Deep learning via message passing algorithms based on belief propagation

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 035005 https://doi.org/10.1088/2632-2153/ac7d3b

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
29 January 2022

REVISED
14 April 2022

ACCEPTED FOR PUBLICATION
29 June 2022

PUBLISHED
15 July 2022

PAPER

Deep learning via message passing algorithms based on belief propagation

Carlo Lucibello[1,*], **Fabrizio Pittorino**[1], **Gabriele Perugini**[1,2] **and Riccardo Zecchina**[1]

[1] Institute for Data Science and Analytics, Bocconi University, Milano, Italy
[2] Department of Applied Science and Technology, Politecnico di Torino, Torino, Italy
* Author to whom any correspondence should be addressed.

E-mail: carlo.lucibello@unibocconi.it

Keywords: artificial neural networks, deep learning, approximate message passing, belief propagation, bayesian neural networks

Abstract

Message-passing algorithms based on the belief propagation (BP) equations constitute a well-known distributed computational scheme. They yield exact marginals on tree-like graphical models and have also proven to be effective in many problems defined on loopy graphs, from inference to optimization, from signal processing to clustering. The BP-based schemes are 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 term that biases distributions towards locally entropic solutions. These algorithms are capable of training multi-layer neural networks with performance comparable to SGD heuristics in a diverse set of experiments on natural datasets including multi-class image classification and continual learning, while being capable of yielding improved performances on sparse networks. Furthermore, they allow to make approximate Bayesian predictions that have higher accuracy than point-wise ones.

1. Introduction

Belief Propagation is a method for computing marginals and entropies in probabilistic inference problems (Bethe 1935, Peierls 1936, Gallager 1962, Pearl 1982). These include optimization problems as well 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 a data dependent loss function. These problems are generally intractable 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 build 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 (Yedidia *et al* 2003), where the Gibbs distribution is naturally factorized, whereas it is approximate for graphs with loops. 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) when also involves disorder averages. 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 asymptotic exact results (Mézard and Montanari 2009). Here we will mostly focus on neural networks with ± 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 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 introduced in Braunstein and 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 create y coupled replicas of the original system. fBP equations are obtained as BP equations for the replicated system. It turns out that the fBP equations are identical to the BP equations for the original system with the only addition of a self-reinforcing term in the message passing scheme. The fBP algorithm can be used as a solver by gradually increasing the effect of the reinforcement: 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, 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 inspired message passing algorithms that are capable of training multi-layer neural networks on real data with generalization performance and computational speed comparable to SGD. This is the first work that shows that learning by message passing in deep neural networks 1) is possible and 2) is a viable alternative to SGD, showing competitive performance with common gradient descent methods. Our version of fBP adds the reinforcement term at each mini-batch step in what we call the Posterior-as-Prior (PasP) rule. Furthermore, using the message-passing algorithm not as a solver but as an estimator of marginals allows us to make locally Bayesian predictions, averaging the predictions over the approximate posterior. 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 learning tasks and on sparse networks as well.

Being amenable to analytical description, message passing algorithms are used as powerful theoretical tool in many problems of interest in inference, optimization, and machine learning. While our work aims at extending the range of practical applications of message passing to deep networks, we believe one of its main contributions is paving the way towards novel theoretical methods for the investigation of neural networks. We also remark that our PasP update scheme is of independent interest and can be combined with different posterior approximation techniques.

The paper is structured as follows: in section 2 we give a brief review of some related works. In section 3 we provide a detailed description of the message-passing equations and of the high level structure of the algorithms. In section 4 we compare the performance of the message passing algorithms versus SGD based approaches in different learning settings.

2. Related works

The literature on message passing algorithms is extensive, we refer to Mézard and Montanari (2009) and Zdeborová and 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). Barbier *et al* (2019) rigorously show that message passing

algorithms in generalized linear models perform asymptotically exact inference under some statistical assumptions. Dictionary learning and matrix factorization are harder problems closely related to deep network learning problems, in particular to the modelling of a single intermediate layer. They have been approached using message passing in Kabashima *et al* (2016) and Parker *et al* (2014), although the resulting predictions are found to be asymptotically inexact (Maillard *et al* 2021). The same problem is faced by the message passing algorithm recently proposed for a multi-layer matrix factorization scenario (Zou *et al* 2021a). Unfortunately, our framework as well does not yield asymptotic exact predictions. Nonetheless, it gives a message passing heuristic that for the first time is able to train deep neural networks on natural datasets, therefore sets a reference for the algorithmic applications of this research line.

Message passing schemes dealing with multi-layer problems and displaying similar equations have appeared in the context of inference problems: (Manoel *et al* 2017, Fletcher *et al* 2018) deal with reconstructing a signal from multi-layered non-linear measurements; (Gabrie *et al* 2019) models priors with untrained networks. An online mini-batch approximate message passing algorithm has been introduced in Manoel *et al* (2017) in the context of inference in generalized linear models. Kabashima *et al* (2016), Aubin *et al* (2021)discuss dictionary learning and matrix factorization problems, which could be interesting applications for variants of our algorithm where theoretical analysis can be pushed further. Parker *et al* (2013), Zou *et al* (2021a)is the work that is most closely related to ours. It defines a message passing scheme for solving multi-layer matrix factorization problems. Minor modifications of that algorithm accounting for the supervised learning setting and its combination with our PasP update scheme across mini-batches would lead to our proposed algorithm. None of these approaches aims at multi-layer learning settings and has been shown to be able to optimize a multi-layer neural network with good generalization performance.

A few papers advocate the success of SGD to the geometrical structure (smoothness and flatness) of the loss landscape in neural networks (Baldassi *et al* 2015, Chaudhari *et al* 2017, Garipov *et al* 2018, Li *et al* 2018, Feng and Tu 2021, Pittorino *et al* 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 and Adams 2015, Wu *et al* 2018). Recent work extends BP by combining it with graph neural networks (Kuck *et al* 2020, Satorras and Welling 2021). Finally, some work in computational neuroscience shows similarities to our approach (Rao 2007).

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 $\mathcal{W} = \{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\}_{\ell=0}^{L}$. 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 just encodes an element-wise activation function. The probability of output y given an input $\boldsymbol{x}$ is then given by:

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

where for convenience we defined $\boldsymbol{x}^0 = \boldsymbol{x}$ and $\boldsymbol{x}^{L+1} = y$. In a Bayesian framework, given a training set $D = \{(\boldsymbol{x}_n, y_n)\}_n$ and a prior distribution over the weights $q_\theta(\mathcal{W})$ 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 the assignment $\propto$ denotes equality up to a normalization factor. Using the posterior one can compute the Bayesian prediction $P(y\,|\,\boldsymbol{x},D,\theta) = \int d\mathcal{W}\, P(y\,|\,\boldsymbol{x},\mathcal{W})\, P(\mathcal{W}\,|\,D,\theta)$ for a new data-point $\boldsymbol{x}$. 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 need a formalism that allows for mini-batch training to speed-up the computation and deal with large datasets. Therefore, we devise an update scheme that we call Posterior-as-Prior (PasP), where we evolve the parameters θ^t of a distribution $q_{\theta^t}(\mathcal{W})$ computed as an approximate mini-batch posterior, in such

a way that the outcome of the previous iteration becomes the prior in the following step. In the PasP scheme, θ^t retains the memory of past observations. We also add an exponential factor ρ, that we typically set close to 1, tuning the forgetting rate and playing a role similar to the learning rate in SGD. 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

$$q_{\theta^{t+1}}(\mathcal{W}) \approx [P(\mathcal{W} \,|\, \boldsymbol{y}^t, \boldsymbol{X}^t, \theta^t)]^{\rho}, \tag{3}$$

where $\approx$ denotes approximate equality and up to a normalization factor. A first approximation may be needed in the computation of the mini-batch posterior, a second to project the approximate posterior onto the distribution manifold spanned by θ (Minka 2001). In practice, we will consider factorized approximate posteriors, although equation (3) generically allows for more refined approximations.

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). 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 and Zecchina 2006). Most importantly, it is related to the flat-minima discovery heuristic known as focusing BP (Baldassi *et al* 2016a) and discussed in the introduction. A different prior updating mechanism which can be understood as empirical Bayes has been used in Baldassi *et al* (2016b) instead.

3.2. Inner message passing loop

While the PasP rule takes care of the reinforcement heuristic across mini-batches, we compute the mini-batch posterior in equation (3) using message passing approaches derived from Belief Propagation. BP is an iterative scheme for computing marginals and entropies of statistical models (Mézard and Montanari 2009). It is most conveniently expressed on factor graphs, that is bipartite graphs where the two sets of nodes are called variable nodes and factor nodes. They respectively represent the variables involved in the statistical model and their interactions. Message from factor nodes to variable nodes and viceversa are exchanged along the edges of the factor graph for a certain number of BP iterations or until a fixed point is reached. Using fixed points messages one is able to compute the variables marginals (see appendix A.2 for a more in depth discussion on the relation between messages and marginals). The factor graph for $P(\mathcal{W} \,|\, \boldsymbol{X}^t, \boldsymbol{y}^t, \theta^t)$ can be derived from equation (2), with the following additional specifications. For simplicity, we will ignore the bias term in each layer. We assume factorized $q_{\theta^t}(\mathcal{W})$, each factor parameterized by its first two moments. In what follows, we drop the PasP iteration index t. For each example $(\boldsymbol{x}_n, y_n)$ in the mini-batch, we introduce the auxiliary variables $\boldsymbol{x}_n^\ell, \ell = 1, \dots, L$, representing the layers' activations. For each example, each neuron in the network contributes a factor node to the factor graph. The scalar components of the weight matrices and the activation vectors become variable nodes.

Given a mini-batch $\mathcal{B} = \{(\boldsymbol{x}_n, y_n)\}_n$, the factor graph defined by equations (1)–(3) is explicitly written as:

$$P(\mathcal{W}, \boldsymbol{x}^{1:L} \,|\, \mathcal{B}, \theta) \propto \prod_{\ell=0}^{L} \prod_{k,n} P^{\ell+1}\left(x_{kn}^{\ell+1} \,\middle|\, \sum_i W_{ki}^\ell x_{in}^\ell\right) \prod_{k,i,\ell} q_\theta(W_{ki}^\ell), \tag{4}$$

where $\boldsymbol{x}_n^0 = \boldsymbol{x}_n$, $\boldsymbol{x}_n^{L+1} = y_n$. This construction is presented in appendix A, where we also derive the message update rules on the factor graph. We give a pictorial representation of the factor graph in figure 1.

The factor graph thus defined is extremely loopy and straightforward iteration of BP has convergence issues. Moreover, in presence of a homogeneous prior over the weights, the neuron permutation symmetry in each hidden layer induces a strongly attractive symmetric fixed point that hinders learning. We work around these issues by breaking the symmetry at time $t = 0$ with an inhomogeneous prior. In our experiments a little initial heterogeneity is sufficient to obtain specialized neurons at each following time step. Additionally, we do not require message passing convergence in the inner loop (see algorithm 1) but perform one or a few iterations for each θ update. We also include an inertia term commonly called damping factor in the message updates (see appendix B.2). As we shall discuss, these simple rules suffice to train deep networks by message passing.

For the inner loop we adapt to deep neural networks four different message passing algorithms, all of which are well known to the literature although derived in simpler settings: Belief Propagation (BP), BP-Inspired (BPI) message passing, mean-field (MF), and approximate message passing (AMP). The last three algorithms can be considered approximations of the first one. In the following paragraphs we will discuss their common traits, present the BP updates as an example, and refer to appendix A for an in-depth exposition. For all algorithms, 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.

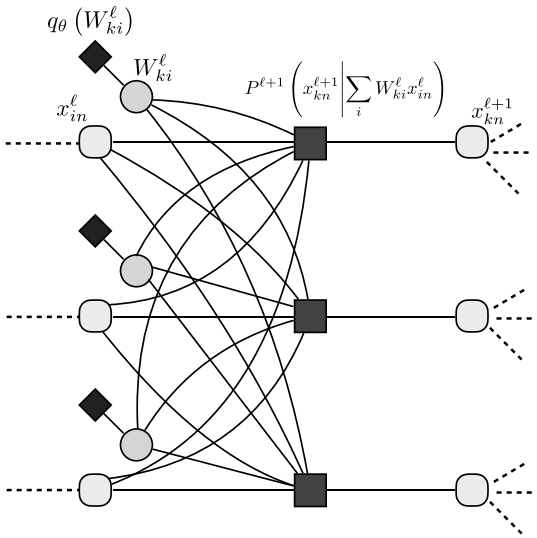

Figure 1. Pictorial representation of the factor graph expressed by equation (4). Dark nodes represent factor nodes corresponding to neurons' activation function (we have such set for each example n) and to the weights' priors $q_\theta(W)$. Light-colored nodes represent variable nodes corresponding to the activations' outputs x and the weights W. Messages are exchanged between variables and factors in both directions along the lines connecting them (see appendix A.2 for a formal discussion).

Algorithm 1: BP for deep neural networks

```
// Message passing used in the PasP equation (3) to approximate.
// the mini-batch posterior.
// Here we specifically refer to BP updates.
// BPI, MF, and AMP updates take the same form but using
// the rules in appendix A.4, A.5, and A.7 respectively
Initialize messages.
for τ = 1, ... τ_max do
    // Forward pass
    for l = 0, ..., L do
        compute x̂^ℓ, Δ^ℓ using (7, 8)
        compute m^ℓ, σ^ℓ using (9, 10)
        compute V^ℓ, ω^ℓ using (11, 12)
    // Backward pass
    for l = L, ..., 0 do
        compute g^ℓ, Γ^ℓ using (13, 14)
        compute A^ℓ, B^ℓ using (15, 16)
        compute G^ℓ, H^ℓ using (13, 18)
```

3.2.1. Meaning of messages

All the messages involved in the message passing can be understood in terms of marginals.

