DEEP LEARNING VIA MESSAGE PASSING ALGORITHMS BASED ON BELIEF PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Message-passing algorithms based on the Belief Propagation (BP) equations constitute a well-known distributed computational scheme. It is exact on tree-like graphical models and has also proven to be effective in many problems defined on graphs with loops (from inference to optimization, from signal processing to clustering). The BP-based scheme is fundamentally different from stochastic gradient descent (SGD), on which the current success of deep networks is based. In this paper, we present and adapt to mini-batch training on GPUs a family of BP-based message-passing algorithms with a reinforcement field that biases distributions towards locally entropic solutions. These algorithms are capable of training multi-layer neural networks with discrete weights and activations with performance comparable to SGD-inspired heuristics (BinaryNet) and are naturally well-adapted to continual learning. Furthermore, using these algorithms to estimate the marginals of the weights allows us to make approximate Bayesian predictions that have higher accuracy than point-wise solutions.

1 INTRODUCTION

Belief Propagation is a method for computing marginals in probabilistic inference problems (Bethe, 1935; Peierls, 1936; Gallager, 1962; Pearl, 1982). This includes optimization problems once they are written as zero temperature limit of a Gibbs distribution that uses the cost function as energy. Learning is one particular case, in which one wants to minimize a cost which is the loss error function. All these problems are typically NP-hard and message-passing techniques have been particularly successful at providing principled approximations through efficient distributed computations.

A particularly compact representation of inference/optimization problems that is used to built massage-passing algorithms is provided by factor graphs. A factor graph is a bipartite graph composed of variables nodes and factor nodes expressing the interactions among variables. Belief Propagation is exact for tree like factor graphs where the Gibbs distribution is naturally factorized, whereas it is approximate for graphs with loops, where the BP messages are known to minimize the so called Bethe free energy (originally introduced in statistical physics (Yedidia et al., 2003)). Still, loopy BP is routinely used with success in many real world applications ranging from error correcting codes, vision, clustering, just to mention a few. In all these problems, loops are indeed present in the factor graph and yet the variables are weakly correlated at long range and BP gives good results. A field in which BP has a long history is the statistical physics of disordered systems where it is known as Cavity Method (Mézard et al., 1987). It has been used to study the typical properties of spin glass models which represent binary variables interacting through random interactions over a given graph. It is very well known that in spin glass models defined on complete graphs and in locally tree-like random graphs, which are both loopy, the weak correlation conditions between variables may hold and BP give exact results (Mézard & Montanari, 2009).

Deep learning can be framed as a constraint satisfaction problem on an extremely loopy factor graph. Here we will consider mostly the case of ± 1 binary weights and sign activation functions, for which the messages and the marginals can be described simply by the difference between the probabilities associated with the +1 and -1 states, the so called *magnetizations*. The effectiveness of BP for Deep Learning has never been numerically tested in a systematic way, however there is clear evidence that the weak correlation decay condition does not hold and thus BP convergence and approximation quality is unpredictable.

In this paper we want to explore the effectiveness of a variant of BP that has shown excellent convergence properties in hard optimization problems and in non-convex shallow networks. It goes under the name of focusing BP (fBP) and is based on a probability distribution, a likelihood, that focuses on highly entropic wide minima, neglecting the contribution to marginals from narrow minima even when they are the majority (and hence dominate the Gibbs distribution). This version of BP is thus expected to give good results only in models that have such wide entropic minima as part of their energy landscape.

As discussed in Baldassi et al. (2016a), a simple way to define fBP is to add a "reinforcement" term to the BP equations: An iteration-dependent local field is introduced for each variable, with an intensity proportional to its marginal probability computed in the previous iteration step. This field is gradually increased until the entire system becomes fully biased on a configuration. The first version of reinforced BP was introduce in Braunstein & Zecchina (2006) as a heuristic algorithm to solve the learning problem in shallow binary networks.

Baldassi et al. (2016a) showed that this version of BP is a limiting case of fBP, i.e., BP equations written for a likelihood that uses the local entropy function instead of the error (energy) loss function. As discussed in depth in that study, one way to introduce a likelihood that focuses on highly entropic regions is to replicate the original factor graph y times; then, for each weight variable $W_{ij}^{a\,\ell}$, where ℓ identifies the layer and a the replicas, one must add an extra variable $W_{ij}^{*\,\ell}$ that plays the role of the center of the replicas, and introduce interactions between each replicated variable and the center to control the distance between the replicas and the center itself. Given the symmetry of the problem, one can make the simplifying assumption that the messages exchanged along the edges of the factor graph do not depend on the replica index. This assumption neglects some of the correlations between the replicate variables, but allows us to deal with only one factor graph and leads to a final set of equations that are identical to the BP equations for the original factor graph, plus an additional reinforcement message. The fBP algorithm can be used as a solver by gradually increasing the effect of the reinforcement message: one can control the size of the regions over which the fBP equations estimate the marginals by tuning the parameters that appear in the expression of the reinforcement message, until the high entropy regions reduce to a single configuration. Interestingly, by keeping the size of the high entropy region fixed, the fBP fixed point allows one to estimate the marginals and entropy relative to the region.

In this work, we present and adapt to GPU computation a family of fBP based message-passing algorithms that are capable of training multi-layer neural networks with generalization performance and computational speed comparable to SGD, thus revealing a family of parallel algorithms in high-dimensional spaces useful in realistic neural network contexts. Our version of fBP adds the reinforcement message at each mini-batch step. This is an additional approximation that however turns out to be quite benign, and allows us to choose the size of the mini-batch from the minimum size of 1 up to a fairly large number, without substantially affecting performance. We show through systematic simulations that fBP-based message-passing approaches prove to be efficient and competitive with the stochastic gradient variant from Hubara et al. (2016). Furthermore, using the message-passing algorithm not as a solver but as an estimator of marginals allows us to make local Bayesian predictions, in which we take the majority vote on all configurations represented by the marginals themselves. The resulting generalization error is significantly better than those of the solver, showing that, although approximate, the marginals of the weights estimated by message-passing retain useful information.

Consistently with the assumptions underlying fBP, we find that the solutions provided by the message passing algorithms belong to flat entropic regions of the loss landscape and have good performance in continual and online learning tasks as well.

The paper is structured as follows: In Sec. 2 we give a brief review of some related works. In Sec. 3 we provide a detailed description of the message-passing equations and of the high level structure of the algorithms. In Sec. 4 we compare performance of the message passing algorithms versus BinaryNet (Hubara et al., 2016), an efficient SGD inspired solver for binary neural networks, on a variety of deep architectures and for different data sets. We also analyze the type of solutions found by checking their flatness and their robustness in the continual learning setting.

2 RELATED WORKS

The literature on message passing algorithms is extensive, we refer to (Mézard & Montanari, 2009; Zdeborová & Krzakala, 2016) for a general overview. More related to our work, multilayer message-passing algorithms have been developed in inference contexts (Manoel et al., 2017; Fletcher et al., 2018), where they have been shown to produce exact marginals under certain statistical assumptions on (unlearned) weight matrices. The properties of message-passing for learning shallow neural networks have been extensively studied (see (Baldassi et al., 2020) and reference therein). Recent work improves BP by combining it with (Kuck et al., 2020; Satorras & Welling, 2021) graph neural networks. Interestingly, some work in computational neuroscience, e.g. (Rao, 2007), shows similarities to our approach.

The success of SGD is thought to be in part due to the geometrical structure of the loss landscape in neural networks (Baldassi et al., 2015; Chaudhari et al., 2017; Garipov et al., 2018; Pittorino et al., 2021; Feng & Tu, 2021). These considerations do not depend on the particular form of the SGD dynamics and should extend also to other types of algorithms, although SGD is by far the most popular choice among NNs practitioners due to its simplicity, flexibility, speed, and generalization performance.

While our work focuses on message passing schemes, some of the ideas presented here, such as the PasP rule, can be combined with algorithms for Bayesian neural networks' training (Hernández-Lobato & Adams, 2015; Wu et al., 2018).

3 LEARNING BY MESSAGE PASSING

3.1 POSTERIOR-AS-PRIOR UPDATES

We consider a multi-layer perceptron with L hidden neuron layers, having weight and bias parameters $\mathcal{W} = \{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\}_{\ell=0}^{L+1}$. We allow for stochastic activations $P^\ell(\boldsymbol{x}^{\ell+1}|\boldsymbol{z}^\ell)$, where $\boldsymbol{z}^\ell$ is the neuron's pre-activation vector for layer ℓ, and P^ℓ is assumed to be factorized over the neurons. If no stochasticity is present, P^ℓ will just encode an element-wise activation function. The probability of output y given an input $\boldsymbol{x}$ is than given by

$$P(y \,|\, \boldsymbol{x}, \mathcal{W}) = \int d\boldsymbol{x}^{1:L} \prod_{\ell=0}^{L} P^{\ell+1}(\boldsymbol{x}^{\ell+1} \mid \boldsymbol{W}^\ell \boldsymbol{x}^\ell + \boldsymbol{b}^\ell), \quad \text{where } \boldsymbol{x}^0 = \boldsymbol{x},\ \boldsymbol{x}^{L+1} = y. \tag{1}$$

In order to derive a set of message passing equation for learning the weights $\mathcal{W}$, we define a graphical model within the Bayesian formalism. Given a training set $D = \{(\boldsymbol{x}_n, y_n)\}_n$ and a prior distribution $q_\theta(\mathcal{W})$ in some parametric family, the posterior distribution is given by

$$P(\mathcal{W} \,|\, D, \theta) \propto \prod_n P(y_n \,|\, \boldsymbol{x}_n, \mathcal{W})\, q_\theta(\mathcal{W}). \tag{2}$$

