac{1}{2} \|\hat{h}_n - p_n\|^2 + \sum_{n=1}^{N-1} \gamma_n^{decay} \frac{1}{2} \|\hat{h}_n\|^2, \tag{3}$$

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 approximated, 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 and either the LMS rule (below) or Adam optimizers to update weights (e.g. [36, 46]). 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:

$$\text{argmin}_{W_n} F \approx W_n + \Delta W_n = W_n - \alpha \frac{\partial l_{n+1}}{\partial W_n} = W_n + \alpha e_{n+1} f(\hat{h}_n)^T, \tag{4}$$

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) [31] is

$$\text{argmin}_{W_n} F = W_n + \Delta W_n = W_n + \|f(\hat{h}_n)\|^{-2} e_{n+1} f(\hat{h}_n)^T. \tag{5}$$

The NLMS rule performs a gradient update but with an inverse squared ℓ_2 norm of the pre-synaptic activities as its learning rate 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 A.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 2) 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. 2

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

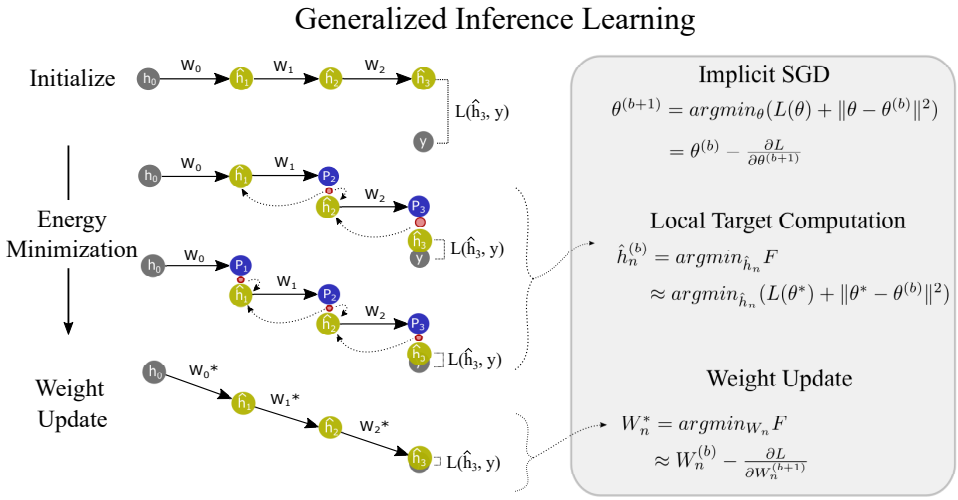

Figure 1: A depiction of IL. Target activities $\hat{h}$ are initialized to FF activities h then updated to minimize energy F. Afterward, weights are updated to further minimize F. Weight updates 'align' with $\hat{h}$ activities so $\hat{h}$ become FF activities given the same data point. $\hat{h}$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

4 Theoretical Results

4.1 Implicit Gradient Descent

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

$$\theta^{(b+1)} = \theta^{(b)} - \alpha \frac{\partial L(\theta^{(b)}, x^{(b)}, y^{(b)})}{\partial \theta^b}, \tag{6}$$

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:

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

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 is equivalent to the solution of the following optimization problem (see appendix equation 12):

$$\theta^{(b+1)} = \text{argmin}_\theta(L(\theta, x^{(b)}, y^{(b)}) + \frac{1}{2\alpha}\|\theta - \theta^{(b)}\|^2). \tag{8}$$

This update is equivalent to the proximal operator/update [33]. 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 A.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 A.5. We first define a kind of IL algorithm that minimizes the proximal loss (equation 8) 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.*

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 A.1 and proposition A.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* $\gamma_n^{decay} \to \|e_{n+1}\|^2(1 + \frac{-2}{\alpha_n^6})$, $\gamma_N \to \frac{\alpha_{N-1}}{\alpha}$, *and* $\gamma_n \to \alpha_{n-1}^{-1}$ *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 A.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 3) 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 11) 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 $\frac{1}{2\alpha}\|\alpha e_{n+1} f(\hat{h}_n)^T\|^2$ intuitively depends on the magnitude of local prediction error e_{n+1}. F is a sum of the magnitudes of errors (see equation 3), 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 IL approximates BP/explicit SGD, which is the limit where $\hat{h}_n \to h_n$ and thus $\Delta W_n \to 0$ [46], a limit that does not typically obtain in practice.

Theorem 4.2. *Let* $\theta^{(b)}$ *be a set of MLP parameters at training iteration* b. *Let* $\theta_{prox}^{(b+1)} = argmin_\theta L(\theta) + \frac{1}{2\alpha}\|\theta - \theta^{(b)}\|^2$. *Let* $\theta_{IL-prox}^{(b+1)}$ *be the parameters updated by IL-prox (see def. 4.1) and* $\theta_{IL}^{(b+1)}$ *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* $\theta_{prox}^{(b+1)} = \theta_{IL-prox}^{(b+1)} = \theta_{IL}^{(b+1)}$.