Of particular relevance are m_{ki}^ℓ and σ_{ki}^ℓ, denoting the mean and variance of the weights W_{ki}^ℓ. The quantities $\hat{x}_{in}^\ell$ and Δ_{in}^ℓ instead denote the mean and variance of the ith neuron's activation in layer ℓ for a given input $\boldsymbol{x}_n$.

3.2.2. Scalar free energies

All message passing schemes are conveniently expressed in terms of two functions that can be understood as effective free energies (Zdeborová and Krzakala 2016), i.e. logarithms of normalization factors (partition functions), corresponding to a single neuron and a single weight respectively :

$$\varphi^\ell(B,A,\omega,V) = \log \int \mathrm{d}x\,\mathrm{d}z\, e^{-\frac{1}{2}Ax^2+Bx} P^\ell(x|z)\, e^{-\frac{(\omega-z)^2}{2V}} \qquad \ell = 1,\dots,L, \tag{5}$$

$$\psi(H,G,\theta) = \log \int \mathrm{d}w\, e^{-\frac{1}{2}G^2w^2+Hw} q_\theta(w). \tag{6}$$

Notice that for common deterministic activations such as ReLU and *sign*, the function φ has analytic and smooth expressions (see appendix A.8). The same holds for the function ψ when $q_\theta(w)$ is Gaussian

(continuous weights) or a mixture of atoms (discrete weights). At the last layer we impose $P^{L+1}(y|z) = \mathbb{I}(y = sign(z))$ in binary classification tasks and $P^{L+1}(y|\boldsymbol{z}) = \mathbb{I}(y = arg\,max(\boldsymbol{z}))$ in multi-class classification (see appendix A.9). While in our experiments we use hard constraints for the final output, 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.

3.2.3. 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's parameters θ^{t+1} from the posterior given by the message passing.

3.2.4. BP forward pass

After initialization of the messages at time $\tau = 0$, for each following time we propagate a set of message from the first to the last layer and then another set from the last to the first. For an intermediate layer ℓ the forward pass reads:

$$\hat{x}^{\ell,\tau}_{in\to k} = \partial_B \varphi^\ell \left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in} \right) \tag{7}$$

$$\Delta^{\ell,\tau}_{in} = \partial^2_B \varphi^\ell \left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in} \right) \tag{8}$$

$$m^{\ell,\tau}_{ki\to n} = \partial_H \psi(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^\ell_{ki}) \tag{9}$$

$$\sigma^{\ell,\tau}_{ki} = \partial^2_H \psi(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^\ell_{ki}) \tag{10}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki\to n} \right)^2 \Delta^{\ell,\tau}_{in} + \sigma^{\ell,\tau}_{ki} (\hat{x}^{\ell,\tau}_{in\to k})^2 + \sigma^{\ell,\tau}_{ki} \Delta^{\ell,\tau}_{in} \right) \tag{11}$$

$$\omega^{\ell,\tau}_{kn\to i} = \sum_{i'\neq i} m^{\ell,\tau}_{ki'\to n} \hat{x}^{\ell,\tau}_{i'n\to k}. \tag{12}$$

The equations for the first layer differ slightly and in an intuitive way from the ones above (see appendix A.3).

3.2.5. BP backward pass

The backward pass updates a set of messages from the last to the first layer:

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1} \left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn} \right) \tag{13}$$

$$\Gamma^{\ell,\tau}_{kn} = -\partial^2_\omega \varphi^{\ell+1} \left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn}, V^{\ell,\tau}_{kn} \right) \tag{14}$$

$$A^{\ell,\tau}_{in} = \sum_k \left(\left(m^{\ell,\tau}_{ki\to n} \right)^2 + \sigma^{\ell,\tau}_{ki} \right) \Gamma^{\ell,\tau}_{kn} - \sigma^{\ell,\tau}_{ki} \left(g^{\ell,\tau}_{kn\to i} \right)^2 \tag{15}$$

$$B^{\ell,\tau}_{in\to k} = \sum_{k'\neq k} m^{\ell,\tau}_{k'i\to n} g^{\ell,\tau}_{k'n\to i} \tag{16}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left(\left(\hat{x}^{\ell,\tau}_{in\to k} \right)^2 + \Delta^{\ell,\tau}_{in} \right) \Gamma^{\ell,\tau}_{kn} - \Delta^{\ell,\tau}_{in} \left(g^{\ell,\tau}_{kn\to i} \right)^2 \tag{17}$$

$$H^{\ell,\tau}_{ki\to n} = \sum_{n'\neq n} \hat{x}^{\ell,\tau}_{in'\to k} g^{\ell,\tau}_{kn'\to i}. \tag{18}$$

As with the forward pass, we add the caveat that for the last layer the equations are slightly different from the ones above.

3.2.6. Computational complexity

The message passing equations boil down to element-wise operations and tensor contractions that we easily implement using the GPU friendly julia library Tullio.jl (Abbott *et al* 2021). 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 $O(N^2B)$ 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.8). 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**.

4. Numerical results

We implement our message passing algorithms on neural networks with continuous and binary weights and with binary activations. In our experiments we fix $\tau_{\max} = 1$. We typically do not observe an increase in performance 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 numerical results we will only investigate BP.

We compare our algorithms with a SGD-based algorithm adapted to binary architectures (Hubara *et al* 2016) which we call BinaryNet along the paper (see appendix B.5 for details). Comparison of Bayesian predictions are with the gradient-based expectation backpropagation (EBP) algorithm (Soudry *et al* 2014), also able to deal with discrete weights and activations. In all architectures we avoid the use of bias terms and batch-normalization layers.

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 multi-class 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 message passing algorithms are able to solve these optimization problems with generalization performance comparable to or better than SGD-based algorithms. Some of the message passing algorithms (BP and AMP in particular) need fewer epochs to achieve low error than the ones required by SGD-based algorithms, even if adaptive methods like Adam are considered. Timings of our GPU implementations of message passing algorithms are competitive with SGD (see appendix B.8).

4.1. Experiments across architectures

We select a specific task, multi-class classification on Fashion-MNIST, and we compare the message passing algorithms with BinaryNet for different choices of the architecture (i.e. we vary the number and the size of the hidden layers). In figure 2 (left) we present the learning curves for a MLP with 3 hidden layers with 501 units with binary weights and activations. Similar results hold in our experiments with 2 or 3 hidden layers of 101, 501 or 1001 units and with batch sizes from 1 to from 1024. The parameters used in our simulations are reported in appendix B.3. Results on networks with continuous weights can be found in figure 3 (right).

4.2. Sparse layers

Since the BP algorithm has notoriously been successful on sparse graphs, we perform a straightforward implementation of pruning at initialization, i.e. we impose a random boolean mask on the weights that we keep fixed along the training. We call *sparsity* the fraction of zeroed weights. This kind of non-adaptive pruning is known to largely hinder learning (Frankle *et al* 2021, Sung *et al* 2021). In the right panel of figure 2, we report results on sparse binary networks in which we train a MLP with 2 hidden layers of 101 units on the MNIST dataset. For reference, results on pruning quantized/binary networks can be found in Han *et al* (2016), Ardakani *et al* (2017), Tung and Mori (2018), Diffenderfer and Kailkhura (2021). Experimenting with sparsity up to 90%, we observe that BP and MF perform better than BinaryNet. AMP struggles behind BinaryNet instead.

4.3. Experiments across datasets

We now fix the architecture, a MLP with 2 hidden layers of 501 neurons each with binary weights and activations. 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, in both the multi-class and binary classification tasks. In table 1 we report the final test errors obtained by the message passing algorithms compared to the BinaryNet baseline. See appendix B.4 for the corresponding training errors and the parameters used in the simulations. We mention that while the test performance is mostly comparable, the train error tends to be lower for the message passing algorithms.

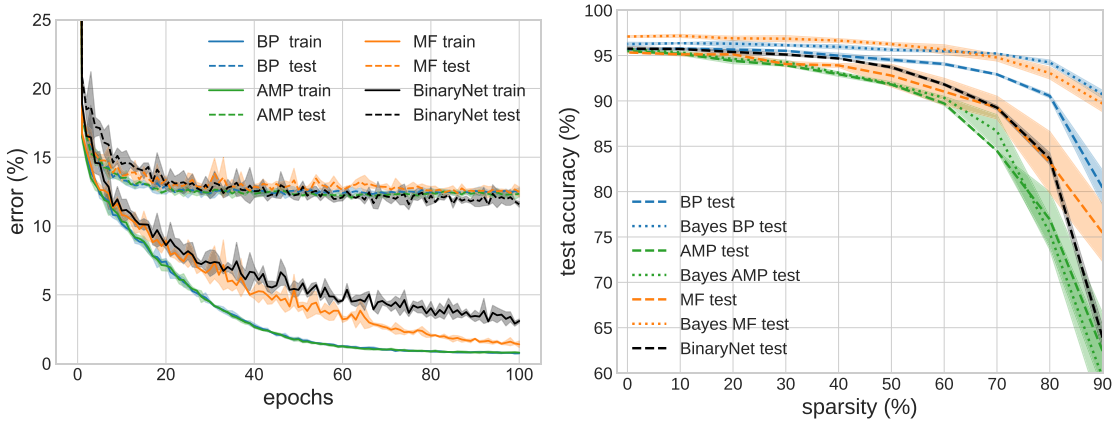

Figure 2. (Left) Training curves of message passing algorithms compared with BinaryNet on the Fashion-MNIST dataset (multi-class classification) with a binary MLP with 3 hidden layers of 501 units. (Right) Final test accuracy when varying the layer's sparsity in a binary MLP with 2 hidden layers of 101 units trained on the MNIST dataset (multi-class). In both panels the batch-size is 128 and curves are averaged over 5 realizations of the initial conditions (and sparsity pattern in the right panel).

Table 1. Test error (%) on MNIST, Fashion-MNIST and CIFAR-10 (both binary and multiclass classification) of various algorithms on a MLP with 2 hidden layers of 501 units, binary weights and activations. All algorithms are trained with batch-size 128 and for 100 epochs. Mean and standard deviations are calculated over 5 random initializations.

Dataset	BinaryNet	BP	AMP	MF
MNIST (2 classes)	1.3 ± 0.1	1.4 ± 0.2	1.4 ± 0.1	1.3 ± 0.2
Fashion-MNIST (2 classes)	2.4 ± 0.1	2.3 ± 0.1	2.4 ± 0.1	2.3 ± 0.1
CIFAR-10 (2 classes)	30.0 ± 0.3	31.4 ± 0.1	31.1 ± 0.3	31.1 ± 0.4
MNIST	2.2 ± 0.1	2.6 ± 0.1	2.6 ± 0.1	2.3 ± 0.1
Fashion-MNIST	12.0 ± 0.6	11.8 ± 0.3	11.9 ± 0.2	12.1 ± 0.2
CIFAR-10	59.0 ± 0.7	58.7 ± 0.3	58.5 ± 0.2	60.4 ± 1.1

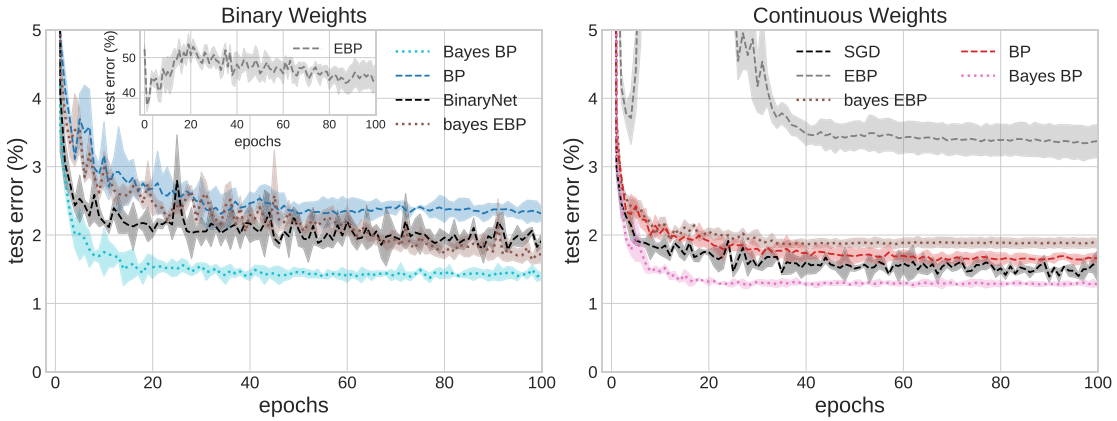

Figure 3. (Left) Test error curves for Bayesian and point-wise predictions for a MLP with 2 hidden layers of 101 units on the 2-classes MNIST dataset. We report the results for (Left) binary and (Right) continuous weights. In both cases, we compare SGD, BP (point-wise and Bayesian) and EBP (point-wise and Bayesian). See appendix B.3 for details.