Here $\propto$ denotes equality up to a normalization factor. Given the posterior, one can then compute the Bayesian prediction for a new data-point $\boldsymbol{x}$ through the average $P(y \,|\, \boldsymbol{x}, D, \theta) = \int d\mathcal{W}\ P(y \,|\, \boldsymbol{x}, \mathcal{W})\, P(\mathcal{W} \,|\, D, \theta)$. Unfortunately, the posterior is generically intractable due to the hard-to-compute normalization factor. On the other hand, we are mainly interested in training a distribution that covers wide minima of the loss landscape that generalize well (Baldassi et al., 2016a) and in recovering pointwise estimators within these regions. The Bayesian modeling becomes an auxiliary tool to set the stage for the message passing algorithms seeking flat minima. We also want to leverage the possibility to perform mini-batch updates in order to speed-up the training. We therefore devise an update scheme that we call posterior-as-prior (PasP), where we evolve a distribution $q_\theta^t(\mathcal{W})$ computed as an approximate and exponentially discounted mini-batch posterior. The previous iterate becomes the prior in the following step. In the PasP scheme, $q_\theta^t(\mathcal{W})$ retains the memory of past observations. Given a mini-batch $(\boldsymbol{X}^t, \boldsymbol{y}^t)$ sampled from the training set at time t and a scalar $\rho > 0$, the PasP update reads

$$\tilde{q}_{t+1}(\mathcal{W}) \approx P(\mathcal{W} \,|\, \boldsymbol{y}^t, \boldsymbol{X}^t, \theta^t), \tag{3}$$

$$q_{\theta^{t+1}}(\mathcal{W}) \propto [\tilde{q}_{t+1}(\mathcal{W})]^\rho. \tag{4}$$

We assume $\tilde{q}^{\rho}_{t+1}$ to lay in the distributions' space parametrized by θ, otherwise some projection is implicitly assumed Minka (2001). Notice that setting $\rho = 1$, the batch-size to 1, and taking a single pass over the dataset, we recover the Assumed Density Filtering algorithm (Minka, 2001). In this scheme, ρ plays a similar role as the learning rate. For large enough ρ (including $\rho = 1$), the iterations of q_{θ^t} will concentrate on a pointwise estimator. This mechanism mimics the reinforcement heuristic commonly used to turn belief propagation into a solver for constrained satisfaction problems (Braunstein & Zecchina, 2006) and related to flat-minima discovery (see focusing-BP in Baldassi et al. (2016a)]. A different prior updating mechanism which can be understood as empirical Bayes has been used in Baldassi et al. (2016b).

3.2 Belief Propagation for Neural Networks

We now describe the inner-loop of the PasP updates where we approximately compute the mini-batch posterior in 3 using message passing approaches derived by belief propagation. The factor graph for $P(\mathcal{W} \,|\, \boldsymbol{X}^t, \boldsymbol{y}^t, \theta^t)$ is the one given in equation 2, with the following additional specifications. For simplicity, we will ignore the bias term in each layer. We assume factorized $q_\theta(\mathcal{W})$, each factor parameterized by its first two moments, and we also drop the PasP iteration index t. For each example $(\boldsymbol{x}_n, y_n)$ in the mini-batch we augment the factor graph with the auxiliary variables $\boldsymbol{x}^\ell_n, \ell = 1, \dots, L$, representing the activation outputs. Moreover, we don't treat $P(y_n, | \, \boldsymbol{x}_n, \mathcal{W})$ in equation 2 as a unique factor since it would be intractable, but each example contributes the factors

$$\prod_{\ell=0}^{L} \prod_{k} P^{\ell+1}\left(x^{\ell+1}_{kn} \,\middle|\, \sum_i W^\ell_{ki} x^\ell_{in} \right) \quad \text{where } \boldsymbol{x}^0_n = \boldsymbol{x}_n,\ \boldsymbol{x}^{L+1}_n = y_n. \tag{5}$$

The factor graph thus defined is extremely loopy. Straightforward iteration of Belief Propagation will fail to converge or give meaningful results. Moreover, the neuron permutation symmetry in each hidden layer in presence of homogeneous weight prior induces a strongly attractive symmetric fixed point that hinders learning. The PasP scheme works around these issues by breaking the symmetry at time $t = 0$ with a non-homogeneous prior q_{θ^0}. In our experiments a little initial heterogeneity is enough to obtain specialized neurons at each time. Moreover, we don't require message passing convergence in the inner loop but perform one or a few iterations for each θ update. Another crucial ingredient is the use of a damping factor in the iterations. As we show in the following sections, these simple rules allow training deep networks by message passing.

We extend four different message passing algorithms, all of which are well known to the literature, to the deep network training setting: Belief Propagation (BP), BP-Inspired (BPI) message passing, mean-field (MF), and approximate message passing (AMP). The last three can be considered approximations of the first one. In the following paragraphs we will discuss the common traits, present the BP updates as an example, and refer to Appendix A for an in-depth exposition.

In all cases, message updates can be divided in a forward pass and backward pass, as also done in Fletcher et al. (2018) in a multi-layer inference setting. The BP algorithm is compactly reported in Algorithm 1.

Meaning of messages. All the messages involved in the message passing can be understood in terms of cavity marginals or full marginals (as mentioned in the introduction BP is also known as cavity method, see (Mézard & Montanari, 2009)). Of particular relevance are m^ℓ_{ki} and σ^ℓ_{ki}, denoting the mean and variance of the weights W^ℓ_{ki}. $\hat{x}^\ell_{in}$ and Δ^ℓ_{in} instead denote mean and variance of the i-th neuron's activation in layer ℓ for a given input $\boldsymbol{x}_n$.

Channel functions. All message passing schemes can be expressed using the following scalar channels, corresponding to single neuron and single weight effective free-energies:

$$\varphi^\ell(B, A, \omega, V) = \log \int \mathrm{d}x \, \mathrm{d}z \; e^{-\frac{1}{2}Ax^2 + Bx} \, P^\ell\left(x|z\right) e^{-\frac{(\omega - z)^2}{2V}} \qquad \ell = 1, \dots, L \tag{6}$$

$$\psi(H, G, \theta) = \log \int \mathrm{d}w \; e^{-\frac{1}{2}G^2 w^2 + Hw} \, q_\theta(w) \tag{7}$$

Notice that for common deterministic activations such as ReLU and sign, the function φ has analytic and smooth expressions. Same goes for ψ when $q_\theta(w)$ is Gaussian (continuous weights) or a mixture

of atoms (discrete weights). At the last layer we impose $P^{L+1}(y|z) = \mathbb{I}(y = \text{sign}(z))$ in binary classification tasks and $P^{L+1}(y|\boldsymbol{z}) = \mathbb{I}(y = \arg\max(\boldsymbol{z}))$ in multi-class classification (see A.5). While in our experiments we use hard constraints for the final output and deterministic activation functions, therefore solving a constraint satisfaction problem, it would be interesting to also consider soft constraints and introduce a temperature, but this is beyond the scope of our work.

Start and end of message passing. At the beginning of a new PasP iteration t, we reset the messages (see Appendix A) and run message passing for $\tau_{\max}$ iterations. We then compute the new prior $q_{\theta^{t+1}}(\mathcal{W})$ from the posterior given by the message passing.

BP Forward pass. Given an initialization of the messages at time $\tau = 0$, at time $\tau \geq 1$, starting from $\ell = 0$ and up to $\ell = L$, the forward BP updates read:

$$\hat{x}^{\ell,\tau}_{in\to k} = \partial_B \varphi^\ell \left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in} \right) \tag{8}$$

$$\Delta^{\ell,\tau}_{in\to k} = \partial^2_B \varphi^\ell \left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in} \right) \tag{9}$$

$$m^{\ell,\tau}_{ki\to n} = \partial_H \psi(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^\ell_{ki}) \tag{10}$$

$$\sigma^{\ell,\tau}_{ki\to n} = \partial^2_H \psi(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^\ell_{ki}) \tag{11}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki\to n} \right)^2 \Delta^{\ell,\tau}_{in\to k} + \sigma^{\ell,\tau}_{ki\to n} (\hat{x}^{\ell,\tau}_{in\to k})^2 + \sigma^{\ell,\tau}_{ki\to n} \Delta^{\ell,\tau}_{in\to k} \right) \tag{12}$$

$$\omega^{\ell,\tau}_{kn\to i} = \sum_{i' \neq i} m^{\ell,\tau}_{ki'\to n} \hat{x}^{\ell,\tau}_{i'n\to k} \tag{13}$$

BP Backward pass. Starting from $\ell = L$ and down to $\ell = 0$, for the BP backward pass we have:

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1} \left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn} \right) \tag{14}$$

$$\Gamma^{\ell,\tau}_{kn\to i} = -\partial^2_\omega \varphi^{\ell+1} \left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn} \right) \tag{15}$$

$$A^{\ell,\tau}_{in} = \sum_k \left((m^{\ell,\tau}_{ki\to n})^2 + \sigma^{\ell,\tau}_{ki\to n} \right) \Gamma^{\ell,\tau}_{kn\to i} - \sigma^{\ell,\tau}_{ki\to n} \left(g^{\ell,\tau}_{kn\to i} \right)^2 \tag{16}$$

$$B^{\ell,\tau}_{in\to k} = \sum_{k' \neq k} m^{\ell,\tau}_{k'i\to n} g^{\ell,\tau}_{k'n\to i} \tag{17}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left((\hat{x}^{\ell,\tau}_{in\to k})^2 + \Delta^{\ell,\tau}_{in\to k} \right) \Gamma^{\ell,\tau}_{kn\to i} - \Delta^{\ell,\tau}_{in\to k} \left(g^{\ell,\tau}_{kn\to i} \right)^2 \tag{18}$$

$$H^{\ell,\tau}_{ki\to n} = \sum_{n' \neq n} \hat{x}^{\ell,\tau}_{in'\to k} g^{\ell,\tau}_{kn'\to i} \tag{19}$$

On the parameter $\tau_{\max}$ In our experiments we fix $\tau_{\max} = 1$. We typically don't observe an increase of perfomance taking more steps, except for some specific cases and in particular for MF layers. We remark that for $\tau_{\max} = 1$ the BP and the BPI equations are identical, so in most of the subsequent numeuical results we will only investigate BPI.