This theorem states that the IL-prox update is equivalent to the proximal update (equation 8), 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 B.5. The closed form description shows local targets approximate a regularized Gauss-Newton update on FF activities (see section B). 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 $\hat{h}_N^{(b)} - \hat{h}_N^{(b+1)} = j\frac{\partial L}{\partial \hat{h}_N^{(b)}}$ where j is a positive scalar (theorem C.1). We show the same is

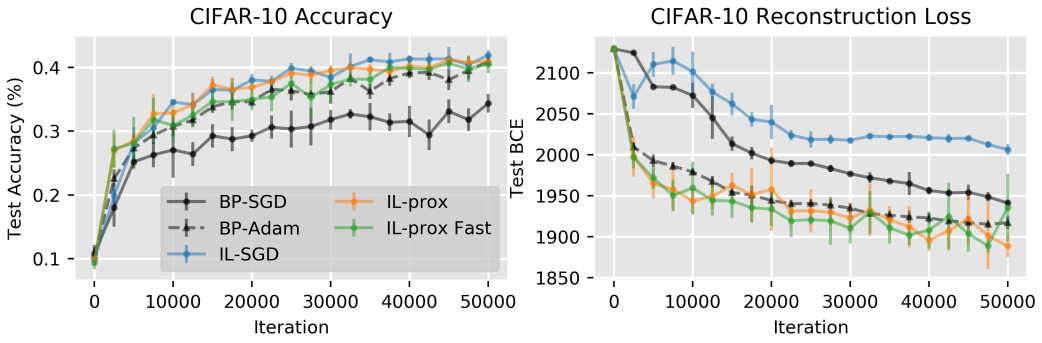

Figure 2: **(Left)** Test accuracies on CIFAR-10 with mini-batch size 1 for one epoch of training. **(Right)** Reconstruction losses for autoencoders on CIFAR-10 for over one epoch of training, mini-batch size 1.

not true of BP-GN, which only has this property for a small range of learning rates (theorem C.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 D). These results make the important point that the way IL computes and uses local targets in its weight update significantly contributes to its stability, while the way G-BP computes and uses local targets in its weight update (which is distinct from IL, see section 2) is largely responsible for its instability. The IL learning rule and method of computing $\hat{h}$ is advantageous over that of G-BP in this sense.

5 Experiments

In this section, we test and compare BP based algorithms to several 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 [36, 46]. 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. As theorem 4.1 shows, to minimize the proximal loss, 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 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. Best test scores from these averages are shown in tables. All models use ReLU activations at hidden layers. Softmax is used at output layer for classification, while sigmoid is used on the autoencoder task.

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 in two conditions. First, mini-batch size 1 trained for 50,000 iterations (1 epoch). Second, is mini-batch size 64 trained for 77,000 iteration ($\approx$ 100 epochs). Fully connected networks with layer sizes 3072-3x1024-10 were used. Top two scores highlighted in bold.

Reconstruction BCE - CIFAR-10 (mean±std.)				
BP-SGD	BP-Adam	IL-SGD	IL-prox	IL-prox fast
1937.884(±1.826)	1912.202(±3.548)	2004.062(±5.543)	**1878.463** (±3.641)	**1875.899** (±3.134)

Table 3: Best test reconstruction loss after one epoch of training on with CIFAR-10 images, mini-batch size 1. BCE was summed across pixels and averaged across data-points. Fully connected network layer sizes 3072-1024-512-100-512-1024-3072. Top two scores highlighted in bold.

Supervised and Self-Supervised Learning We begin by training these models on supervised classification tasks. Results for CIFAR-10 are summarized in table 2. Consistent with previous results [46, 2], in the case where large mini-batches are used, IL algorithms achieve near equal accuracy as BP-SGD and BP-Adam when using Adam optimizers. But IL-SGD and IL-prox algorithms, which do not use Adam optimizers, do not perform as well as BP algorithms in this scenario. However, in the case where they are trained with mini-batch size 1, IL algorithms improve accuracy significantly faster than BP-SGD for the entire first epoch (50,000 iterations) of training. 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. Previous works have not tested IL in case where small mini-batches are used (e.g., [46, 2]). To further test this improvement in optimization speed, we also trained these models on MNIST and fashion-MNIST classification tasks (table 6), and we trained autoencoders on CIFAR-10 images (table 3), with mini-batch size 1. Slight speed-ups in training were seen on MNIST data sets (see figure E.1.1). On the autoencoder task with CIFAR-10, we found again IL-prox algorithms improve loss as quickly and even slightly quicker than BP-Adam (see figure 2).