4.4. Locally Bayesian error

The message passing framework used as an estimator of the mini-batch posterior marginals allows us to perform approximate Bayesian prediction, i.e. averaging the pointwise predictions over the approximate posterior. We observe better generalization error from Bayesian predictions compared to point-wise ones, showing that the marginals retain useful information. However, we roughly estimate the marginals with the PasP 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 the local energy measure performed in section 4.5), the Bayesian error we compute can be considered as a local approximation of the full one. We report results for binary classification on the MNIST

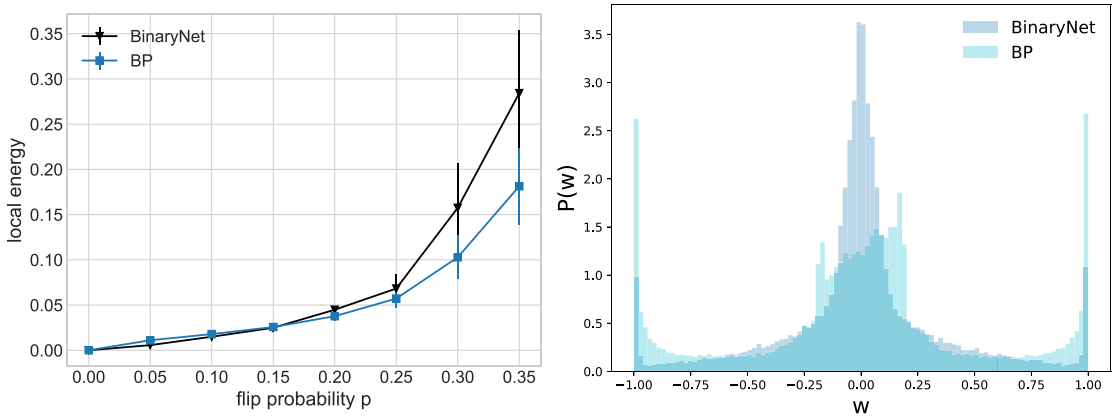

Figure 4. Left panel: Local energy curve of the point-wise configuration found by the BP algorithm compared with BinaryNet on a MLP with 2 hidden layers of 101 units on the 2-class MNIST dataset. Right panel: comparison of the weight distributions in the first layer found by Bayesian BP and BinaryNet (continuous accumulated weights for BinaryNet, magnetizations in the BP case).

dataset in figure 3, 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 posterior distribution does not concentrate too much, otherwise the Bayesian prediction will converge to the prediction of a single configuration.

In figure 3. we also perform a comparison of BP (point-wise and Bayesian) with SGD and another algorithm able to perform Bayesian predictions, Expectation Backpropagation (Soudry *et al* 2014) see appendix B.6 for implementation details.

4.5. 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:

$$\delta E_{\text{train}}(\boldsymbol{w}, p) = \mathbb{E}_{\boldsymbol{z}} E_{\text{train}}(\boldsymbol{w} \odot \boldsymbol{z}) - E_{\text{train}}(\boldsymbol{w}), \tag{19}$$

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 figure 4 left panel for BP 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 architectures and datasets.

In addition to the comparison through the local energy, we have also provided a comparison of the different weight distributions found by SGD and Bayesian BP, in order to add insight into the type of solutions that the two algorithms find, see the right panel of figure 4. Analogously to Liu *et al* (2021) (that compares vanilla SGD with Adam) we find that the weight histogram of BP solutions develops more latent real-valued weights with larger absolute values compared to SGD.

4.6. Continual learning

Given the high local entropy (i.e. the flatness) of the solutions found by the BP-based algorithms (see 4.5), we perform additional tests in a classic setting, continual learning, where the possibility of locally rearranging the solutions while keeping low training error can be an advantage. When a deep network is trained sequentially on different tasks, it tends to forget exponentially fast previously seen tasks while learning new ones (McCloskey and Cohen 1989, Robins 1995, Fusi *et al* 2005). Recent work (Feng and Tu 2021) has shown that searching for a flat region in the loss landscape can indeed help to prevent catastrophic forgetting. Several heuristics have been proposed to mitigate the problem (Kirkpatrick *et al* 2017, Zenke *et al* 2017, Aljundi *et al* 2018, Laborieux *et al* 2021) but all require specialized adjustments to the loss or the dynamics.

Here we show instead that our message passing schemes are naturally prone to learn multiple tasks sequentially, mitigating the characteristic memory issues of the gradient-based schemes without the need for

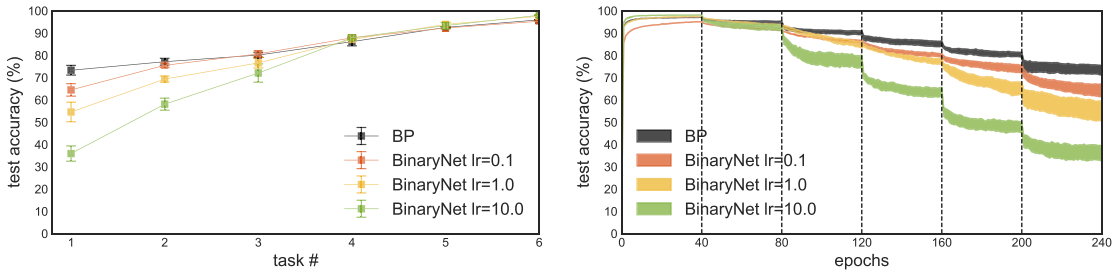

Figure 5. Performance of BP and BinaryNet on the permuted MNIST task (see text) for a two hidden layer network with 2001 units on each layer and binary weights and activations. The model is trained sequentially on 6 different versions of the MNIST dataset (the tasks), where the pixels have been permuted. (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, shaded areas are errors on the mean.

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. We present the results for a two hidden layer neural network with 2001 units on each layer (see appendix B.3 for details). As can be seen in figure 5, at the end of the training the BP algorithm is able to reach good generalization performances on all the tasks. We compared the BP performance with BinaryNet, which already performs better than SGD with continuous weights (see the discussion in Laborieux *et al* 2021). While our BP implementation is not competitive with ad-hoc techniques specifically designed for this problem, it beats non-specialized heuristics. Moreover, we believe that specialized approaches like the one of Laborieux *et al* (2021) can be adapted to message passing as well.

5. Discussion and conclusions

While successful in many fields, message passing algorithms, have notoriously struggled to scale to deep neural networks training problems. Here we have developed a class of fBP-based message passing algorithms and used them within an update scheme, Posterior-as-Prior (PasP), that makes it possible to train deep and wide multilayer perceptrons by message passing.

We performed experiments binary activations and either binary or continuous weights. Future work should try to include different activations, biases, batch-normalization, and convolutional layers as well. 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 methods. Recent developments in message passing algorithms (Rangan *et al* 2019) and related theoretical analysis (Goldt *et al* 2020) could provide fruitful inspirations. While our algorithms can be used for approximate Bayesian inference, exact posterior calculation is still out of reach for message passing approaches and much technical work is needed in that direction. Another relevant line of investigation is to derive state evolution equations (Donoho *et al* 2009) in order to obtain a concise statistical description of the iterations of our algorithm in terms of a few scalar quantities.

Data availability statement

The data that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/ArtLabBocconi/DeepMP.jl. Data will be available from 28 April 2022.

Appendix A. BP-based message passing algorithms

A.1. Preliminary considerations

Given a mini-batch $\mathcal{B} = \{(\boldsymbol{x}_n, y_n)\}_n$, the factor graph defined by equations (1)–(3) is explicitly written as:

$$P(\mathcal{W}, \boldsymbol{x}^{1:L} \,|\, \mathcal{B}, \theta) \propto \prod_{\ell=0}^{L} \prod_{k,n} P^{\ell+1}\left(x_{kn}^{\ell+1} \,\middle|\, \sum_i W_{ki}^{\ell} x_{in}^{\ell}\right) \prod_{k,i,\ell} q_\theta(W_{ki}^{\ell}), \tag{20}$$

where $\boldsymbol{x}_n^0 = \boldsymbol{x}_n$, $\boldsymbol{x}_n^{L+1} = y_n$. The derivation of the BP equations for this model is straightforward albeit lengthy and involved. It is obtained following the steps presented in multiple papers, books, and reviews, see for instance (Mézard and Montanari 2009, Zdeborová and Krzakala 2016, Mézard 2017), although it has not been attempted before in deep neural networks. 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 distributed and therefore parameterized by their mean and variance. This goes by the name of relaxed belied propagation (rBP), just referred to as BP throughout the paper.

We derive the BP equations in A.2 and present them all together in A.3. From BP, we derive other 3 message passing algorithms useful for the deep network training setting, all of which are well known to the literature: BP-Inspired (BPI) message passing A.4, mean-field (MF) A.5, and approximate message passing (AMP) A.7. The AMP derivation is the more involved and given in A.6. In all these cases, message updates can be divided in a forward pass and a 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 present in the main text.

A.1.1. Meaning of messages

All the messages involved in the message passing equations can be understood in terms of cavity marginals or full marginals (as mentioned in the introduction BP is also known as the Cavity Method, see Mézard and Montanari 2009). Of particular relevance are the quantities m_{ki}^ℓ and σ_{ki}^ℓ, denoting the mean and variance of the weights W_{ki}^ℓ. The quantities $\hat{x}_{in}^\ell$ and Δ_{in}^ℓ instead denote mean and variance of the i-th neuron's activation in layer ℓ in correspondence of an input $\boldsymbol{x}_n$.

A.1.2. Scalar free energies

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

$$\varphi^\ell(B, A, \omega, V) = \log \int \mathrm{d}x \mathrm{d}z \, e^{-\frac{1}{2}Ax^2 + Bx} P^\ell(x|z) \, e^{-\frac{(\omega - z)^2}{2V}}, \tag{21}$$

$$\psi(H, G, \theta) = \log \int \mathrm{d}w \, e^{-\frac{1}{2}G^2 w^2 + Hw} q_\theta(w). \tag{22}$$

These free energies will naturally arise in the derivation of the BP equations in appendix A.2. For the last layer, the neuron function has to be slightly modified:

$$\varphi^{L+1}(y, \omega, V) = \log \int \mathrm{d}z \, P^{L+1}(y|z) \, e^{-\frac{(\omega - z)^2}{2V}}. \tag{23}$$

Notice that for common deterministic activations such as ReLU and *sign*, the function φ has analytic and smooth expressions that we give in appendix A.8. 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 $P^{L+1}(y|z) = \mathbb{I}(y = sign(z))$ in binary classification tasks. For multi-class classification instead, we have to adapt the formalism to vectorial pre-activations $\boldsymbol{z}$ and assume $P^{L+1}(y|\boldsymbol{z}) = \mathbb{I}(y = argmax(\boldsymbol{z}))$ (see appendix A.9). While in our experiments we use hard constraints for the final output, therefore solving a constraint satisfaction problem, it would be interesting to also consider generic loss functions. That would require minimal changes to our formalism, but this is beyond the scope of our work.

A.1.3. Binary weights

In our experiments we use ± 1 weights in each layer. Therefore each marginal can be parameterized by a single number and our prior/posterior takes the form:

$$q_\theta(W_{ki}^\ell) \propto e^{\theta_{ki}^\ell W_{ki}^\ell}. \tag{24}$$

The effective free energy function equation (22) becomes:

$$\psi(H, G, \theta_{ki}^\ell) = \log 2 \cosh(H + \theta_{ki}^\ell), \tag{25}$$

and the messages G can be dropped from the message passing.

A.1.4. Start and end of message passing
At the beginning of a new PasP iteration t, we reset the messages to zero 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 iterations.

A.2. Derivation of the BP equations
In order to derive the BP equations, we start with the following portion of the factor graph reported in equation (20) in the main text, describing the contribution of a single data example in the inner loop of the PasP updates:

$$\prod_{\ell=0}^{L}\prod_{k} P^{\ell+1}\left(x_{kn}^{\ell+1}\,\middle|\,\sum_{i} W_{ki}^{\ell}x_{in}^{\ell}\right) \quad \text{where } \boldsymbol{x}_n^0=\boldsymbol{x}_n,\ \boldsymbol{x}_n^{L+1}=y_n. \tag{26}$$

where we recall that the quantity x_{kn}^{ℓ} corresponds to the activation of neuron k in layer ℓ in correspondence of the input example n.

Let us start by analyzing the single factor:

$$P^{\ell+1}\left(x_{kn}^{\ell+1}\,\middle|\,\sum_{i} W_{ki}^{\ell}x_{in}^{\ell}\right). \tag{27}$$

We refer to messages that travel from input to output in the factor graph as *upgoing* or *upwards* messages, while to the ones that travel from output to input as *downgoing* or *backwards* messages.

A.2.1. Factor-to-variable-$\boldsymbol{W}$ messages
The factor-to-variable-W messages read:

$$\hat{\nu}_{kn\to ki}^{\ell+1}(W_{ki}^{\ell}) \propto \int \prod_{i'\neq i} d\nu_{ki'\to n}^{\ell}(W_{ki'}^{\ell}) \prod_{i'} d\nu_{i'n\to k}^{\ell}(x_{i'n}^{\ell})\, d\nu_{\downarrow}(x_{kn}^{\ell+1})\, P^{\ell+1}\left(x_{kn}^{\ell+1}\,\middle|\,\sum_{i'} W_{ki'}^{\ell}x_{i'n}^{\ell}\right), \tag{28}$$

where $\nu_{\downarrow}$ denotes the messages travelling downwards (from output to input) in the factor graph.