Computational complexity The message passing equations boil down to elementwise operations and tensor contractions that we easily implement using the GPU friendly julia library Tullio.jl. For a layer of input and output size N and considering a batch-size of B, the time complexity of a forth-and-back iteration is $O(N^2 B)$ for all message passing algorithms (BP, BPI, MF, and AMP), the same as SGD. The prefactor varies and it is generally larger than SGD (see Appendix B.3). Also, time complexity for message passing is proportional to $\tau_{\max}$ (which we typically set to 1). We provide our implementation in the GitHub repo **anonymous**.

Algorithm 1: Deep Message Passing (DeepMP)

```
// Message passing used as inner loop in PasP Eqs.(3,4).
// Here we specifically refer to BP updates.
// BPI, MF, and AMP updates take the same form but using
// the rules in Appendix A.2, A.3, and A.4 respectively
```

Initialize messages.
for $\tau = 1, \dots \tau_{\max}$ **do**
  // Forward Pass
  for $l = 0, \dots, L$ **do**
    compute $\hat{\boldsymbol{x}}^\ell, \boldsymbol{\Delta}^\ell$ using (8, 9)
    compute $\boldsymbol{m}^\ell, \boldsymbol{\sigma}^\ell$ using (10, 11)
    compute $\mathbf{V}^\ell, \boldsymbol{\omega}^\ell$ using (12, 13)
  // Backward Pass
  for $l = L, \dots, 0$ **do**
    compute $\boldsymbol{g}^\ell, \boldsymbol{\Gamma}^\ell$ using (14, 15)
    compute $\boldsymbol{A}^\ell, \boldsymbol{B}^\ell$ using (16, 17)
    compute $\boldsymbol{G}^\ell, \boldsymbol{H}^\ell$ using (18, 19)

4 NUMERICAL RESULTS

We specialize and test our message passing algorithms on neural networks with binary weights and activations. In this section we compare their performance with SGD adapted to this type of architectures. Our procedure for SGD is based on the one described in Hubara et al. (2016), which we call BinaryNet along the paper (see Appendix B.1.4 for details). Neither in the message passing nor in the BinaryNet algorithms do we use levels of bias or batch normalization, both because they are not necessary for achieving satisfactory performance and to maintain fair comparison. These variants, along with the training of continuous weights, continuous activations, and convolutional layers will be studied in future works.

We find that message-passing algorithms are able to train generic MLP architectures (with varying numbers and sizes of hidden layers); as for the datasets, we are able to perform both binary classification and multiclass classification on standard computer vision datasets such as MNIST, Fashion-MNIST, and CIFAR-10. Since these datasets consist of 10 classes, for the binary classification task we divide each dataset in two classes (even vs odd).

We report that it is possible to find values of the hyper-parameters such that BP-based algorithms are able to solve these optimization problems with generalization performance comparable to BinaryNet. The training error is usually lower for optimized configurations from message passing schemes, suggesting that these algorithms are able to achieve higher capacity than SGD-based algorithms (given the same test error). In our experiments, this is true for fully connected generic multilayer architectures (i.e., with generic depth/hidden layer sizes) and realistic datasets, and for generic dataset sizes and batch sizes (from $bs = 1$ up to $bs = 1024$ in our experiments). The computational complexity of message passing schemes is the same as that of BinaryNet. In fact, we can implement BP-based algorithms on GPUs with computation times comparable to those of BinaryNet, see Appendix B.3.

4.1 TESTING ARCHITECTURES

We fix the dataset, namely we perform multi-class classification on Fashion-MNIST, and we compare the BP-based algorithms with BinaryNet for different choices of the architecture (i.e. we vary the number and the size of the hidden layers). We present the learning curves for a MLP with 2 hidden layers with 101 units each and a MLP with 3 hidden layers with 501 units each in Fig.1, and similar results hold in our experiments with 2 or 3 hidden layers of 101, 501 or 1001 units each. The parameters used in our simulations are reported in Appendix B.1.

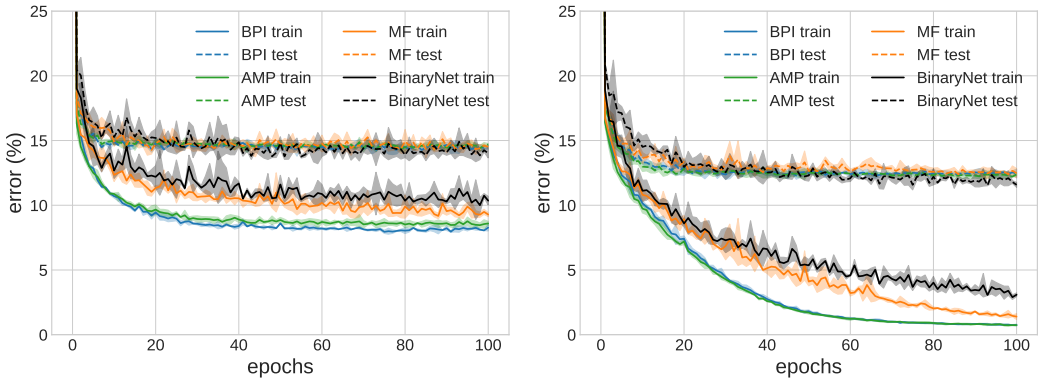

Figure 1: Test and training error curves of the message passing algorithms compared with BinaryNet on the Fashion-MNIST dataset (multi-class classification). (Left) MLP with 2 hidden layers of 101 units; (Right) MLP with 3 hidden layers of 501 units. The batch-size is $bs = 128$, curves are averaged over 5 realizations of the initial condition.

4.2 Testing Datasets

We now fix the architecture (MLP with 2 hidden layers of 501 neurons each) and we vary the dataset, i.e. we test the BP-based algorithms on standard computer vision benchmark datasets such as MNIST, Fashion-MNIST and CIFAR-10 (both multi-class and binary classification). In Tab. 1 we report the final test errors (averaged over 5 realizations of the initial conditions) obtained by the BP-based algorithms (BPI, AMP, MF) compared to the baseline (BinaryNet). See Appendix B.1 for the corresponding train errors and the parameters used in the simulations. While test performance is mostly comparable, the train error tends to be lower for the BP-based algorithms.

Dataset	BinaryNet	BP(I)	AMP	MF
MNIST (2 classes)	1.3 ± 0.1	1.4 ± 0.2	1.6 ± 0.1	1.3 ± 0.2
Fashion-MNIST (2 classes)	2.4 ± 0.1	2.3 ± 0.1	2.5 ± 0.1	2.31 ± 0.02
CIFAR-10 (2 classes)	30.0 ± 0.3	31.4 ± 0.1	31.0 ± 0.6	31.1 ± 0.4
MNIST	2.2 ± 0.1	2.3 ± 0.1	2.3 ± 0.1	2.3 ± 0.1
Fashion-MNIST	12.0 ± 0.6	11.8 ± 0.2	11.8 ± 0.1	12.1 ± 0.2
CIFAR-10	59.0 ± 0.7	58.6 ± 0.7	59.2 ± 0.6	60.4 ± 1.1

Table 1: Test error (%) on Fashion-MNIST of a MLP with 2 hidden layers of 501 units for BinaryNet (baseline), BPI, AMP and MF. All algorithms are trained with batch-size 128 and for 100 epochs. Mean and standard deviations are calculated over 5 random initializations.

4.3 Locally Bayesian error and local energy

Bayesian error. The BP-based message passing framework, used as an estimator of the marginals, allows us to compute the Bayesian error, i.e., the majority vote on all possible solutions represented by the marginals themselves. This allows us to make predictions with better generalization error than point-wise solutions, showing that the marginals retain useful information. However, we roughly estimate the marginals with the mini-batch procedure (the exact ones should be computed with a full-batch procedure, but this converges with difficulty in our tests). Since BP-based algorithms tend to focus on dense states (as also confirmed by Fig. 2, right panel), the Bayesian error we compute can be considered as a local approximation of the full one. We report results on binary classification on the MNIST dataset in Fig. 2 (left panel), and we observe the same performance increase on different datasets and architectures. We obtain the Bayesian prediction from the output marginal given by a single forward pass of the message passing.

To obtain good Bayesian estimates, it is important that the marginal distributions do not concentrate too much (otherwise, the Bayesian error will to converge to the error of a single configuration).