Stability Test Implicit SGD is unconditionally stable, which, roughly, means that it is able to discount the loss in a stable manner for nearly any positive learning rate [44]. 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 also test 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. This control helps to show the stability of IL-prox is not only due to normalizing the weight updates and softly clamping the output layer. Rather, the way IL computes and uses local targets in its learning rule also contributes to stability. 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. These results provide evidence that the way IL computes and uses targets contributes to stability. Additionally, the 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.

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.

Further Results We computed the update norms $\|\Delta\theta\|$ during training of each model on the CIFAR-10 classification task. Average magnitudes and standard deviations are shown in appendix E. IL algorithms had similar or smaller update magnitudes than BP-SGD on most of their training runs. This suggests IL algorithms do not minimize loss more quickly because their updates are greater in magnitude than BP-SGD. One possibility is that IL updates take a more direct path to local minima than the path of steepest descent. This is consistent with other experiments in which we analyzed the output layer of BP-SGD and IL-SGD. We measured how the FF output layer values h_N changed after updates. IL updates tended to change the output layer values more and in a direction nearer the minimum norm (shortest) path to that target y than BP-SGD updates (figures E.1.3, E.1.3).

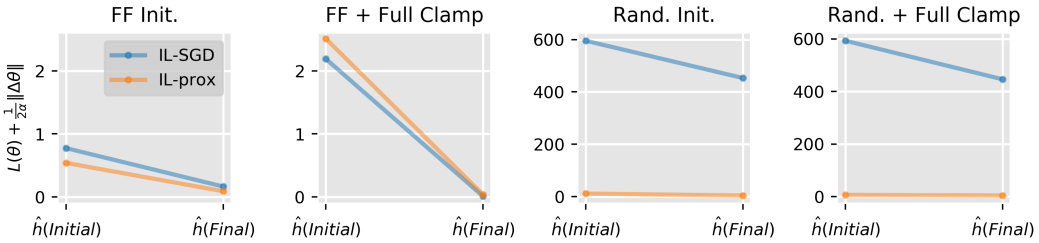

Figure 3: We measure how the proximal loss changes during the minimization of F w.r.t. activities $\hat{h}$ under four different initializations. First, all $\hat{h}$ are initialized to h (FF Init.). Second, hidden layer $\hat{h}$ are initialized to h, while output layer $\hat{h}_n = y$ (FF + Full Clamp). For both of these initialization, the proximal quantity is reduced during inference to near zero. 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 network needs to be initialized to FF activities when F is minimized with SGD. These results generally support theorem 4.1, which states minimizing F minimizes the proximal loss.

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 A.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

IL was originally used to train predictive coding models of cortical circuits [36]. The success of subsequent predictive coding models lead some neuroscientists to propose the hypothesis that predictive coding and IL are canonical computations used throughout the brain (e.g., [13, 14, 4, 16]). In this paper, we provided novel mathematical justification for IL by showing it closely approximates a proper optimization method known as implicit SGD, which is distinct from the explicit SGD executed by BP. This finding, in conjunction with the theory IL and predictive coding are canonical computations in the brain, suggests the interesting possibility that implicit SGD, and not explicit SGD, is the central optimization method utilized by biological neural circuits. We observed two performance advantages that fit with this possibility. First, consistent with our theoretical interpretation, IL algorithms are more stable across learning rates than BP. This property is compatible with the fact that synaptic plasticity in the brain can fluctuate rapidly, e.g. due to neuromodulation, yet maintains stable learning properties. The stability of IL is arguably its main advantage over BP, which is consistent with claims that the main advantage of implicit SGD over standard/explicit SGD is its stability across learning rates (e.g., see [44, 45]). Second, we found IL minimized the loss more quickly than BP in the more biologically realistic scenario where single data points, rather than large mini-batches, are presented each training iteration. These simulations provide evidence that IL is more compatible with biologically realistic learning scenarios than BP. Collectively, our findings suggest IL is a promising basis for developing biologically constrained, high performing, local learning algorithms.

8 Broader Impacts

Our work is largely theoretical so it does not suggest any direct, possibly negative, societal impacts, as far as we can see.

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] Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical programming*, 129(2):163–195, 2011.

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

[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 back-propagation. *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] Georg B Keller and Thomas D Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.

[17] Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] Nikola T Markov, Julien Vezoli, Pascal Chameau, Arnaud Falchier, René Quilodran, Cyril Huissoud, Camille Lamy, Pierre Misery, Pascale Giroud, Shimon Ullman, et al. Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex. *Journal of Comparative Neurology*, 522(1):225–259, 2014.

[26] 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.

[27] 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.

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

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

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

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

[32] 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.

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

[34] 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.

[35] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[36] 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.

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

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

[39] 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.

[40] 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.

[41] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[42] 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.

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

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

[45] Panos Toulis, Dustin Tran, and Edo Airoldi. Towards stability and optimality in stochastic gradient descent. In *Artificial Intelligence and Statistics*, pages 1290–1298. PMLR, 2016.

[46] 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.

[47] James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

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

[49] 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]