We denote the means and variances of the incoming messages respectively with $m_{ki\to n}^{\ell}$, $\hat{x}_{in\to k}^{\ell}$ and $\sigma_{ki\to n}^{\ell}$, $\Delta_{in\to k}^{\ell}$:

$$m_{ki\to n}^{\ell} = \int d\nu_{ki\to n}^{\ell}(W_{ki}^{\ell})\ W_{ki}^{\ell} \tag{29}$$

$$\sigma_{ki\to n}^{\ell} = \int d\nu_{ki\to n}^{\ell}(W_{ki}^{\ell})\ \left(W_{ki}^{\ell}-m_{ki\to n}^{\ell}\right)^2 \tag{30}$$

$$\hat{x}_{in\to k}^{\ell} = \int d\nu_{in\to k}^{\ell}(x_{in}^{\ell})\ x_{in}^{\ell} \tag{31}$$

$$\Delta_{in\to k}^{\ell} = \int d\nu_{in\to k}^{\ell}(x_{in}^{\ell})\ \left(x_{in}^{\ell}-\hat{x}_{in\to k}^{\ell}\right)^2. \tag{32}$$

We now use the central limit theorem to observe that with respect to the incoming messages distributions—assuming independence of these messages—in the large input limit the preactivation is a Gaussian random variable:

$$\sum_{i'\neq i} W_{ki'}^{\ell}x_{i'n}^{\ell} \sim \mathcal{N}(\omega_{kn\to i}^{\ell}, V_{kn\to i}^{\ell}), \tag{33}$$

where:

$$\omega_{kn\to i}^{\ell} = \mathbb{E}_{\nu}\left[\sum_{i'\neq i} W_{ki'}^{\ell}x_{i'n}^{\ell}\right] = \sum_{i'\neq i} m_{ki'\to n}^{\ell}\hat{x}_{i'n\to k}^{\ell} \tag{34}$$

$$V^{\ell}_{kn\to i} = Var_{\nu}\left[\sum_{i'\neq i} W^{\ell}_{ki'}x^{\ell}_{i'n}\right] = \sum_{i'\neq i}\left(\sigma^{\ell}_{ki'\to n}\Delta^{\ell}_{i'n\to k} + \left(m^{\ell}_{ki'\to n}\right)^2\Delta^{\ell}_{i'n\to k} + \sigma^{\ell}_{ki'\to n}\left(\hat{x}^{\ell}_{i'n\to k}\right)^2\right). \tag{35}$$

Therefore we can rewrite the outgoing messages as:

$$\hat{\nu}^{\ell+1}_{kn\to i}(W^{\ell}_{ki}) \propto \int dz\, d\nu^{\ell}_{in\to k}(x^{\ell}_{in})\, d\nu_{\downarrow}(x^{\ell+1}_{kn})\, e^{-\frac{\left(z-\omega_{kn\to i}-W^{\ell}_{ki}x^{\ell}_{in}\right)^2}{2V_{kn\to i}}} P^{\ell+1}\left(x^{\ell+1}_{kn}\,\middle|\, z\right). \tag{36}$$

We now assume $W^{\ell}_{ki}x^{\ell}_{in}$ to be small compared to the other terms. With a second order Taylor expansion we obtain:

$$\hat{\nu}^{\ell}_{kn\to i}(W^{\ell}_{ki}) \propto \int dz\; d\nu_{\downarrow}(x^{\ell+1}_{kn})\, e^{-\frac{(z-\omega_{kn\to i})^2}{2V_{kn\to i}}} P^{\ell+1}\left(x^{\ell+1}_{kn}\,\middle|\, z\right) \times\left(1+\frac{z-\omega_{kn\to i}}{V_{kn\to i}}\hat{x}^{\ell}_{in\to k}W^{\ell}_{ki} + \frac{(z-\omega_{kn\to i})^2 - V_{kn\to i}}{2V_{kn\to i}}\left(\Delta+\left(\hat{x}^{\ell}_{in\to k}\right)^2\right)\left(W^{\ell}_{ki}\right)^2\right). \tag{37}$$

Introducing now the function:

$$\varphi^{\ell}(B,A,\omega,V) = \log\int \mathrm{d}x\,\mathrm{d}z\, e^{-\frac{1}{2}Ax^2+Bx}P^{\ell}(x|z)\, e^{-\frac{(\omega-z)^2}{2V}}, \tag{38}$$

and defining:

$$g^{\ell}_{kn\to i} = \partial_{\omega}\varphi^{\ell+1}(B^{\ell+1},A^{\ell+1},\omega^{\ell}_{kn\to i},V^{\ell}_{kn\to i}), \tag{39}$$

$$\Gamma^{\ell}_{kn\to i} = -\partial^2_{\omega}\varphi^{\ell+1}(B^{\ell+1},A^{\ell+1},\omega^{\ell}_{kn\to i},V^{\ell}_{kn\to i}), \tag{40}$$

the expansion for the log-message reads:

$$\log\hat{\nu}^{\ell}_{kn\to i}(W^{\ell}_{ki}) \approx const + \hat{x}^{\ell}_{in\to k}g^{\ell}_{kn\to i}W^{\ell}_{ki} - \frac{1}{2}\left(\left(\Delta^{\ell}_{in\to k}+\left(\hat{x}^{\ell}_{in\to k}\right)^2\right)\Gamma^{\ell}_{kn\to i} - \Delta^{\ell}_{in\to k}\left(g^{\ell}_{kn\to i}\right)^2\right)\left(W^{\ell}_{ki}\right)^2. \tag{41}$$

*A.2.2. Factor-to-variable-**x** messages*

The derivation of these messages is analogous to the factor-to-variable-W ones in equation (28) just reported. The final result for the log-message is:

$$\log\hat{\nu}^{\ell}_{kn\to i}(x^{\ell}_{in}) \approx const + m^{\ell}_{ki\to n}g^{\ell}_{kn\to i}x^{\ell}_{in} - \frac{1}{2}\left(\left(\sigma^{\ell}_{ki\to n}+\left(m^{\ell}_{ki\to n}\right)^2\right)\Gamma^{\ell}_{kn\to i} - \sigma^{\ell}_{ki\to n}\left(g^{\ell}_{kn\to i}\right)^2\right)\left(x^{\ell}_{in}\right)^2. \tag{42}$$

*A.2.3. Variable-**W**-to-output-factor messages*

The message from variable W^{ℓ}_{ki} to the output factor kn reads:

$$\nu^{\ell}_{ki\to n}(W^{\ell}_{ki}) \propto P^{\ell}_{\theta_{ki}}(W^{\ell}_{ki})e^{\sum_{n'\neq n}\log\hat{\nu}^{\ell}_{kn'\to i}(W^{\ell}_{ki})} \approx P^{\ell}_{\theta_{ki}}(W^{\ell}_{ki})e^{H^{\ell}_{ki\to n}W^{\ell}_{ki}-\frac{1}{2}G^{\ell}_{ki\to n}\left(W^{\ell}_{ki}\right)^2}, \tag{43}$$

where we have defined:

$$H^{\ell}_{ki\to n} = \sum_{n'\neq n}\hat{x}^{\ell}_{in'\to k}g^{\ell}_{kn'\to i} \tag{44}$$

$$G^{\ell}_{ki\to n} = \sum_{n'\neq n}\left(\left(\Delta^{\ell}_{in'\to k}+\left(\hat{x}^{\ell}_{in'\to k}\right)^2\right)\Gamma^{\ell}_{kn'\to i} - \Delta^{\ell}_{in'\to k}\left(g^{\ell}_{kn'\to i}\right)^2\right). \tag{45}$$

Introducing now the effective free energy:

$$\psi(H,G,\theta) = \log \int \mathrm{d}W\, P_\theta^\ell(W)\, e^{HW - \frac{1}{2}GW^2}, \tag{46}$$

we can express the first two cumulants of the message $\nu^\ell_{ki\to n}(W^\ell_{ki})$ as:

$$m^\ell_{ki\to n} = \partial_H \psi(H^\ell_{ki\to n}, G^\ell_{ki\to n}, \theta_{ki}), \tag{47}$$

$$\sigma^\ell_{ki\to n} = \partial_H^2 \psi(H^\ell_{ki\to n}, G^\ell_{ki\to n}, \theta_{ki}). \tag{48}$$

A.2.4. Variable-$\boldsymbol{x}$-to-input-factor messages
We can write the downgoing message as:

$$\begin{aligned}\nu_\downarrow(x^\ell_{in}) &\propto e^{\sum_k \log \hat{\nu}^\ell_{kn\to i}(x^\ell_{in})} \\ &\approx e^{B^\ell_{in}x - \frac{1}{2}A^\ell_{in}x^2},\end{aligned} \tag{49}$$

where:

$$B^\ell_{in} = \sum_n m^\ell_{ki\to n} g^\ell_{kn\to i} \tag{50}$$

$$A^\ell_{in} = \sum_n \left(\left(\sigma^\ell_{ki\to n} + \left(m^\ell_{ki\to n}\right)^2 \right) \Gamma^\ell_{kn\to i} - \sigma^\ell_{ki\to n} \left(g^{\ell+1}_{kn\to i}\right)^2 \right). \tag{51}$$

A.2.5. Variable-$\boldsymbol{x}$-to-output-factor messages
By defining the following cavity quantities:

$$B^\ell_{in\to k} = B^\ell_{in\to k} - m^\ell_{ki\to n} g^\ell_{kn\to i} \tag{52}$$

$$A^\ell_{in\to k} = A^\ell_{in\to k} - \left(\left(\sigma^\ell_{ki\to n} + \left(m^\ell_{ki\to n}\right)^2 \right) \Gamma^\ell_{kn\to i} - \sigma^\ell_{ki\to n} \left(g^\ell_{kn\to i}\right)^2 \right), \tag{53}$$

and the following non-cavity ones:

$$\omega^\ell_{kn} = \sum_i m^\ell_{ki\to n} \hat{x}^\ell_{in\to k} \tag{54}$$

$$V^\ell_{kn} = \sum_i \left(\sigma^\ell_{ki\to n} \Delta^\ell_{in\to k} + \left(m^\ell_{ki\to n}\right)^2 \Delta^\ell_{in\to k} + \sigma^\ell_{ki\to n} \left(\hat{x}^\ell_{i'n\to k}\right)^2 \right), \tag{55}$$

we can express the first 2 cumulants of the upgoing messages as:

$$\hat{x}^\ell_{in\to k} = \partial_B \varphi^\ell(B^\ell_{in\to k}, A^\ell_{in\to k}, \omega^{\ell-1}_{in}, V^{\ell-1}_{in}) \tag{56}$$

$$\Delta^\ell_{in\to k} = \partial_B^2 \varphi^\ell(B^\ell_{in\to k}, A^\ell_{in\to k}, \omega^{\ell-1}_{in}, V^{\ell-1}_{in}). \tag{57}$$

A.2.6. Wrapping it up
Additional but straightforward considerations are required for the final input and output layers ($\ell = 0$ and $\ell = L$ respectively), since they do not receive messages from below and above respectively. In the end, thanks to independence assumptions and the central limit theorem that we used throughout the derivations, we arrive to a closed set of equations involving the means and the variances (or otherwise the corresponding natural parameters) of the messages. Within the same approximation assumption, we also replace the cavity quantities corresponding to variances with the non-cavity counterparts. Dividing the update equations in a *forward* and *backward* pass, and ordering them using time indexes in such a way that we have an efficient flow of information, we obtain the set of BP equations presented in the main text equations (7)–(18) and in the appendix equations (62)–(73).

A.3. BP equations

We report here the end result of the derivation in last section, the complete set of BP equations also presented in the main text as equations (7)–(18).

A.3.1. Initialization

At $\tau = 0$:

$$B^{\ell,0}_{in\to k} = 0, \tag{58}$$

$$A^{\ell,0}_{in} = 0, \tag{59}$$

$$H^{\ell,0}_{ki\to n} = 0, \tag{60}$$

$$G^{\ell,0}_{ki} = 0. \tag{61}$$

A.3.2. Forward pass

At each $\tau = 1,\dots,\tau_{max}$, for $\ell = 0,\dots,L$:

$$\hat{x}^{\ell,\tau}_{in\to k} = \partial_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in\to k}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right), \tag{62}$$

$$\Delta^{\ell,\tau}_{in} = \partial^2_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right), \tag{63}$$

$$m^{\ell,\tau}_{ki\to n} = \partial_H \psi\left(H^{\ell,\tau-1}_{ki\to n}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}\right), \tag{64}$$

$$\sigma^{\ell,\tau}_{ki} = \partial^2_H \psi\left(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}\right), \tag{65}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki}\right)^2 \Delta^{\ell,\tau}_{in} + \sigma^{\ell,\tau-1}_{ki}\left(\hat{x}^{\ell,\tau}_{i'n}\right)^2 + \sigma^{\ell,\tau-1}_{ki}\Delta^{\ell,\tau}_{in}\right), \tag{66}$$

$$\omega^{\ell,\tau}_{kn\to i} = \sum_{i'\neq i} m^{\ell,\tau}_{ki'\to n}\hat{x}^{\ell,\tau}_{i'n\to k}. \tag{67}$$

In these equations for simplicity we abused the notation, in fact for the first layer $\hat{\boldsymbol{x}}^{\ell=0,\tau}_n$ is fixed and given by the input $\boldsymbol{x}_n$ while $\boldsymbol{\Delta}^{\ell=0,\tau}_n = 0$ instead.