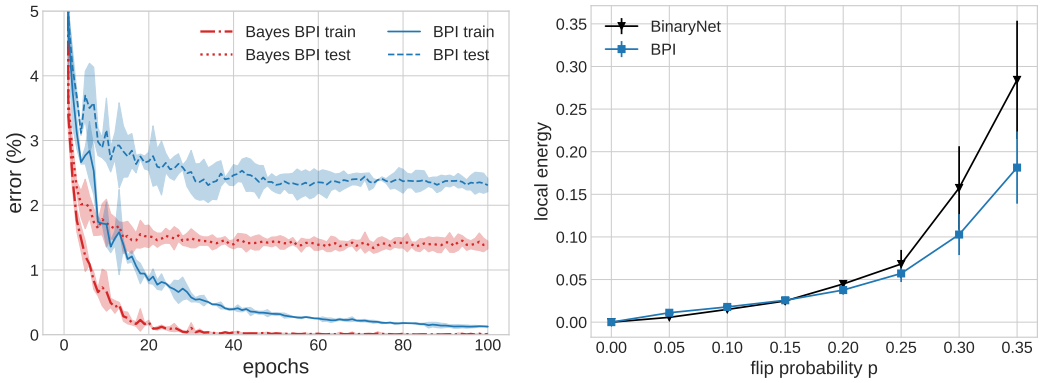

Figure 2: (Left) Test and training error curves of Bayes and point-wise predictions obtained with BPI by a multi-layer perceptron with 2 hidden layers of 101 units on the MNIST dataset (binary classification). The training hyper-parameters in the two cases are independently selected and generally differ. (Right) Local energy curve of the point-wise configuration found by the BPI algorithm compared with BinaryNet on the MNIST dataset (multi-class classification), same architecture.

Local energy We adapt the notion of flatness used in (Jiang et al., 2020; Pittorino et al., 2021), that we call local energy, to configurations with binary weights. Given a weight configuration $\boldsymbol{w} \in \{\pm 1\}^N$, we define the local energy $\delta E_{\text{train}}(\boldsymbol{w}, p)$ as the average difference in training error $E_{\text{train}}(\boldsymbol{w})$ when perturbing $\boldsymbol{w}$ by flipping a random fraction p of its elements:

$$\delta E_{\text{train}}(\boldsymbol{w}, p) = \mathbb{E}_{\boldsymbol{z}}\, E_{\text{train}}(\boldsymbol{w} \odot \boldsymbol{z}) - E_{\text{train}}(\boldsymbol{w}), \tag{20}$$

where $\odot$ denotes the Hadamard (element-wise) product and the expectation is over i.i.d. entries for $\boldsymbol{z}$ equal to -1 with probability p and to $+1$ with probability $1-p$.

We report the resulting local energy profiles (in a range $[0, p_{\max}]$) in Fig. 2 right panel for BPI and BinaryNet. The relative error grows slowly when perturbing the trained configurations (notice the convexity of the curves). This shows that both BP-based and SGD-based algorithms find configurations that lie in relatively flat minima in the energy landscape. The same qualitative phenomenon holds for different data sets, different MLP architectures, and different BP approximations.

4.4 Message passing for continual learning

Given the high local entropy (i.e., the flatness) of the solutions found by the BP-based algorithms, we tested them on a classic machine learning problem where the possibility of rearranging solutions at low training error can be advantageous: how to prevent catastrophic forgetting. When a deep network is trained sequentially on different tasks, it tends to forget exponentially fast the previously seen tasks when a new task is learned (McCloskey & Cohen, 1989; Robins, 1995; Fusi et al., 2005). Quite consistently, recent work (Feng & Tu, 2021) has shown that searching for a flat region in the loss landscape can indeed help to prevent catastrophic forgetting.

In recent years several implementations have been proposed to mitigate the catastrophic forgetting effect in deep neural networks (see e.g. Kirkpatrick et al. (2017); Aljundi et al. (2018); Zenke et al. (2017)). All of these methods however succeed in the continual learning scenario by imposing some constraint on the loss function based on the previously learned task. Here we show that our message passing scheme is in fact naturally prone to learn multiple tasks sequentially, mitigating the characteristic memory issues of the gradient-like optimization scheme, without the need for explicit modifications.

As a prototypical experiment, we sequentially trained a multi-layer neural network on 6 different versions of the MNIST dataset, where the pixels of the images have been randomly permuted (Goodfellow et al., 2013), giving a fixed budget of 40 epochs on each task. As a representative

architecture, we present the results for a two hidden layer neural network with 2001 units on each layer (see Appendix B.1 for the implementation details of BPI). As can be seen in Fig. 3, at the end of the training the BPI algorithm is able to reach good generalization performances on all the tasks. We compared the BPI performance with BinaryNet, which is in general much more adapt to progressively learn new datasets than standard SGD with continuous weights (see e.g. the discussion in Laborieux et al. (2021)). As we show in Fig. 3, one can leverage the BinaryNet performances by lowering the learning rate, but this comes at the cost of poor single-task generalization. While our BP implementation is not competitive with ad-hoc techniques specifically designed for this problem, these preliminary results are encouraging and we are currently investigating a simple modification of the marginals update equations (in the spirit of Laborieux et al. (2021)) that we believe can set message passing as a competitive technique for the continual learning task.

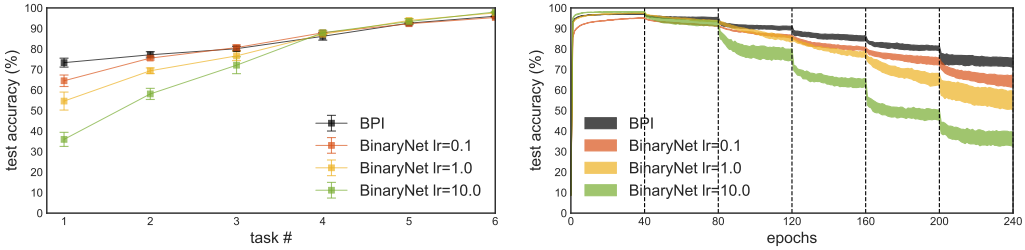

Figure 3: Performance of BPI and BinaryNet on the permuted MNIST task (see text) for a two hidden layer network with 2001 units on each layer. In this scenario the model is trained sequentially on 6 different versions of the MNIST dataset (the tasks), where the pixels have been permuted. In this setting we fixed the number of epochs for each task to 40. (Left) Test accuracy on each task after the network has been trained on all the tasks. (Right) Test accuracy on the first task as a function of the number of epochs. Points are averages over 5 independent runs, while shaded areas are errors on the mean.

5 Discussion and conclusions

Message passing algorithms, while successful in many fields, have notoriously struggled to scale to the training problems of deep neural networks. Here we have developed a class of fBP-based message passing algorithms, used within an update scheme that makes message passing training on deep and wide multilayer perceptrons possible. An important algorithmic solution we have introduced is the Posterior-as-Prior (PasP) step in the iteration process. While we focus on binary networks in our experiments, the formalism also accounts for continuous weights and continuous activations, and can be extended to incorporate bias. Future work will fill in these gaps and attempt to embrace convolutional layers as well. The BP-based scheme can also be adapted to various effective training procedures such as batch normalization (Feizi et al., 2018). Another interesting direction is the algorithmic computation of the (local) entropy of the model from the messages.

Further theoretical work is needed for a more complete understanding of the robustness of our approach. Recent developments in message passing algorithms Rangan et al. (2019) and related theoretical analysis Goldt et al. (2020) could provide fruitful inspirations.

Acknowledgments

We thank Carlo Baldassi for useful discussions on the numerical implementation of the BP equations.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Carlo Baldassi, Alfredo Braunstein, Nicolas Brunel, and Riccardo Zecchina. Efficient supervised learning in networks with binary synapses. *Proceedings of the National Academy of Sciences*, 104(26):11079–11084, 2007. ISSN 0027-8424. doi: 10.1073/pnas.0700324104. URL https://www.pnas.org/content/104/26/11079.

Carlo Baldassi, Alessandro Ingrosso, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Phys. Rev. Lett.*, 115:128101, Sep 2015. doi: 10.1103/PhysRevLett.115.128101. URL https://link.aps.org/doi/10.1103/PhysRevLett.115.128101.

Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Alessandro Ingrosso, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences*, 113(48):E7655–E7662, 2016a. ISSN 0027-8424. doi: 10.1073/pnas.1608103113. URL https://www.pnas.org/content/113/48/E7655.

Carlo Baldassi, Federica Gerace, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Learning may need only a few bits of synaptic precision. *Phys. Rev. E*, 93:052313, May 2016b. doi: 10.1103/PhysRevE.93.052313. URL https://link.aps.org/doi/10.1103/PhysRevE.93.052313.

Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1): 161–170, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1908636117. URL https://www.pnas.org/content/117/1/161.

Hans Bethe. Statistical theory of superlattices. *Proc. R. Soc. A*, 150:552, 1935.

Alfredo Braunstein and Riccardo Zecchina. Learning by message passing in networks of discrete synapses. *Phys. Rev. Lett.*, 96:030201, Jan 2006. doi: 10.1103/PhysRevLett.96.030201. URL https://link.aps.org/doi/10.1103/PhysRevLett.96.030201.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1YfAfcgl.

David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

Soheil Feizi, Hamid Javadi, Jesse Zhang, and David Tse. Porcupine neural networks: Approximating neural network landscapes. *Advances in Neural Information Processing Systems*, 31:4831–4841, 2018.

Yu Feng and Yuhai Tu. The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9), 2021.

Alyson K Fletcher, Sundeep Rangan, and Philip Schniter. Inference in deep networks in high dimensions. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1884–1888. IEEE, 2018.

Stefano Fusi, Patrick J Drew, and Larry F Abbott. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, 2005.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1): 21–28, 1962.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/be3087e74e9100d4bc4c6268cdbe8456-Paper.pdf.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10 (4):041044, 2020.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1861–1869. JMLR.org, 2015.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Jonathan Kuck, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon. Belief propagation neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 667–678. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/07217414eb3fbe24d4e5b6cafb91ca18-Paper.pdf.