A.3.3. Backward pass

For $\tau = 1,\dots,\tau_{max}$, for $\ell = L,\dots,0$:

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn\to i}, V^{\ell,\tau}_{kn}\right), \tag{68}$$

$$\Gamma^{\ell,\tau}_{kn} = -\partial^2_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn}, V^{\ell,\tau}_{kn}\right), \tag{69}$$

$$A^{\ell,\tau}_{in} = \sum_k \left(\left(\left(m^{\ell,\tau}_{ki}\right)^2 + \sigma^{\ell,\tau}_{ki}\right)\Gamma^{\ell,\tau}_{kn} - \sigma^{\ell,\tau}_{ki}\left(g^{\ell,\tau}_{kn}\right)^2\right), \tag{70}$$

$$B^{\ell,\tau}_{in\to k} = \sum_{k'\neq k} m^{\ell,\tau}_{k'i\to n} g^{\ell,\tau}_{k'n\to i}, \tag{71}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left(\left(\left(\hat{x}^{\ell,\tau}_{in}\right)^2 + \Delta^{\ell,\tau}_{in}\right)\Gamma^{\ell,\tau}_{kn} - \Delta^{\ell,\tau}_{in}\left(g^{\ell,\tau}_{kn}\right)^2\right), \tag{72}$$

$$H^{\ell,\tau}_{ki\to n} = \sum_{n'\neq n} \hat{x}^{\ell,\tau}_{in'\to k} g^{\ell,\tau}_{kn'\to i}. \tag{73}$$

In these equations as well we abused the notation: calling L the number of hidden neuron layers, when $\ell = L$ one should use $\varphi^{L+1}(y,\omega,V)$ from equation (23) instead of $\varphi^{L+1}(B,A,\omega,V)$.

A.4. BPI equations

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).

A.4.1. Forward pass

$$\hat{x}^{\ell,\tau}_{in} = \partial_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right), \tag{74}$$

$$\Delta^{\ell,\tau}_{in} = \partial^2_B \varphi^{\ell}\left(B^{\ell,\tau-1}_{in}, A^{\ell,\tau-1}_{in}, \omega^{\ell-1,\tau}_{in}, V^{\ell-1,\tau}_{in}\right), \tag{75}$$

$$m^{\ell,\tau}_{ki} = \partial_H \psi\left(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}\right), \tag{76}$$

$$\sigma^{\ell,\tau}_{ki} = \partial^2_H \psi\left(H^{\ell,\tau-1}_{ki}, G^{\ell,\tau-1}_{ki}, \theta^{\ell}_{ki}\right), \tag{77}$$

$$V^{\ell,\tau}_{kn} = \sum_i \left(\left(m^{\ell,\tau}_{ki}\right)^2 \Delta^{\ell,\tau}_{in} + \sigma^{\ell,\tau}_{ki}\left(\hat{x}^{\ell,\tau}_{in}\right)^2 + \sigma^{\ell,\tau}_{ki}\Delta^{\ell,\tau}_{in}\right), \tag{78}$$

$$\omega^{\ell,\tau}_{kn} = \sum_i m^{\ell,\tau}_{ki}\hat{x}^{\ell,\tau}_{in}. \tag{79}$$

A.4.2. Backward pass

$$g^{\ell,\tau}_{kn\to i} = \partial_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn} - m^{\ell,\tau}_{ki}\hat{x}^{\ell,\tau}_{ai}, V^{\ell,\tau}_{kn}\right), \tag{80}$$

$$\Gamma^{\ell,\tau}_{kn} = -\partial^2_\omega \varphi^{\ell+1}\left(B^{\ell+1,\tau}_{kn}, A^{\ell+1,\tau}_{kn}, \omega^{\ell,\tau}_{kn}, V^{\ell,\tau}_{kn}\right), \tag{81}$$

$$A^{\ell,\tau}_{in} = \sum_k \left(\left(m^{\ell,\tau}_{ki}\right)^2 + \sigma^{\ell,\tau}_{ki}\right)\Gamma^{\ell,\tau}_{kn} - \sigma^{\ell,\tau}_{ki}\left(g^{\ell,\tau}_{kn}\right)^2, \tag{82}$$

$$B^{\ell,\tau}_{in} = \sum_k m^{\ell,\tau}_{ki} g^{\ell,\tau}_{kn\to i}, \tag{83}$$

$$G^{\ell,\tau}_{ki} = \sum_n \left(\left(\hat{x}^{\ell,\tau}_{in}\right)^2 + \Delta^{\ell,\tau}_{in}\right)\Gamma^{\ell,\tau}_{kn} - \Delta^{\ell,\tau}_{in}\left(g^{\ell,\tau}_{kn}\right)^2, \tag{84}$$

$$H^{\ell,\tau}_{ki} = \sum_n \hat{x}^{\ell,\tau}_{in} g^{\ell,\tau}_{kn\to i}. \tag{85}$$

A.5. MF equations

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 non-negligible discrepancy between the two algorithms in terms of generalization performance and computational time.

A.5.1. Forward pass

$$\hat{x}_{in}^{\ell,\tau} = \partial_B \varphi^{\ell}\left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau}\right), \tag{86}$$

$$\Delta_{in}^{\ell,\tau} = \partial_B^2 \varphi^{\ell}\left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau}\right), \tag{87}$$

$$m_{ki}^{\ell,\tau} = \partial_H \psi\left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell}\right), \tag{88}$$

$$\sigma_{ki}^{\ell,\tau} = \partial_H^2 \psi\left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell}\right), \tag{89}$$

$$V_{kn}^{\ell,\tau} = \sum_i \left(\left(m_{ki}^{\ell,\tau}\right)^2 \Delta_{in}^{\ell,\tau} + \sigma_{ki}^{\ell,\tau} (\hat{x}_{in}^{\ell,\tau})^2 + \sigma_{ki}^{\ell,\tau} \Delta_{in}^{\ell,\tau}\right), \tag{90}$$

$$\omega_{kn}^{\ell,\tau} = \sum_i m_{ki}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau}. \tag{91}$$

A.5.2. Backward pass

$$g_{kn}^{\ell,\tau} = \partial_\omega \varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right), \tag{92}$$

$$\Gamma_{kn}^{\ell,\tau} = -\partial_\omega^2 \varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right), \tag{93}$$

$$A_{in}^{\ell,\tau} = \sum_k \left(\left(m_{ki}^{\ell,\tau}\right)^2 + \sigma_{ki}^{\ell,\tau}\right) \Gamma_{kn}^{\ell,\tau} - \sigma_{ki}^{\ell,\tau} \left(g_{kn}^{\ell,\tau}\right)^2, \tag{94}$$

$$B_{in}^{\ell,\tau} = \sum_k m_{ki}^{\ell,\tau} g_{kn}^{\ell,\tau}, \tag{95}$$

$$G_{ki}^{\ell,\tau} = \sum_n \left(\left(\hat{x}_{in}^{\ell,\tau}\right)^2 + \Delta_{in}^{\ell,\tau}\right) \Gamma_{kn}^{\ell,\tau} - \Delta_{in}^{\ell,\tau} \left(g_{kn}^{\ell,\tau}\right)^2, \tag{96}$$

$$H_{ki}^{\ell,\tau} = \sum_n \hat{x}_{in}^{\ell,\tau} g_{kn}^{\ell,\tau}. \tag{97}$$

A.6. Derivation of the AMP equations

In order to obtain the AMP equations, we approximate cavity quantities with non-cavity ones in the BP equations equations (62)–(73) using a first order expansion. We start with the mean activation:

$$\begin{aligned}
\hat{x}_{in\to k}^{\ell,\tau} &= \partial_B \varphi^{\ell}\left(B_{in}^{\ell,\tau-1} - m_{ki\to n}^{\ell,\tau-1} g_{kn\to i}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau}\right) \\
&\approx \partial_B \varphi^{\ell}\left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau}\right) \\
&\quad - m_{ki\to n}^{\ell,\tau-1} g_{kn\to i}^{\ell,\tau-1} \partial_B^2 \varphi^{\ell}\left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau}\right) \\
&\approx \hat{x}_{in}^{\ell,\tau} - m_{ki}^{\ell,\tau-1} g_{kn}^{\ell,\tau-1} \Delta_{in}^{\ell,\tau}.
\end{aligned} \tag{98}$$

Analogously, for the weight's mean we have:

$$\begin{aligned}
m_{ki\to n}^{\ell,\tau} &= \partial_H \psi\left(H_{ki}^{\ell,\tau-1} - \hat{x}_{in\to k}^{\ell,\tau-1} g_{kn\to i}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell}\right) \\
&\approx \partial_H \psi\left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell}\right) - \hat{x}_{in\to k}^{\ell,\tau-1} g_{kn\to i}^{\ell,\tau-1} \partial_H^2 \psi\left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell}\right) \\
&\approx m_{ki}^{\ell,\tau} - \hat{x}_{in}^{\ell,\tau-1} g_{kn}^{\ell,\tau-1} \sigma_{ki}^{\ell,\tau}.
\end{aligned} \tag{99}$$

This brings us to:

$$\begin{aligned}\omega_{kn}^{\ell,\tau} &= \sum_i m_{ki\to n}^{\ell,\tau}\hat{x}_{in\to k}^{\ell,\tau} \\ &\approx \sum_i m_{ki}^{\ell,\tau}\hat{x}_{in}^{\ell,\tau} - g_{kn}^{\ell,\tau-1}\sum_i \sigma_{ki}^{\ell,\tau}\hat{x}_{in}^{\ell,\tau}\hat{x}_{in}^{\ell,\tau-1} - g_{kn}^{\ell,\tau-1}\sum_i m_{ki}^{\ell,\tau}m_{ki}^{\ell,\tau-1}\Delta_{in}^{\ell,\tau} \\ &\quad + \left(g_{kn}^{\ell,\tau-1}\right)^2\sum_i \sigma_{ki}^{\ell,\tau}m_{ki}^{\ell,\tau-1}\hat{x}_{in}^{\ell,\tau-1}\Delta_{in}^{\ell,\tau}. \end{aligned} \tag{100}$$

Let us now apply the same procedure to the other set of cavity messages:

$$\begin{aligned} g_{kn\to i}^{\ell,\tau} &= \partial_\omega\varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau} - m_{ki\to n}^{\ell,\tau}\hat{x}_{in\to k}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right) \\ &\approx \partial_\omega\varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right) \\ &\quad - m_{ki\to n}^{\ell,\tau}\hat{x}_{in\to k}^{\ell,\tau}\partial_\omega^2\varphi^{\ell+1}\left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau}\right) \\ &\approx g_{kn}^{\ell,\tau} + m_{ki}^{\ell,\tau}\hat{x}_{in}^{\ell,\tau}\Gamma_{kn}^{\ell,\tau}, \end{aligned} \tag{101}$$

$$\begin{aligned} B_{in}^{\ell,\tau} &= \sum_k m_{ki\to n}^{\ell,\tau}g_{kn\to i}^{\ell,\tau} \\ &\approx \sum_k m_{ki}^{\ell,\tau}g_{kn}^{\ell,\tau} - \hat{x}_{in}\sum_k \left(g_{kn}^{\ell,\tau}\right)^2\sigma_{ki}^{\ell,\tau} + \hat{x}_{in}^{\ell,\tau}\sum_k \left(m_{ki}^{\ell,\tau}\right)^2\Gamma_{kn}^{\ell,\tau} \\ &\quad - \left(\hat{x}_{in}^{\ell,\tau}\right)^2\sum_k \sigma_{ki}^{\ell,\tau}m_{ki}^{\ell,\tau}g_{kn}^{\ell,\tau}\Gamma_{kn}^{\ell,\tau}, \end{aligned} \tag{102}$$

$$\begin{aligned} H_{ki}^{\ell,\tau} &= \sum_n \hat{x}_{in\to k}^{\ell,\tau}g_{kn\to i}^{\ell,\tau} \\ &\approx \sum_n \hat{x}_{in}^{\ell,\tau}g_{kn}^{\ell,\tau} + m_{ki}^{\ell,\tau}\sum_n \left(\hat{x}_{in}^{\ell,\tau}\right)^2\Gamma_{kn}^{\ell,\tau} - m_{ki}^{\ell,\tau}\sum_n \left(g_{kn}^{\ell,\tau}\right)^2\Delta_{in}^{\ell,\tau} \\ &\quad - \left(m_{ki}^{\ell,\tau}\right)^2\sum_n g_{kn}^{\ell,\tau}\Gamma_{kn}^{\ell,\tau}\Delta_{in}^{\ell,\tau}\hat{x}_{in}^{\ell,\tau}. \end{aligned} \tag{103}$$

We are now able to write down the full AMP equations, that we present in the next section.

A.7. AMP equations

In summary, in the last section we derived the AMP algorithm as a closure of the BP messages passing over non-cavity quantities, relying on some statistical assumptions on messages and interactions. With respect to the MF message passing, we find some additional terms that go under the name of Onsager corrections. In-depth overviews of the AMP (also known as Thouless-Anderson-Palmer (TAP)) approach can be found in Zdeborová and Krzakala (2016), Mézard (2017), Gabrié (2020). The final form of the AMP equations for the multi-layer perceptron is given below.

A.7.1. Initialization

At $\tau = 0$:

$$B_{in}^{\ell,0} = 0, \tag{104}$$

$$A_{in}^{\ell,0} = 0, \tag{105}$$

$$H_{ki}^{\ell,0} = 0 \text{ or some values}, \tag{106}$$

$$G_{ki}^{\ell,0} = 0 \text{ or some values}, \tag{107}$$

$$g_{kn}^{\ell,0} = 0. \tag{108}$$

A.7.2. Forward pass

At each $\tau = 1, \dots, \tau_{max}$, for $\ell = 0, \dots, L$:

$$\hat{x}_{in}^{\ell,\tau} = \partial_B \varphi^{\ell} \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right), \tag{109}$$

$$\Delta_{in}^{\ell,\tau} = \partial_B^2 \varphi^{\ell} \left(B_{in}^{\ell,\tau-1}, A_{in}^{\ell,\tau-1}, \omega_{in}^{\ell-1,\tau}, V_{in}^{\ell-1,\tau} \right), \tag{110}$$

$$m_{ki}^{\ell,\tau} = \partial_H \psi \left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell} \right), \tag{111}$$

$$\sigma_{ki}^{\ell,\tau} = \partial_H^2 \psi \left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell} \right), \tag{112}$$

$$V_{kn}^{\ell,\tau} = \sum_i \left(\left(m_{ki}^{\ell,\tau} \right)^2 \Delta_{in}^{\ell,\tau} + \sigma_{ki}^{\ell,\tau} \left(\hat{x}_{i'n}^{\ell,\tau} \right)^2 + \sigma_{ki}^{\ell,\tau} \Delta_{in}^{\ell,\tau} \right), \tag{113}$$

$$\begin{aligned} \omega_{kn}^{\ell,\tau} = & \sum_i m_{ki}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau} - g_{kn}^{\ell,\tau-1} \sum_i \sigma_{ki}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau-1} - g_{kn}^{\ell,\tau-1} \sum_i m_{ki}^{\ell,\tau} m_{ki}^{\ell,\tau-1} \Delta_{in}^{\ell,\tau} \\ & + \left(g_{kn}^{\ell,\tau-1} \right)^2 \sum_i \sigma_{ki}^{\ell,\tau} m_{ki}^{\ell,\tau-1} \hat{x}_{in}^{\ell,\tau-1} \Delta_{in}^{\ell,\tau}. \end{aligned} \tag{114}$$

A.7.3. Backward pass

$$g_{kn}^{\ell,\tau} = \partial_\omega \varphi^{\ell+1} \left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn\to i}^{\ell,\tau}, V_{kn}^{\ell,\tau} \right), \tag{115}$$

$$\Gamma_{kn}^{\ell,\tau} = -\partial_\omega^2 \varphi^{\ell+1} \left(B_{kn}^{\ell+1,\tau}, A_{kn}^{\ell+1,\tau}, \omega_{kn}^{\ell,\tau}, V_{kn}^{\ell,\tau} \right), \tag{116}$$

$$A_{in}^{\ell,\tau} = \sum_k \left(\left(\left(m_{ki}^{\ell,\tau} \right)^2 + \sigma_{ki}^{\ell,\tau} \right) \Gamma_{kn}^{\ell,\tau} - \sigma_{ki}^{\ell,\tau} \left(g_{kn}^{\ell,\tau} \right)^2 \right), \tag{117}$$

$$\begin{aligned} B_{in}^{\ell,\tau} = & \sum_k m_{ki}^{\ell,\tau} g_{kn}^{\ell,\tau} - \hat{x}_{in} \sum_k \left(g_{kn}^{\ell,\tau} \right)^2 \sigma_{ki}^{\ell,\tau} + \hat{x}_{in}^{\ell,\tau} \sum_k \left(m_{ki}^{\ell,\tau} \right)^2 \Gamma_{kn}^{\ell,\tau} \\ & - \left(\hat{x}_{in}^{\ell,\tau} \right)^2 \sum_k \sigma_{ki}^{\ell,\tau} m_{ki}^{\ell,\tau} g_{kn}^{\ell,\tau} \Gamma_{kn}^{\ell,\tau}, \end{aligned} \tag{118}$$

$$G_{ki}^{\ell,\tau} = \sum_n \left(\left(\left(\hat{x}_{in}^{\ell,\tau} \right)^2 + \Delta_{in}^{\ell,\tau} \right) \Gamma_{kn}^{\ell,\tau} - \Delta_{in}^{\ell,\tau} \left(g_{kn}^{\ell,\tau} \right)^2 \right), \tag{119}$$

$$\begin{aligned} H_{ki}^{\ell,\tau} = & \sum_n \hat{x}_{in}^{\ell,\tau} g_{kn}^{\ell,\tau} + m_{ki}^{\ell,\tau} \sum_n \left(\hat{x}_{in}^{\ell,\tau} \right)^2 \Gamma_{kn}^{\ell,\tau} - m_{ki}^{\ell,\tau} \sum_n \left(g_{kn}^{\ell,\tau} \right)^2 \Delta_{in}^{\ell,\tau} \\ & - \left(m_{ki}^{\ell,\tau} \right)^2 \sum_n g_{kn}^{\ell,\tau} \Gamma_{kn}^{\ell,\tau} \Delta_{in}^{\ell,\tau} \hat{x}_{in}^{\ell,\tau}. \end{aligned} \tag{120}$$

A.8. Activation functions

A.8.1. Sign

In most of our experiments we use *sign* activations in each layer. With this choice, the neuron's free energy (21) takes the form:

$$\varphi(B, A, \omega, V) = \log \left(\frac{1}{2} \sum_{x \in \{-1,+1\}} e^{Bx} \mathcal{H} \left(-\frac{x\omega}{\sqrt{V}} \right) \right) + \frac{1}{2} \log(2\pi V), \tag{121}$$

where

$$\mathcal{H} = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right). \tag{122}$$

Notice that for *sign* activations the messages A can be dropped.

A.8.2. ReLU
For $ReLU(x) = \max(0,x)$ activations the free energy (21) becomes:

$$\varphi(B,A,\omega,V) = \int \mathrm{d}x\mathrm{d}z\, e^{-\frac{1}{2}Ax^2+Bx}\,\delta(x-\max(0,z))\; e^{-\frac{(\omega-z)^2}{2V}} \tag{123}$$

$$= \log\left(H\left(\frac{\omega}{\sqrt{V}}\right) + \frac{\mathcal{N}(\omega;B/A,V+\frac{1}{A})}{A\mathcal{N}(B;0,A)} H\left(-\frac{BV+\omega}{\sqrt{V+AV^2}}\right)\right) + \frac{1}{2}\log(2\pi V), \tag{124}$$

where

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{(x-\mu)^2}{2\Sigma}}. \tag{125}$$

A.9. The 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 $k = 1,\dots,K$, 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 Z_{k^*} corresponding to the output constraint reads:

$$Z_{k^*} = \int \prod_k dz_k \mathcal{N}(z_k;\omega_k,V_k) \prod_{k\neq k^*} \Theta(z_{k^*}-z_k), \tag{126}$$

$$= \int dz_{k^*}\, \mathcal{N}(z_{k^*};\omega_{k^*},V_{k^*}) \prod_{k\neq k^*} \mathcal{H}\left(-\frac{z_{k^*}-\omega_k}{\sqrt{V_k}}\right), \tag{127}$$

here $\Theta(x)$ is the Heaviside indicator function and we used the definition of $\mathcal{H}$ from equation (122). The integral on the last line cannot be expressed analytically, therefore we have to resort to approximations.

A.9.1. Approach 1: Jensen inequality
Using the Jensen inequality we obtain:

$$\phi_{k^*} = \log Z_{k^*} = \log \mathbb{E}_{z\sim\mathcal{N}(\omega_{k^*},V_{k^*})} \prod_{k\neq k^*} \mathcal{H}\left(-\frac{z-\omega_k}{\sqrt{V_k}}\right), \tag{128}$$

$$\geqslant \sum_{k\neq k^*} \mathbb{E}_{z\sim\mathcal{N}(\omega_{k^*},V_{k^*})} \log \mathcal{H}\left(-\frac{z-\omega_k}{\sqrt{V_k}}\right). \tag{129}$$

Reparameterizing the expectation we have:

$$\tilde{\phi}_{k^*} = \sum_{k\neq k^*} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \log \mathcal{H}\left(-\frac{\omega_{k^*}+\epsilon\sqrt{V_{k^*}}-\omega_k}{\sqrt{V_k}}\right). \tag{130}$$

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

$$\partial_{\omega_k}\tilde{\phi}_{k^*} = \begin{cases} -\frac{1}{\sqrt{V_k}}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\,\mathcal{K}\left(-\frac{\omega_{k^*}+\epsilon\sqrt{V_{k^*}}-\omega_k}{\sqrt{V_k}}\right) & k\neq k^* \\ \sum_{k'\neq k^*} \frac{1}{\sqrt{V_{k'}}}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\,\mathcal{K}\left(-\frac{\omega_{k^*}+\epsilon\sqrt{V_{k^*}}-\omega_{k'}}{\sqrt{V_{k'}}}\right) & k=k^*, \end{cases} \tag{131}$$

where we have defined:

$$\mathcal{K}(x) = \frac{\mathcal{N}(x)}{\mathcal{H}(x)} = \frac{\sqrt{2/\pi}}{erfcx(x/2)}. \tag{132}$$

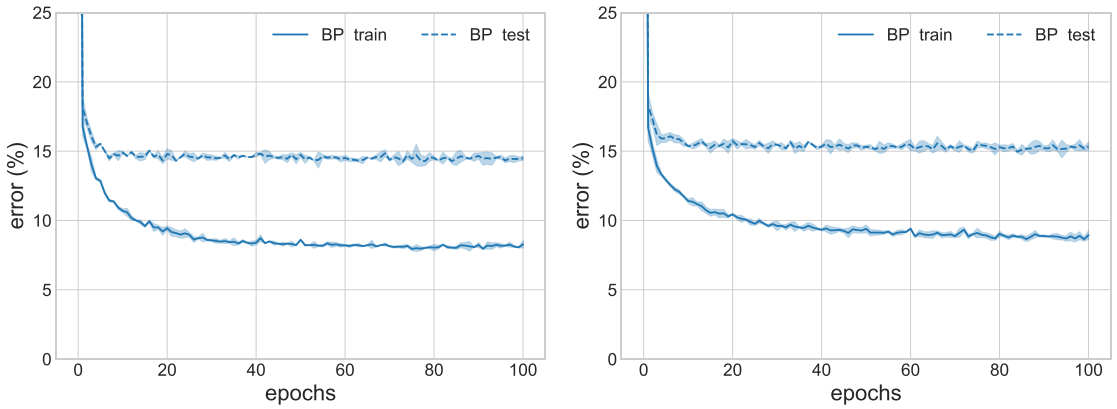

Figure 6. MLP with 2 hidden layers with 101 hidden units each, batch-size 128 on the Fashion-MNIST dataset. In the first two layers we use the BP equations, while in the last layer the ArgMax ones. (Left) ArgMax layer first version; (Right) ArgMax layer second version. Even if it is possible to reach similar accuracies with the two versions, we decide to use the first one as it is simpler to use.

A.9.2. Approach 2: Jensen again

A further simplification is obtained by applying Jensen inequality again to (130) but in the opposite direction, therefore we renounce to having a bound and look only for an approximation. We have the new effective free energy:

$$\tilde{\phi}_{k^*} = \sum_{k\neq k^*} \log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \mathcal{H}\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}}\right), \tag{133}$$

$$= \sum_{k\neq k^*} \log \mathcal{H}\left(-\frac{\omega_{k^*} - \omega_k}{\sqrt{V_k + V_{k^*}}}\right). \tag{134}$$

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

$$\partial_{\omega_k}\tilde{\phi}_{k^*} = \begin{cases} -\frac{1}{\sqrt{V_k+V_{k^*}}}\mathcal{K}\left(-\frac{\omega_{k^*}-\omega_k}{\sqrt{V_k+V_{k^*}}}\right) & k \neq k^* \\ \sum_{k'\neq k^*} \frac{1}{\sqrt{V_{k'}+V_{k^*}}}\mathcal{K}\left(-\frac{\omega_{k^*}-\omega_{k'}}{\sqrt{V_{k'}+V_{k^*}}}\right) & k = k^* \end{cases}. \tag{135}$$

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 of $\mathcal{K}$ in equation (132).

We show in figure 6 the negligible difference between the two ArgMax versions when using BP on the layers before the last one (which performs only the ArgMax).

Appendix B. Experimental details

B.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 mini-batch size *bs*; the parameter ρ (that has to be tuned similarly to the learning rate in SGD); the damping parameter α (that performs a running smoothing on the BP fields along the dynamics by adding a fraction of the field at the previous iteration, see equations (136) and (137)); the initialization coefficient ϵ that we use to to sample the parameters of our prior distribution $q_\theta(\mathcal{W})$ according to $\theta_{ki}^{\ell,t=0} \sim \epsilon\mathcal{N}(0,1)$. Different choices of ϵ correspond to different initial distribution of the weights' magnetization $m_{ki}^{\ell} = \tanh(\theta_{ki}^{\ell})$, as is shown in figure 7); the number of internal steps of reinforcement $\tau_{\max}$ and the associated intensity of the internal reinforcement r. The performances of the BP-based algorithms are robust in a reasonable range of these hyper-parameters. A more principled choice of a good initialization condition could be made by adapting the technique from Stamatescu *et al* (2020).