Axel Laborieux, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. Synaptic metaplasticity in binarized neural networks. *Nature Communications*, 12(1):2549, May 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-22768-y. URL https://doi.org/10.1038/s41467-021-22768-y.

Andre Manoel, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Multi-layer generalized linear estimation. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2098–2102, 2017. doi: 10.1109/ISIT.2017.8006899.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Marc Mézard. Mean-field message-passing equations in the hopfield model and its generalizations. *Physical Review E*, 95(2):022117, 2017.

Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, Inc., USA, 2009. ISBN 019857083X.

Judea Pearl. *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science . . . , 1982.

R. Peierls. On ising's model of ferromagnetism. *Mathematical Proceedings of the Cambridge Philosophical Society*, 32(3):477–481, 1936. doi: 10.1017/S0305004100019174.

Fabrizio Pittorino, Carlo Lucibello, Christoph Feinauer, Gabriele Perugini, Carlo Baldassi, Elizaveta Demyanenko, and Riccardo Zecchina. Entropic gradient descent algorithms and wide flat minima. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=xjXg0bnoDmS`.

Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Vector approximate message passing. *IEEE Transactions on Information Theory*, 65(10):6664–6684, 2019.

Rajesh P. N. Rao. *Neural models of Bayesian belief propagation.*, pp. 239–267. Bayesian brain: Probabilistic approaches to neural coding. MIT Press, Cambridge, MA, US, 2007. ISBN 026204238X (Hardcover); 978-0-262-04238-3 (Hardcover).

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.

Victor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, pp. 685–693. PMLR, 2021.

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. *Understanding Belief Propagation and Its Generalizations*, pp. 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1558608117.

Lenka Zdeborová and Florent Krzakala. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

A BP-BASED MESSAGE PASSING ALGORITHMS

Given a mini-batch $\mathcal{B} = \{(\boldsymbol{x}_n, y_n)\}_n$, the factor graph defined by Eqs. (1, 2, 5) is explicitly written as

$$P(\mathcal{W}, \boldsymbol{x}^{1:L} \,|\, \mathcal{B}, \theta) \propto \prod_{\ell=0}^{L} \prod_{k,n} P^{\ell+1}\left(x_{kn}^{\ell+1} \,\middle|\, \sum_i W_{ki}^{\ell} x_{in}^{\ell} \right) \prod_{k,i,\ell} q_\theta(W_{ki}^{\ell}). \tag{21}$$

where $\boldsymbol{x}_n^0 = \boldsymbol{x}_n,\ \boldsymbol{x}_n^{L+1} = y_n$. Derivation of the BP equations for this model is straightforward although lengthy and involved. Although not attempted before in deep neural networks, it is obtained following the step presented in multiple papers, books, and reviews, see for instance (Mézard & Montanari, 2009; Zdeborová & Krzakala, 2016; Mézard, 2017). We present here only the final result and remand to the aforementioned references for details of the generic scheme and it's specialization in simpler settings. It should be noted that a (common) approximation that we take here with respect to the standard BP scheme, is that messages are assumed to be Gaussian and therefore parameterized by mean and variance. This goes by the name of relaxed belied propagation (rBP), just BP throughout the paper.

From BP, we derive other 3 message passing algorithms, all of which are well known to the literature, to the deep network training setting: BP-Inspired (BPI) message passing, mean-field (MF), and

approximate message passing (AMP). In all cases, message updates can be divided in a forward pass and backward pass, as also done in Fletcher et al. (2018) in a multi-layer inference setting. The BP algorithm is compactly reported in Algorithm 1.

In our notation, ℓ denotes the layer index, τ the BP iteration index, k an output neuron index, i an input neuron index, and n a sample index.

We report below, for convenience, some of the considerations also given in the main text.

Meaning of messages. All the messages involved in the message passing can be understood in terms of cavity marginals or full marginals (as mentioned in the introduction BP is also known as cavity method, see (Mézard & Montanari, 2009)). Of particular relevance are m^{ℓ}_{ki} and σ^{ℓ}_{ki}, denoting the mean and variance of the weights W^{ℓ}_{ki}. $\hat{x}^{\ell}_{in}$ and Δ^{ℓ}_{in} instead denote mean and variance of the i-th neuron's activation in layer ℓ for a given input $\boldsymbol{x}_n$.

Channel functions. All message passing schemes can be expressed using the following scalar channels, corresponding to single neuron and single weight effective free-energies:

$$\varphi^{\ell}(B, A, \omega, V) = \log \int \mathrm{d}x\, \mathrm{d}z\; e^{-\frac{1}{2}Ax^2+Bx}\, P^{\ell}\left(x|z\right) e^{-\frac{(\omega-z)^2}{2V}} \qquad \ell = 1, \dots, L \tag{22}$$

$$\psi(H, G, \theta) = \log \int \mathrm{d}w\; e^{-\frac{1}{2}G^2w^2+Hw}\, q_{\theta}(w) \tag{23}$$

Notice that for common deterministic activations such as ReLU and sign, the function φ has analytic and smooth expressions. Same goes for ψ when $q_{\theta}(w)$ is Gaussian (continuous weights) or a mixture of atoms (discrete weights). At the last layer we impose $P^{L+1}(y|z) = \mathbb{I}(y = \mathrm{sign}(z))$ in binary classification tasks and $P^{L+1}(y|\boldsymbol{z}) = \mathbb{I}(y = \arg\max(\boldsymbol{z}))$ in multi-class classification (see A.5). While in our experiments we use hard constraints for the final output and deterministic activation functions, therefore solving a constraint satisfaction problem, it would be interesting to also consider soft constraints and introduce a temperature, but this is beyond the scope of our work.

Start and end of message passing. At the beginning of a new PasP iteration t, we reset the messages (see Appendix A) and run message passing for $\tau_{\max}$ iterations. We then compute the new prior $q_{\theta^{t+1}}(\mathcal{W})$ from the posterior given by the message passing.

A.1 BP ALGORITHM

This is the base BP algorithm also reported in the main text.

Forward pass.

$$\hat{x}^{\ell,\tau}_{in\to k} = \partial_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right) \tag{24}$$

$$\Delta^{\ell,\tau}_{in\to k} = \partial^2_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right) \tag{25}$$

$$m^{\ell,\tau}_{ki\to n} = \partial_H \psi(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}) \tag{26}$$

$$\sigma^{\ell,\tau}_{ki\to n} = \partial^2_H \psi(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}) \tag{27}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki\to n}\right)^2 \Delta^{\ell,\tau}_{in\to k} + \sigma^{\ell,\tau}_{ki\to n}(\hat{x}^{\ell,\tau}_{in\to k})^2 + \sigma^{\ell,\tau}_{ki\to n}\Delta^{\ell,\tau}_{in\to k} \right) \tag{28}$$

$$\omega^{\ell,\tau}_{kn\to i} = \sum_{i'\neq i} m^{\ell,\tau}_{ki'\to n}\hat{x}^{\ell,\tau}_{i'n\to k} \tag{29}$$

Backward pass.

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn}\right) \tag{30}$$

$$\Gamma^{\ell,\tau}_{kn\to i} = -\partial^2_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn}\right) \tag{31}$$

$$A^{\ell,\tau}_{in} = \sum_k \left((m^{\ell,\tau}_{ki\to n})^2 + \sigma^{\ell,\tau}_{ki\to n}\right)\Gamma^{\ell,\tau}_{kn\to i} - \sigma^{\ell,\tau}_{ki\to n}\left(g^{\ell,\tau}_{kn\to i}\right)^2 \tag{32}$$

$$B^{\ell,\tau}_{in\to k} = \sum_{k'\neq k} m^{\ell,\tau}_{k'i\to n} g^{\ell,\tau}_{k'n\to i} \tag{33}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left((\hat{x}^{\ell,\tau}_{in\to k})^2 + \Delta^{\ell,\tau}_{in\to k}\right)\Gamma^{\ell,\tau}_{kn\to i} - \Delta^{\ell,\tau}_{in\to k}\left(g^{\ell,\tau}_{kn\to i}\right)^2 \tag{34}$$

$$H^{\ell,\tau}_{ki\to n} = \sum_{n'\neq n} \hat{x}^{\ell,\tau}_{in'\to k}\, g^{\ell,\tau}_{kn'\to i} \tag{35}$$

A.2 BPI ALGORITHM

The BP-Inspired algorithm (BPI) is obtained as an approximation of BP replacing some cavity quantities with their non-cavity counterparts. What we obtain is a generalization of the single layer algorithm of (Baldassi et al., 2007).

Forward pass.

$$\hat{x}^{\ell,\tau}_{in} = \partial_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right) \tag{36}$$

$$\Delta^{\ell,\tau}_{in} = \partial^2_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right) \tag{37}$$

$$m^{\ell,\tau}_{ki} = \partial_H \psi(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}) \tag{38}$$

$$\sigma^{\ell,\tau}_{ki} = \partial^2_H \psi(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}) \tag{39}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki}\right)^2 \Delta^{\ell,\tau}_{in} + \sigma^{\ell,\tau}_{ki}(\hat{x}^{\ell,\tau}_{in})^2 + \sigma^{\ell,\tau}_{ki}\Delta^{\ell,\tau}_{in}\right) \tag{40}$$

$$\omega^{\ell,\tau}_{kn} = \sum_i m^{\ell,\tau}_{ki}\,\hat{x}^{\ell,\tau}_{in} \tag{41}$$

Backward pass.