Notice that among these parameters, the BP dynamics at each layer is mostly sensitive to ρ and α, so that in general we consider them layer-dependent. See appendix B.7 for details on the effect of these parameters

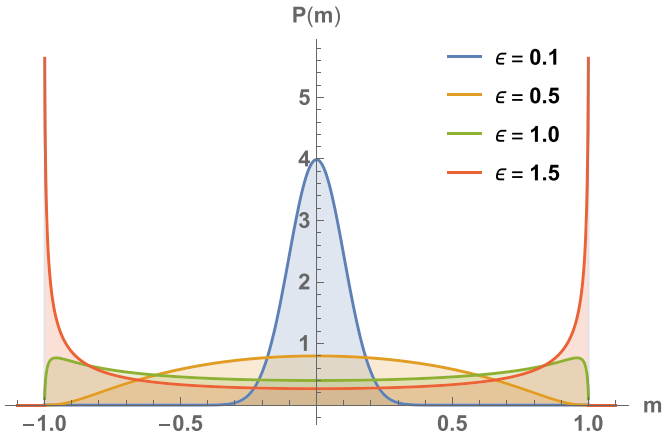

Figure 7. Initial distribution of the magnetizations varying the parameter ϵ. The initial distribution is more concentrated around ± 1 as ϵ increases (i.e. it is more bimodal and the initial configuration is more polarized).

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), $\epsilon = 1.0$, $\tau_{\max} = 1$, $r = 0$.

B.2. Damping scheme for the message passing

We use a damping parameter $\alpha \in (0, 1)$ to stabilize the training, changing the updated rule for the weights' means as follows:

$$\tilde{m}_{ki}^{\ell,\tau} = \partial_H \psi \left(H_{ki}^{\ell,\tau-1}, G_{ki}^{\ell,\tau-1}, \theta_{ki}^{\ell} \right), \tag{136}$$

$$m_{ki}^{\ell,\tau} = \alpha\, m_{ki}^{\ell,\tau-1} + (1-\alpha)\, \tilde{m}_{ki}^{\ell,\tau}. \tag{137}$$

B.3. Architectures

In the experiments in which we vary the architecture (see section 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 batch-size is $bs = 128$, the initialization coefficient is $\epsilon = 1.0$.

For all architectures and all BP approximations, we use $\alpha = 0.8$ for each layer, apart for the 501-501-501 MLP in which we use $\alpha = (0.1, 0.1, 0.1, 0.9)$. 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 and the 501-501 MLPs, $\rho = 1.0001$ for all BP approximations; for the 101-101-101 MLP, $\rho = 1.0$ for BP and AMP 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 $\alpha = 0.8$ 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 with minimal hyper-parameter tuning.

The experiments on the Bayesian error are performed on a MLP with 2 hidden layers of 101 units on the MNIST dataset (binary classification). Learning is performed on the totality of the training dataset, the batch-size is $bs = 128$, the initialization coefficient is $\epsilon = 1.0$. In order to find the pointwise configurations we use $\alpha = 0.8$ on each layer and $\rho = (1.0001, 1.0001, 0.9)$, while to find the Bayesian ones we use $\alpha = 0.8$ on each layer and $\rho = (0.9999, 0.9999, 0.9)$ (these value prevent an excessive polarization of the network towards a particular pointwise configurations).

For the continual learning task (see section 4.6) we fixed $\rho = 1$ and $\alpha = 0.8$ on each layer as we empirically observed that polarizing the last layer helps mitigating the forgetting while leaving the single-task performances almost unchanged.

In figure 8 we report training curves on architectures different from the ones reported in the main paper.

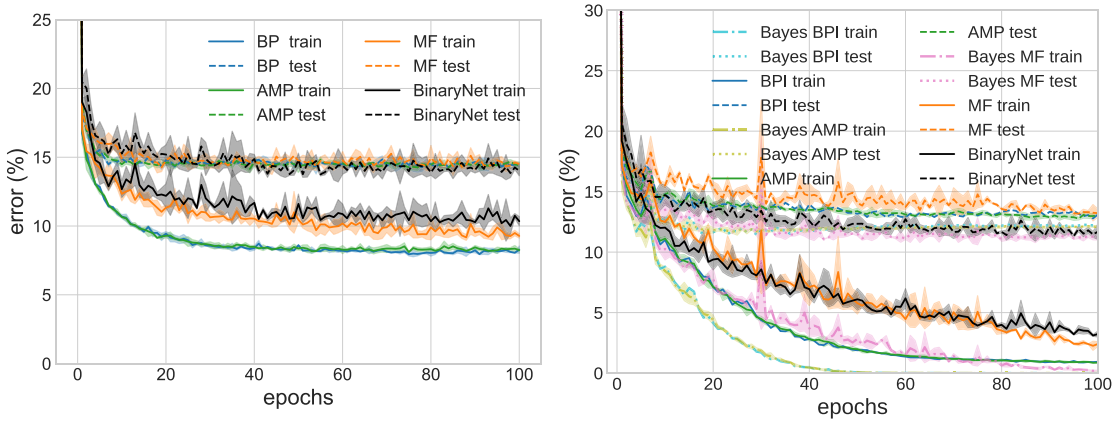

Figure 8. Training curves of message passing algorithms compared with BinaryNet on the Fashion-MNIST dataset (multi-class classification). (Left) Binary MLP with 2 hidden layers of 101 units. (Right) Binary MLP with 4 hidden layers of 501 units. The batch-size is 128 and curves are averaged over 5 realizations of the initial conditions.

Table 2. Train error (%) on Fashion-MNIST of a multilayer perceptron with two hidden layers of 501 units each for BinaryNet (baseline), BP, AMP and MF. All algorithms are trained with batch-size 128 and for 100 epochs. Mean and standard deviations are calculated over five random initializations.

Dataset	BinaryNet	BP	AMP	MF
MNIST (2 classes)	0.05 ± 0.05	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
FashionMNIST (2 classes)	0.3 ± 0.1	0.06 ± 0.01	0.06 ± 0.01	0.09 ± 0.01
CIFAR10 (2 classes)	1.2 ± 0.5	0.37 ± 0.01	0.4 ± 0.1	0.9 ± 0.2
MNIST	0.09 ± 0.01	0.12 ± 0.01	0.12 ± 0.01	0.03 ± 0.01
FashionMNIST	4.0 ± 0.5	3.4 ± 0.1	3.7 ± 0.1	2.5 ± 0.2
CIFAR10	13.0 ± 0.9	4.7 ± 0.1	4.7 ± 0.2	9.2 ± 0.5

B.4. Varying the dataset

When varying the dataset (see section 4.3), all simulation 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 batch-size is $bs = 128$, the initialization coefficient is $\epsilon = 1.0$. For all datasets (MNIST (2 classes), FashionMNIST (2 classes), CIFAR-10 (2 classes), MNIST, FashionMNIST, CIFAR-10) and all algorithms (BP, AMP, MF) we use $\rho = (1.0001, 1.0001, 0.9)$ and $\alpha = 0.8$ for each layer. Using in the first layers values of $\rho = 1 + \epsilon$ with $\epsilon \geqslant 0$ and sufficiently small typically leads to good results.

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, $lr = 100, 10, 1, 0.1, 0.001$. In table 2 we report the final train errors obtained on the different datasets.

B.5. 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 $w_b = sign(w)$. At inference time the forward pass is calculated with the parameters w_b. The backward pass with binary activations is performed 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 and Bengio 2010) for the continuous weights. In figure 3. of the main paper, we apply the Adam optimization rule, noticing that it performs slightly better in train and test generalization performance compared to the pure SGD one.

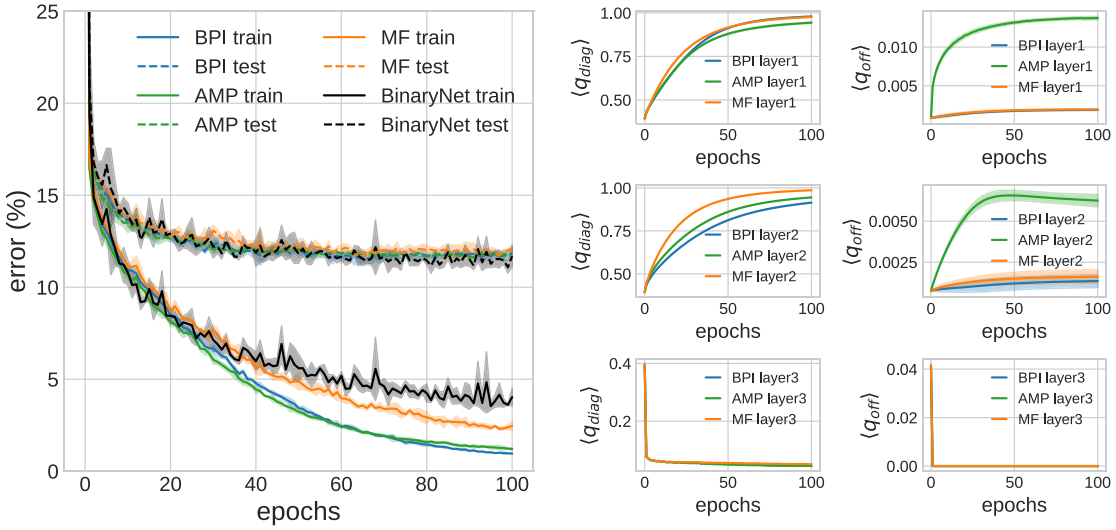

Figure 9. (Right panels) Polarizations q_{diag} and overlaps q_{off} on each layer of a MLP with 2 hidden layers of 501 units on the Fashion-MNIST dataset (multi-class), the batch-size is $bs = 128$. (Right) Corresponding train and test error curves.

B.6. EBP implementation

Expectation back propagation (EBP) (Soudry *et al* 2014b) is parameter-free Bayesian algorithm that uses a mean-field (MF) approximation (fully factorized form for the posterior) in an online environment to estimate the Bayesian posterior distribution after the arrival of a new data point. The main differences between EBP and our approach relies in the approximation for the posterior distribution. Moreover we explicitly base the estimation of the marginals on the local high entropy structure. The fact that EBP works has no clear explanation: certainly it cannot be that the MF assumption holds for multi-layer neural networks. Still, it is certainly very interesting that it works. We argue that it might work precisely by virtue of the existence of high local entropy minima and expect it to give similar performance to the MF case of our algorithm. The online iteration could in fact be seen as way of implementing a reinforcement.

We implemented the EBP code along the lines of the original matlab implementation (https://github.com/ExpectationBackpropagation/EBP_Matlab_Code). In order to perform a fair comparison we removed the biases both in the binary and continuous weights versions. It is worth noticing that we faced numerical issues in training with a moderate to big batchsize All the experiments were consequently limited to a batchsize of 10 patterns.

B.7. Unit polarization and overlaps

We define the self-overlap or polarization of a given hidden unit k as $q_k = \frac{1}{N}\sum_i \langle w_{ki}\rangle^2$, where N is the number of parameters of the unit, $\{w_{ki}\}_{i=1}^N$ its binary weights, and the $\langle w_{ki}\rangle$ the mean according to the posterior. It quantifies how much the unit is polarized towards a unique point-wise binary configuration ($q_k = 1$ corresponding to full polarization). The overlap between two units k and k' in the same layer is $q_{kk'} = \frac{1}{N}\sum\langle w_{ki}\rangle\langle w_{k'i}\rangle$. We denote by $q_{diag} = \frac{1}{N_{out}}\sum_{k=1}^{N_{out}} q_k$ and $q_{off} = \frac{2}{N_{out}(N_{out}-1)}\sum_{k<k'}^{N_{out}} q_{kk'}$ the mean polarization and mean overlap in a given layer. We mention that a replica computation corresponding to this model would involve the overlaps $q_{kk'}^{ab}$, where a and b are replica indexes. Within a replica symmetric assumption, $q_{kk'}^{ab}$, with $a \neq b$ corresponds to the $q_{kk'}$ defined above.

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

We find that it is is beneficial to control the polarization layer-per-layer, see figure 9 for the corresponding typical behavior of the mean polarization and the mean overlaps during training. Empirically, we have found that (as we could expect) when training is successful the layers polarize progressively towards $q_k = 1$, i.e. towards a precise point-wise solution, while the overlaps between units in each hidden layer are such that $q_{kk'} \ll 1$ (indicating low redundancy of the units). To this aim, in most cases $\alpha\ell$ can be the same

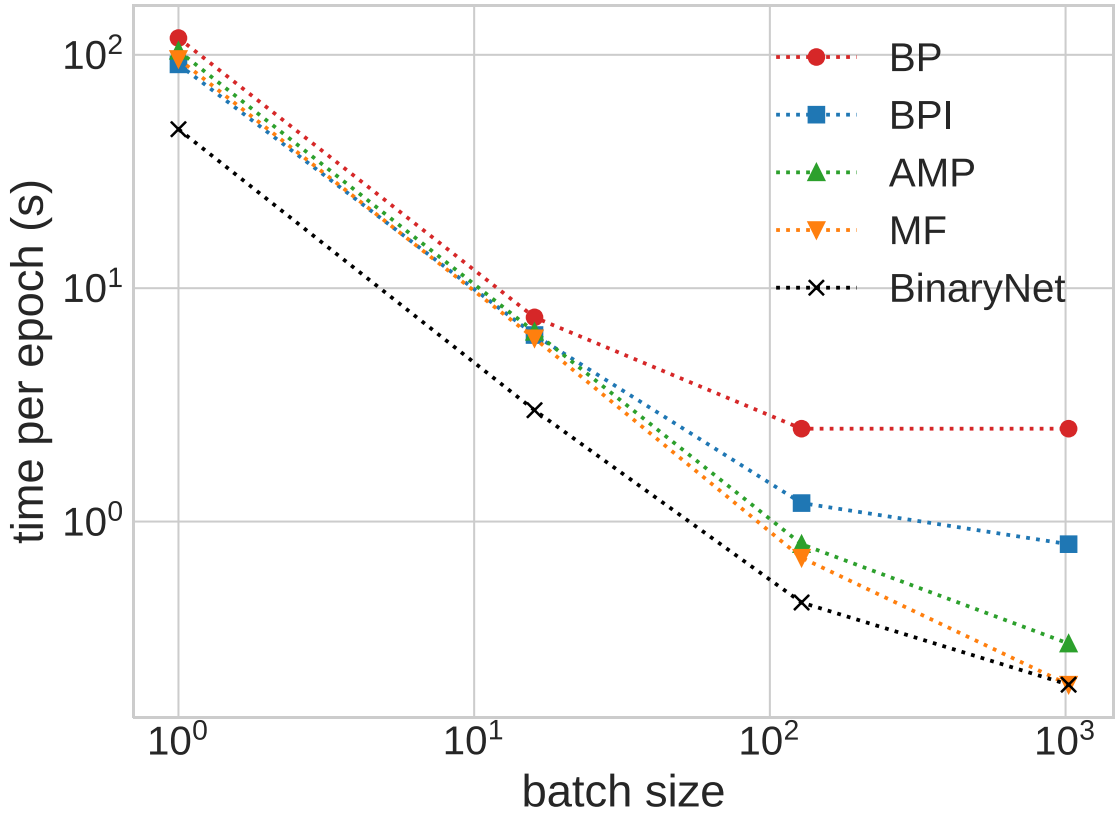

Figure 10. Algorithms time scaling with the batch-size on a MLP with 2 hidden layers of 501 hidden units each on the Fashion-MNIST dataset (multi-class classification). The reported time (in seconds) refers to one epoch for each algorithm.