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn} - m^{\ell,\tau}_{ki}\,\hat{x}^{\ell,\tau}_{ai}, V^{\ell,\tau}_{kn}\right) \tag{42}$$

$$\Gamma^{\ell,\tau}_{kn} = -\partial^2_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn}, V^{\ell,\tau}_{kn}\right) \tag{43}$$

$$A^{\ell,\tau}_{in} = \sum_k \left((m^{\ell,\tau}_{ki})^2 + \sigma^{\ell,\tau}_{ki}\right)\Gamma^{\ell,\tau}_{kn} - \sigma^{\ell,\tau}_{ki}\left(g^{\ell,\tau}_{kn}\right)^2 \tag{44}$$

$$B^{\ell,\tau}_{in} = \sum_k m^{\ell,\tau}_{ki}\, g^{\ell,\tau}_{kn\to i} \tag{45}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left((\hat{x}^{\ell,\tau}_{in})^2 + \Delta^{\ell,\tau}_{in}\right)\Gamma^{\ell,\tau}_{kn} - \Delta^{\ell,\tau}_{in}\left(g^{\ell,\tau}_{kn}\right)^2 \tag{46}$$

$$H^{\ell,\tau}_{ki} = \sum_n \hat{x}^{\ell,\tau}_{in}\, g^{\ell,\tau}_{kn\to i} \tag{47}$$

A.3 MF ALGORITHM

The mean-field (MF) equations are obtained as a further simplification of BPI, using only non-cavity quantities. Although the simplification appears minimal at this point, we empirically observe a large discrepancy between the two algorithms.

Forward pass.

$$\hat{x}_{in}^{\ell,\tau} = \partial_B \varphi^\ell \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right) \tag{48}$$

$$\Delta_{in}^{\ell,\tau} = \partial_B^2 \varphi^\ell \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right) \tag{49}$$

$$m_{ki}^{\ell,\tau} = \partial_H \psi(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^\ell) \tag{50}$$

$$\sigma_{ki}^{\ell,\tau} = \partial_H^2 \psi(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^\ell) \tag{51}$$

$$V_{kn}^{\ell,\tau} = \sum_i \left(\left(m_{ki}^{\ell,\tau} \right)^2 \Delta_{in}^{\ell,\tau} + \sigma_{ki}^{\ell,\tau} (\hat{x}_{in}^{\ell,\tau})^2 + \sigma_{ki}^{\ell,\tau} \Delta_{in}^{\ell,\tau} \right) \tag{52}$$

$$\omega_{kn}^{\ell,\tau} = \sum_i m_{ki}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau} \tag{53}$$

Backward pass.

$$g_{kn}^{\ell,\tau} = \partial_\omega \varphi^{\ell+1} \left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau} \right) \tag{54}$$

$$\Gamma_{kn}^{\ell,\tau} = -\partial_\omega^2 \varphi^{\ell+1} \left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau} \right) \tag{55}$$

$$A_{in}^{\ell,\tau} = \sum_k \left((m_{ki}^{\ell,\tau})^2 + \sigma_{ki}^{\ell,\tau} \right) \Gamma_{kn}^{\ell,\tau} - \sigma_{ki}^{\ell,\tau} \left(g_{kn}^{\ell,\tau} \right)^2 \tag{56}$$

$$B_{in}^{\ell,\tau} = \sum_k m_{ki}^{\ell,\tau} g_{kn}^{\ell,\tau} \tag{57}$$

$$G_{ki}^{\ell,\tau} = \sum_n \left((\hat{x}_{in}^{\ell,\tau})^2 + \Delta_{in}^{\ell,\tau} \right) \Gamma_{kn}^{\ell,\tau} - \Delta_{in}^{\ell,\tau} \left(g_{kn}^{\ell,\tau} \right)^2 \tag{58}$$

$$H_{ki}^{\ell,\tau} = \sum_n \hat{x}_{in}^{\ell,\tau} g_{kn}^{\ell,\tau} \tag{59}$$

A.4 AMP ALGORITHM

A closure of the BP messages using only non-cavity quantities, adding terms neglected by MF according to some statistical assumptions Donoho et al. (2009); Zdeborová & Krzakala (2016); Mézard (2017).

Forward pass.

$$\hat{x}_{in}^{\ell,\tau} = \partial_B \varphi^\ell \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right) \tag{60}$$

$$\Delta_{in}^{\ell,\tau} = \partial_B^2 \varphi^\ell \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right) \tag{61}$$

$$m_{ki}^{\ell,\tau} = \partial_H \psi(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^\ell) \tag{62}$$

$$\sigma_{ki}^{\ell,\tau} = \partial_H^2 \psi(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^\ell) \tag{63}$$

$$V_{kn}^{\ell,\tau} = \sum_i \left(\left(m_{ki}^{\ell,\tau} \right)^2 \Delta_{in}^{\ell,\tau} + \sigma_{ki}^{\ell,\tau} (\hat{x}_{in}^{\ell,\tau})^2 + \sigma_{ki}^{\ell,\tau} \Delta_{in}^{\ell,\tau} \right) \tag{64}$$

$$\omega_{kn}^{\ell,\tau} = \sum_i m_{ki}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau} - g_{kn}^{\ell,\tau-1} \sum_i \left(\left(m_{ki}^{\ell,\tau} \right)^2 \Delta_{in}^{\ell,\tau} + \sigma_{ki}^{\ell,\tau} (\hat{x}_{in}^{\ell,\tau})^2 \right) \tag{65}$$

Backward pass.

$$g_{kn}^{\ell,\tau} = \partial_\omega \varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right) \tag{66}$$

$$\Gamma_{kn}^{\ell,\tau} = -\partial_\omega^2 \varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right) \tag{67}$$

$$A_{in}^{\ell,\tau} = \sum_k \left((m_{ki}^{\ell,\tau})^2 + \sigma_{ki}^{\ell,\tau}\right)\Gamma_{kn}^{\ell,\tau} - \sigma_{ki}^{\ell,\tau}\left(g_{kn}^{\ell,\tau}\right)^2 \tag{68}$$

$$B_{in}^{\ell,\tau} = \sum_k \left(m_{ki}^{\ell,\tau} g_{kn}^{\ell,\tau} - \sigma_{ki}^{\ell,\tau}\Gamma_{kn}^{\ell,\tau}\right) + \hat{x}_{in}^{\ell,\tau}\left(\sum_k (m_{ki}^{\ell,\tau})^2 \Gamma_{kn}^{\ell,\tau}\right) \tag{69}$$

$$G_{ki}^{\ell,\tau} = \sum_n \left((\hat{x}_{in}^{\ell,\tau})^2 + \Delta_{in}^{\ell,\tau}\right)\Gamma_{kn}^{\ell,\tau} - \Delta_{in}^{\ell,\tau}\left(g_{kn}^{\ell,\tau}\right)^2 \tag{70}$$

$$H_{ki}^{\ell,\tau} = \sum_n \left(\hat{x}_{in}^{\ell,\tau} g_{kn}^{\ell,\tau} - \Delta_{in}^{\ell,\tau}\Gamma_{kn}^{\ell,\tau}\right) + m_{ki}^{\ell,\tau}\left(\sum_n (\hat{x}_{in}^{\ell,\tau})^2 \Gamma_{kn}^{\ell,\tau}\right) \tag{71}$$

A.5 ARGMAX LAYER

In order to perform multi-class classification, we have to perform an argmax operation on the last layer of the neural network. Call z_k, for $k = 1, \ldots, K$, the Gaussian random variables output of the last layer of the network in correspondence of some input $\boldsymbol{x}$. Assuming the correct label is class k^*, the effective partition function corresponding to the output constraint reads

$$Z_{k^*} = \int \prod_k dz_k \, \mathcal{N}(z_k; \omega_k, V_k) \prod_{k \neq k^*} \theta(z_{k^*} - z_k) \tag{72}$$

$$= \int dz_{k^*} \, \mathcal{N}(z_{k^*}; \omega_{k^*}, V_{k^*}) \prod_{k \neq k^*} H\left(-\frac{z_{k^*} - \omega_k}{\sqrt{V_k}}\right) \tag{73}$$

This last integral is intractable, therefore we have to resort to approximations.

A.5.1 APPROACH 1: JENSEN INEQUALITY

Using the Jensen inequality we obtain

$$\phi_{k^*} = \log Z_{k^*} = \log \mathbb{E}_{z \sim \mathcal{N}(\omega_{k^*}, V_{k^*})} \prod_{k \neq k^*} H\left(-\frac{z - \omega_k}{\sqrt{V_k}}\right) \tag{74}$$

$$\geq \sum_{k \neq k^*} \mathbb{E}_{z \sim \mathcal{N}(\omega_{k^*}, V_{k^*})} \log H\left(-\frac{z - \omega_k}{\sqrt{V_k}}\right) \tag{75}$$

Reparametrizing the expectation we have:

$$\tilde{\phi}_{k^*} = \sum_{k \neq k^*} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \log H\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}}\right) \tag{76}$$

The derivative $\partial_{\omega_k} \tilde{\phi}_{k^*}$ and $\partial_{\omega_k}^2 \tilde{\phi}_{k^*}$ that we need can then be estimated by sampling (once) ϵ.

$$\partial_{\omega_k} \tilde{\phi}_{k^*} = \begin{cases} -\frac{1}{\sqrt{V_k}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \, GH\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}}\right) & k \neq k^* \\ \sum_{k' \neq k^*} \frac{1}{\sqrt{V_{k'}}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \, GH\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_{k'}}{\sqrt{V_{k'}}}\right) & k = k^* \end{cases} \tag{77}$$

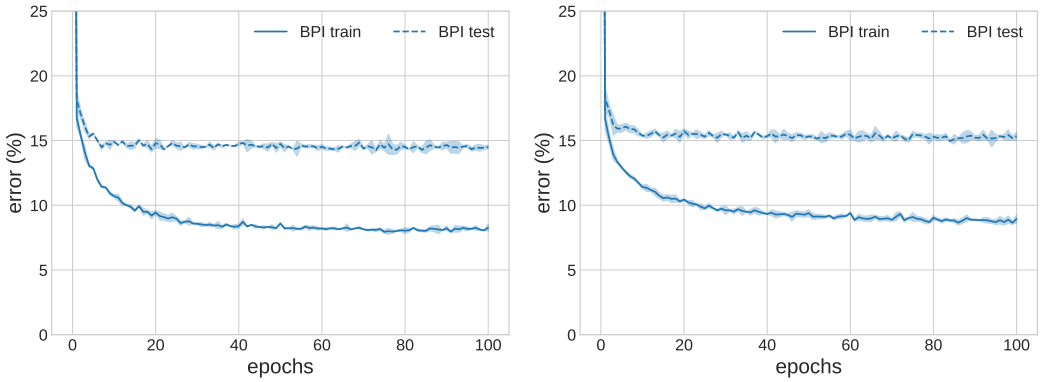

Figure 4: MLP with 2 hidden layers with 101 hidden units each, batch-size 128 on the Fashion-MNIST dataset. In the first two layers we use the BPI equations, while in the last layer the ArgMax ones. (Left) ArgMax layer first version; (Right) ArgMax layer second version. Even if it is possible to reach similar accuracies with the two versions, we decide to use the first version of ArgMax as it is simpler to use.

A.5.2 Approach 2: Jensen again

A further simplification is obtained by applying Jensen inequality again to (76) but in the opposite direction, therefore foregoing the possibility of having a bound. We have the new effective free energy

$$\tilde{\phi}_{k^*} = \sum_{k\neq k^*} \log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} H\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}}\right) \tag{78}$$

$$= \sum_{k\neq k^*} \log H\left(-\frac{\omega_{k^*} - \omega_k}{\sqrt{V_k + V_{k^*}}}\right) \tag{79}$$

This gives, for $k \neq k^*$

$$\partial_{\omega_k}\tilde{\phi}_{k^*} = \begin{cases} -\frac{1}{\sqrt{V_k+V_{k^*}}} GH\left(-\frac{\omega_{k^*}-\omega_k}{\sqrt{V_k+V_{k^*}}}\right) & k \neq k^* \\ \sum_{k'\neq k^*} \frac{1}{\sqrt{V_{k'}+V_{k^*}}} GH\left(-\frac{\omega_{k^*}-\omega_{k'}}{\sqrt{V_{k'}+V_{k^*}}}\right) & k = k^* \end{cases} \tag{80}$$

Notice that $\partial_{\omega_{k^*}}\tilde{\phi}_{k^*} = -\sum_{k\neq k^*} \partial_{\omega_k}\tilde{\phi}_{k^*}$. In last formulas we used the definition

$$GH(x) = \frac{G(x)}{H(x)} = \frac{\sqrt{2/\pi}}{\text{erfcx}(x/2)} \tag{81}$$

We show in Fig. 4 the negligible effect of using the two ArgMax versions when using BPI on the layers before the last one (on which we apply ArgMax).

B Numerical details

B.1 Parameters

B.1.1 Hyper-parameters of the BP-based scheme

We include here a complete list of the hyper-parameters present in the BP-based algorithms. Notice that, like in the SGD type of algorithms, many of them can be fixed or it is possible to find a

prescription for their value that works in most cases. However, we expect future research to find even more effective values of the hyper-parameters, in the same way it has been done for SGD. These hyper-parameters are: the min-batch size bs; the parameter ρ (that has to be tuned similarly to the learning rate in SGD); the damping parameter ψ (that performs a running smoothing on the BP fields along the dynamics by adding a fraction of the field at the previous iteration); the initialization coefficient ϵ (governing the bimodality of the initial distribution of the magnetizations, see Fig. 5); the number of internal steps of reinforcement $\tau_{\max}$ and the associated intensity of the internal reinforcement r. The performances of BP-based algorithms are robust in a reasonable range of these hyper-parameters.

Notice that among these parameters, the BP dynamics at each layer is mostly sensitive to ρ and ψ, so that in general we consider them layer-dependent. See Sec. B.2 for details on the effect of these parameters on the learning dynamics and on layer polarization (i.e. how the BP dynamics tends to bias the weights towards a single point-wise configuration with high probability). Unless otherwise stated we fix some of the hyper-parameters, in particular: $bs = 128$ (results are consistent with other values of the batch-size, from $bs = 1$ up to $bs = 1024$ in our experiments), $\epsilon = 1.0$, $\tau_{\max} = 1$, $r = 0$.

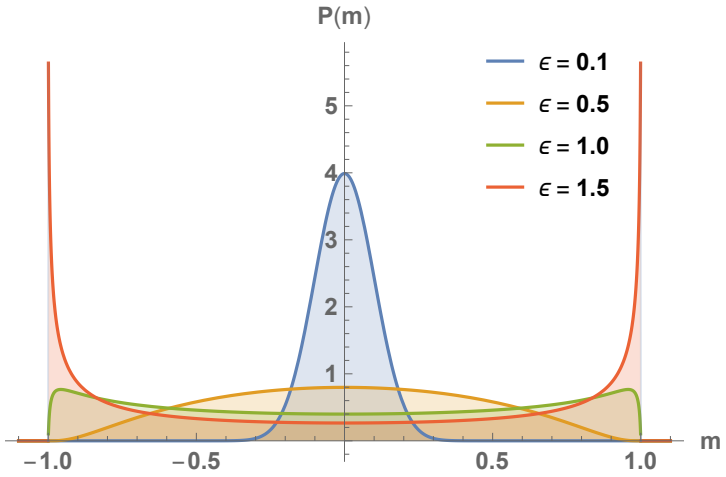

Figure 5: Initial distribution of the magnetizations varying the parameter ϵ. The initial distribution is more concentrated around ± 1 as ϵ increases (i.e. it is more bimodal and the initial configuration is more polarized).

The parameter ϵ governs the initial magnetizations m through the following equation (see Fig. 5):

$$H \sim \epsilon \mathcal{N}(0,1) \tag{82}$$

$$m = \tanh(H) \tag{83}$$

B.1.2 Architectures

In the experiments in which we vary the architecture (see Sec. 4.1), all simulations of the BP-based algorithms use a number of internal reinforcement iterations $\tau_{\max} = 1$. Learning is performed on the totality of the training dataset, the batchisize is $bs = 128$, the initialization coefficient is $\epsilon = 1.0$.

For all architectures and all BP approximations, we use $\psi = 0.8$ for each layer, apart for the 501-501-501 MLP in which we use $\psi = (0.1, 0.1, 0.1, 0.9)$. Concerning the parameter ρ, we use $\rho = 0.9$ on the last layer for all architectures and BP approximations. On the other layers we use: for the 101-101 MLP and for all BP approximations $\rho = 1.0001$; for the 501-501 MLP $\rho = 1.0001$ for all BP approximations; for the 101-101-101 MLP, for BPI and AMP $\rho = 1.0$ while $\rho = 1.001$ for MF; For the 501-501-501 MLP $\rho = 1.0001$ for all BP approximations. For the BinaryNet simulations, the learning rate is $lr = 10.0$ for all MLP architectures, giving the better performance among the learning rates we have tested, $lr = 100, 10, 1, 0.1, 0.001$.

We notice that while we need some tuning of the hyper-parameters to reach the performances of BinaryNet, it is possible to fix them across datasets and architectures (e.g. $\rho = 1$ and $\psi = 0.8$ on each layer) without in general losing more than 20% (relative) of the generalization performances, demonstrating that the BP-based algorithms are effective for learning also in this case. In particular, for the continual learning task (see Sec. 4.4) we fixed $\rho = 1$ and $\psi = 0.8$ on each layer as we empirically observed that polarizing the last layer helps mitigating the forgetting while leaving the single-task performances almost unchanged.

B.1.3 VARYING THE DATASET

When varying the dataset (see Sec. 4.2), all simulation of the BP-based algorithms use a number of internal reinforcement iterations $\tau_{\max} = 1$. For binary classification, for all datasets (MNIST, FashionMNIST, CIFAR-10) and all algorithms (BPI, AMP, MF) we use $\rho = (1.0001, 1.0001, 0.9)$, and $\psi = 0.8$ for each layer (except for AMP on MNIST and Fashion-MNIST where we choose $\psi_3 = 1.0 - 10^{-6}$ on the last layer for stability reasons). For multi-class classification, for all datasets and all algorithms we use $\psi = 0.8$ for each layer. For all datasets and BPI and AMP we use $\rho = (1.0, 1.0, 0.9)$, while for all datasets and MF we use $\rho = (1.0001, 1.0001, 0.9)$.

For the BinaryNet simulations, the learning rate is $lr = 10.0$ (both for binary classification and multi-class classification), giving the better performance among the learning rates we have tested, $lr = 100, 10, 1, 0.1, 0.001$. We report in Tab. 2 we report the final train errors obtained on the different datasets.

Dataset	BinaryNet	BP(I)	AMP	MF
MNIST (2 classes)	0.05 ± 0.05	0.0 ± 0.0	0.010 ± 0.006	0.0 ± 0.0
FashionMNIST (2 classes)	0.3 ± 0.1	0.06 ± 0.01	0.02 ± 0.01	0.09 ± 0.01
CIFAR10 (2 classes)	1.2 ± 0.5	0.37 ± 0.01	0.6 ± 0.2	0.9 ± 0.2
MNIST	0.09 ± 0.01	0.03 ± 0.01	0.06 ± 0.01	0.03 ± 0.01
FashionMNIST	4.0 ± 0.5	0.96 ± 0.02	1.2 ± 0.1	2.5 ± 0.2
CIFAR10	13.0 ± 0.9	1.8 ± 0.1	2.3 ± 0.2	9.2 ± 0.5

Table 2: Train error (%) on Fashion-MNIST of a multiplayer perceptron with 2 hidden layers of 501 units each for binary-SGD (baseline), BPI, AMP and MF. All algorithms are trained with batch-size 128 and for 100 epochs. Mean and standard deviations are calculated over 5 random initializations.

B.1.4 SGD IMPLEMENTATION (BINARYNET)

We compare the BP-based algorithms with SGD training for neural networks with binary weights and activations as introduced in BinaryNet (Hubara et al., 2016). This procedure consists in keeping a continuous version of the parameters w which is updated with the SGD rule, with the gradient calculated on the binarized configuration $w_b = \text{sign}(w)$. At inference time the forward pass is calculated with the parameters w_b. The backward pass with binary activations is done with the so called *straight-through estimator*.

Our implementation presents some differences with respect to the original proposal of the algorithm in Hubara et al. (2016), in order to keep the comparison as fair as possible with the BP-based algorithms, in particular for what concerns the number of parameters. We do not use biases nor batch normalization layers, therefore in order to keep the pre-activations of each hidden layer normalized we rescale them by $\frac{1}{\sqrt{N}}$ where N is the size of the previous layer (or the input size in the case of the pre-activations afferent to the first hidden layer). The standard SGD update rule is applied (instead of Adam), and we use the binary cross-entropy loss. Clipping of the continuous configuration w in $[-1, 1]$ is applied. We use Xavier initialization (Glorot & Bengio, 2010) for the continuous weights.

B.2 UNIT POLARIZATION AND OVERLAPS

We define the self-overlap or polarization of a given unit a as $q_0^a = \frac{1}{N}\sum_i (w_i^a)^2$, where N is the number of parameters of the unit. It quantifies how much the unit is polarized towards a unique point-wise binary configuration ($q_0^a = 1$ corresponding to high confidence in a given configurations while $q_0^a = 0$ to low). The overlap between two units a and b (considered in the same layer) is $q_{ab} = \frac{1}{N}\sum w_i^a w_i^b$. The number of parameters N is the same for units belonging to the same fully

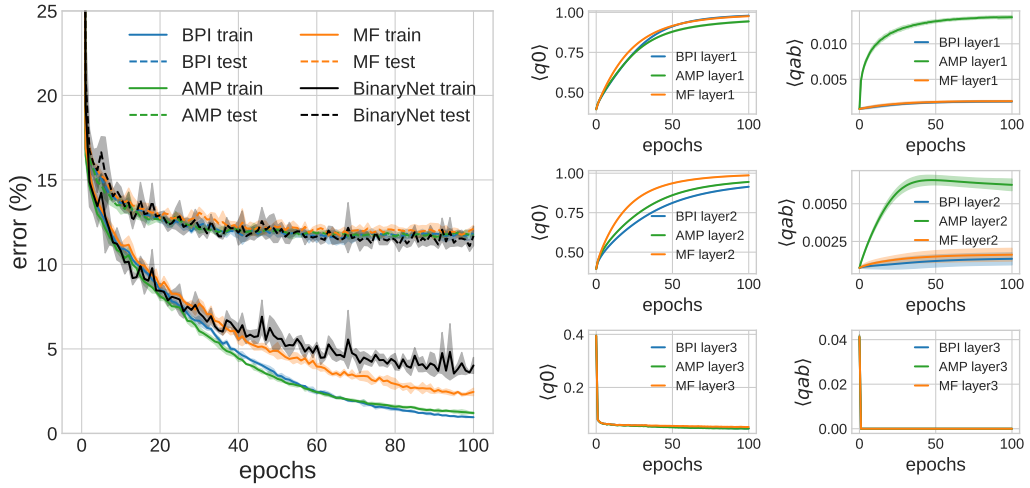

Figure 6: (Right panels) Polarizations $\langle q_0 \rangle$ and overlaps $\langle q_{ab} \langle$ on each layer of a MLP with 2 hidden layers of 501 units on the Fashion-MNIST dataset (multi-class), the batch-size is $bs = 128$. (Right) Corresponding train and test error curves.

connected layer. We denote by $\langle q_0 \rangle = \frac{1}{N_{out}} \sum_{a=1}^{N_{out}} q_0^a$ and $\langle q_{ab} \rangle = \frac{1}{N_{out}} \sum_{a<b}^{N_{out}} q_{ab}$ the mean polarization and mean overlap in a given layer (where N_{out} is the number of units in the layer).

The parameters ρ and ψ govern the dynamical evolution during training of the polarization of each layer. A value $\rho \geq 1$ has the effect to progressively increase the units polarization during training, while $\rho < 1$ disfavours it. The damping ψ which takes values in $[0, 1]$ has the effect to slow the dynamics by an smoothing process (the intensity of which depends on the value of ψ), generically favoring convergence. Given the nature of the updates in Algo.1, each layer presents its own dynamics given the values of ρ_ℓ and ψ_ℓ at that layer, that in general can be different from each other.

We find that it is is beneficial to control the polarization layer-per-layer, see Fig. 6 for typical behavior of the mean polarization and the mean overlaps layer per layer during training. Empirically, we have found that (as we could expect) when training is successful the layers polarize progressively towards $q_0 = 1$, i.e. towards a precise point-wise solution, while the overlaps between units in each hidden layer are such that $q_{ab} \ll 1$ (indicating low symmetry between intra-layer units, as expected for a non-redundant solution). To this aim, in most cases ψ_ℓ can be the same for each layer, while tuning ρ_ℓ for each layer allows to find better generalization performances in some cases (but is not strictly necessary for learning).

In particular, it is possible to use the same value ρ_ℓ for each layer before the last one ($\ell < L$ where L is the number of layers in the network), while we have found that the last layer tends to polarize immediately during the dynamics (probably due to its proximity to the output constraints). Empirically, it is usually beneficial for learning that it does not or only slightly polarize, i.e. $\langle q_0 \rangle \ll 1$ (this can be achieved by imposing $\rho_L < 1$ on the last layer). Learning is anyway possible even when the last layer polarizes towards $\langle q_0 \rangle = 1$ along the dynamics, i.e. by choosing ρ_L sufficiently large.

As a simple general prescription in most experiments we can fix $\psi = 0.8$ and $\rho_L = 0.9$, therefore leaving $\rho_{\ell<L}$ as the only hyper-parameter to be tuned, akin to the learning rate in SGD. Its value has to be very close to 1.0 (a value smaller than 1.0 tends to depolarize the layers (not pointing to a particular point-wise binary configuration), while a value greater than 1.0 tends to make the parameters diverge).

B.3 Varying batch-size: computational performance

In order to compare the time performances of the BP-based algorithms with our implementation of BinaryNet, we report in Fig. 7 the time in seconds taken by a single epoch of each algorithm in function of the batch-size, on a MLP of 2 layers of 501 units on Fashion-MNIST. We test both

algorithms on a NVIDIA GeForce RTX 2080 Ti GPU. Multi-class and binary classification present a very similar time scaling with the batch-size, in both cases comparable with BinaryNet. Let us also notice that BP-based algorithms are able to reach generalization performances comparable to BinaryNet for all the values of the batch-size reported in this section.

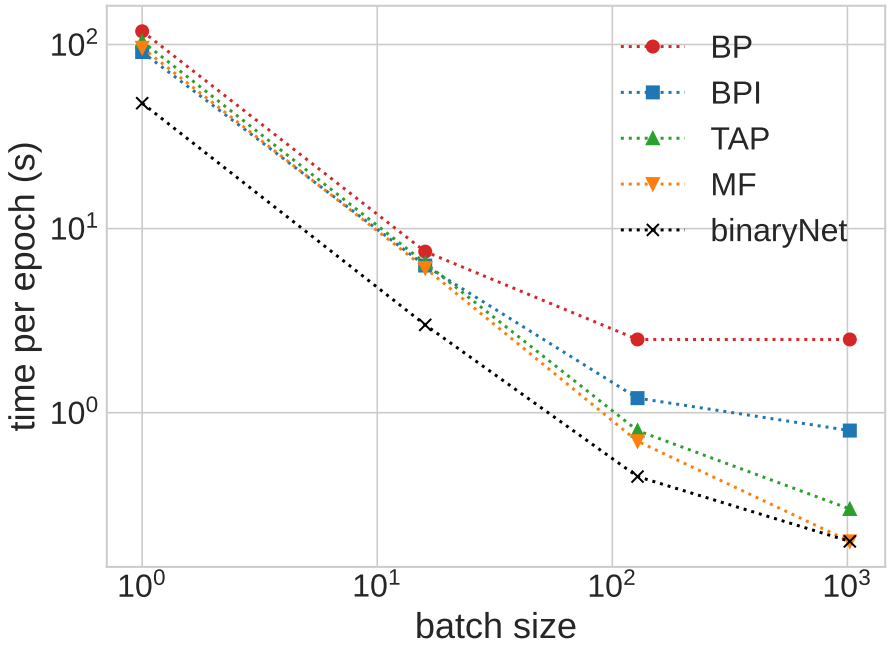

Figure 7: Algorithms time scaling with the batch-size on a MLP with 2 hidden layers with 501 hidden units each on the Fashion-MNIST dataset (multi-class classification). The reported time (in seconds) refers to one epoch for each algorithm.