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 this layer does not or only slightly polarize, i.e. $\langle q_0 \rangle \ll 1$ (this can be achieved by imposing $\rho_L < 1$). 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 $\alpha = 0.8$ 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, without focusing on a particular point-wise binary configuration, while a value greater than 1.0 tends to lead to numerical instabilities and parameters' divergence).

B.8. Computational performance: varying batch-size

In order to compare the time performances of the BP-based algorithms with our implementation of BinaryNet, we report in figure 10 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.

ORCID iD

Carlo Lucibello https://orcid.org/0000-0003-0837-9783

References

Abbott M, Aluthge D, N3N5, Schaub S, Lucibello C, Elrod C and Chen J 2021 Tullio.jl julia package (available at: https://github.com/mcabbott/Tullio.jl)

Aljundi R, Babiloni F, Elhoseiny M, Rohrbach M and Tuytelaars T 2018 Memory aware synapses: learning what (not) to forget *Proc. European Conf. on Computer Vision* (*ECCV*) pp 139–54

Ardakani A, Condo C and Gross W J 2017 Sparsely-connected neural networ VLSI implementation of deep neural networks *5th Int. Conf. on Learning Representations, ICLR 2017, Conf. Track Proc.* (*Toulon, France, 24–26 April 2017*) (available at: OpenReview.net)

Aubin B, Loureiro B, Maillard A, Krzakala F and Zdeborová L 2021 The spiked matrix model with generative priors *IEEE Trans. Inf. Theory* **67** 1156–81

Baldassi C, Borgs C, Chayes J T, Ingrosso A, Lucibello C, Saglietti L and Zecchina R 2016 Unreasonable effectiveness of learning neural networks: from accessible states and robust ensembles to basic algorithmic schemes *Proc. Natl Acad. Sci.* **113** E7655–62

Baldassi C, Braunstein A, Brunel N and Zecchina R 2007 Efficient supervised learning in networks with binary synapses *Proc. Natl Acad. Sci.* **104** 11079–84

Baldassi C, Gerace F, Lucibello C, Saglietti L and Zecchina R 2016 Learning may need only a few bits of synaptic precision *Phys. Rev.* E **93** 052313

Baldassi C, Ingrosso A, Lucibello C, Saglietti L and Zecchina R 2015 Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses *Phys. Rev. Lett.* **115** 128101

Baldassi C, Pittorino F and Zecchina R 2020 Shaping the learning landscape in neural networks around wide flat minima *Proc. Natl Acad. Sci.* **117** 161–70

Barbier J, Krzakala F, Macris N, Miolane Leo and Zdeborová L 2019 Optimal errors and phase transitions in high-dimensional generalized linear models *Proc. Natl Acad. Sci.* **116** 5451–60

Bethe H 1935 Statistical theory of superlattices *Proc. R. Soc.* A **150** 552

Braunstein A and Zecchina R 2006 Learning by message passing in networks of discrete synapses *Phys. Rev. Lett.* **96** 030201

Chaudhari P, Choromanska A, Soatto S, LeCun Y, Baldassi C, Borgs C, Chayes J T, Sagun L and Zecchina R 2017 Entropy-sgd: biasing gradient descent into wide valleys *5th Int. Conf. on Learning Representations, ICLR 2017, Conf. Track Proc. (Toulon, France, 24–26 April 2017)* (available at: OpenReview.net)

Diffenderfer J and Kailkhura B 2021 Multi-prize lottery ticket hypothesis: finding accurate binary neural networks by pruning a randomly weighted network *Int. Conf. on Learning Representations*

Donoho D L, Maleki A and Montanari A 2009 Message-passing algorithms for compressed sensing *Proc. Natl Acad. Sci.* **106** 18914–9

Feng Y and Tu Y 2021 The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima *Proc. Natl Acad. Sci.* **118** e2015617118

Fletcher A K, Rangan S and Schniter P 2018 Inference in deep networks in high dimensions *2018 IEEE Int. Symp. on Information Theory* (*ISIT*) (IEEE) pp 1884–8

Frankle J, Dziugaite G K, Roy D and Carbin M 2021 Pruning neural networks at initialization: why are we missing the mark? *Int. Conf. on Learning Representations*

Fusi S, Drew P J and Abbott L F 2005 Cascade models of synaptically stored memories *Neuron* **45** 599–611

Gabrié M 2020 Mean-field inference methods for neural networks *J. Phys. A: Math. Theor.* **53** 223002

Gabrie M, Manoel, A Luneau C, Barbier J, Macris N, Krzakala F and Zdeborova L 2019 Entropy and mutual information in models of deep neural networks *J. Stat. Mech.* **2019** 124014

Gallager R 1962 Low-density parity-check codes *IRE Trans. Inf. Theory* **8** 21–28

Garipov T, Izmailov P, Podoprikhin D, Vetrov D P and Wilson A G 2018 Loss surfaces, mode connectivity and fast ensembling of dnns *Advances in Neural Information Processing Systems* vol 31, ed S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett (Red Hook, NY: Curran Associates)

Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics* (*Proc. Machine Learning Research*) vol 9, ed Y W Teh and M Titterington (Sardinia: PMLR) pp 249–56

Goldt S, Mézard M, Krzakala F and Zdeborová L 2020 Modeling the influence of data structure on learning in neural networks: the hidden manifold model *Phys. Rev.* X **10** 041044

Goodfellow I J, Mirza M, Xiao D, Courville A and Bengio Y 2013 An empirical investigation of catastrophic forgetting in gradient-based neural networks (arXiv:1312.6211)

Han S, Mao H and Dally W J 2016 Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding *4th Int. Conf. on Learning Representations, ICLR 2016, Conf. Track Proc.* (*San Juan, Puerto Rico, 2–4 May 2016*) ed Y Bengio and Y LeCun

Hernández-Lobato Je M and Adams R P 2015 Probabilistic backpropagation for scalable learning of bayesian neural networks *Proc. 32nd Int. Conf. on Int. Conf. on Machine Learning* (*ICML'15*) vol 37 pp 1861–9 (available at: JMLR.org)

Hubara I, Courbariaux M, Soudry D, El-Yaniv R and Bengio Y 2016 Binarized neural networks *Advances in Neural Information Processing Systems* vol 29, ed D Lee, M Sugiyama, U Luxburg, I Guyon and R Garnett (Red Hook, NY: Curran Associates)

Jiang Y, Neyshabur B, Mobahi H, Krishnan D and Bengio S 2020 Fantastic generalization measures and where to find them *Int. Conf. on Learning Representations*

Kabashima Y, Krzakala F, Mézard M, Sakata A and Zdeborová L 2016 Phase transitions and sample complexity in bayes-optimal matrix factorization *IEEE Trans. Inf. Theory* **62** 4228–65

Kirkpatrick J *et al* 2017 Overcoming catastrophic forgetting in neural networks *Proc. Natl Acad. Sci.* **114** 3521–6

Kuck J, Chakraborty S, Tang H, Luo R, Song J, Sabharwal A and Ermon S 2020 Belief propagation neural networks *Advances in Neural Information Processing Systems* vol 33, ed H Larochelle, M Ranzato, R Hadsell, M F Balcan and H Lin (Red Hook, NY: Curran Associates) pp 667–78

Laborieux A, Ernoult M, Hirtzlin T and Querlioz D 2021 Synaptic metaplasticity in binarized neural networks *Nat. Commun.* **12** 2549

Li H, Xu Z, Taylor G, Studer C and Goldstein T 2018 Visualizing the loss landscape of neural nets *Advances in Neural Information Processing Systems* vol 31, ed S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett (Red Hook, NY: Curran Associates)

Liu Z, Shen Z, Li S, Helwegen K, Huang D and Cheng K-T 2021 How do adam and training strategies help bnns optimization *Proc. 38th Int. Conf. on Machine Learning* (*Proc. Machine Learning Research, 18–24 July 2021*) vol 139, ed M Meila and T Zhang (PMLR) pp 6936–46

Maillard A, Krzakala F, Mézard M and Zdeborová L 2021 Perturbative construction of mean-field equations in extensive-rank matrix factorization and denoising (arXiv:2110.08775)
Manoel A, Krzakala F, Mézard M and Zdeborová L 2017 Multi-layer generalized linear estimation *2017 IEEE Int. Symp. on Information Theory* (*ISIT*) pp 2098–102
Manoel A, Krzakala F, Tramel E W and Zdeborová L 2017 Streaming bayesian inference: theoretical limits and mini-batch approximate message-passing *2017 55th Annual Allerton Conf. on Communication, Control and Computing* (*Allerton*) pp 1048–55
McCloskey M and Cohen N J 1989 Catastrophic interference in connectionist networks: the sequential learning problem *The Psychology of Learning and Motivation* vol 24 (Amsterdam: Elsevier) pp 109–65
Mézard M 2017 Mean-field message-passing equations in the hopfield model and its generalizations *Phys. Rev.* E **95** 022117
Mézard M and Montanari A 2009 *Information, Physics and Computation* (Oxford, NY: Oxford University Press)
Mézard M, Parisi G and Virasoro M A 1987 *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications* vol 9 (Singapore: World Scientific)
Minka T P 2001 Expectation propagation for approximate bayesian inference *Proc. 17th Conf. on Uncertainty in Artificial Intelligence* (*UAI'01*) (*San Francisco, CA*) (Morgan Kaufmann Publishers) pp 362–9
Parker J T, Schniter P, and Cevher V 2013 Bilinear generalized approximate message passing *CoRR* (arXiv:1310.2632)
Parker J T, Schniter P and Cevher V 2014 Bilinear generalized approximate message passing-part I: derivation *IEEE Trans. Signal Process.* **62** 5839–53
Pearl J 1982 Reverend Bayes on inference engines: a distributed hierarchical approach (Cognitive Systems Laboratory, School of Engineering and Applied Science)
Peierls R 1936 On ising's model of ferromagnetism *Math. Proc. Camb. Phil. Soc.* **32** 477–81
Pittorino F, Lucibello C, Feinauer C, Perugini G, Baldassi C, Demyanenko E and Zecchina R 2021 Entropic gradient descent algorithms and wide flat minima *Int. Conf. on Learning Representations*
Rangan S, Schniter P and Fletcher A K 2019 Vector approximate message passing *IEEE Trans. Inf. Theory* **65** 6664–84
Rao R P N 2007 Neural models of Bayesian belief propagation *Bayesian Brain: Probabilistic Approaches to Neural Coding* (Cambridge, MA: MIT Press) pp 239–67
Robins A 1995 Catastrophic forgetting, rehearsal and pseudorehearsal *Connect. Sci.* **7** 123–46
Satorras V G and Welling M 2021 Neural enhanced belief propagation on factor graphs *Int. Conf. on Artificial Intelligence and Statistics* (PMLR) pp 685–93
Soudry D, Hubara I and Meir R 2014 Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights *Advances in Neural Information Processing Systems* vol 1 p 2
Soudry D, Hubara I and Meir R 2014 Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights *Advances in Neural Information Processing Systems* vol 27, ed Z Ghahramani, M Welling, C Cortes, N Lawrence and K Q Weinberger (Red Hook, NY: Curran Associates)
Stamatescu G, Gerace F, Lucibello C, Fuss I and White L B 2020 Critical initialisation in continuous approximations of binary neural networks (available at: https://openreview.net/forum?id=rylmoxrFDH)
Sung Y-L, Nair V and Raffel C 2021 Training neural networks with fixed sparse masks (arXiv:2111.09839)
Tung F and Mori G 2018 Clip-q: deep network compression learning by in-parallel pruning-quantization *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 7873–82
Wu A, Nowozin S, Meeds E, Turner R E, Hernandez-Lobato J M and Gaunt A L 2018 Deterministic variational inference for robust bayesian neural networks (arXiv:1810.03958)
Yedidia J S, Freeman W T and Weiss Y 2003 Understanding belief propagation and its generalizations *Exploring Artificial Intelligence in the New Millennium* (San Francisco, CA: Morgan Kaufmann Publishers) pp 239–69
Zdeborová L and Krzakala F 2016 Statistical physics of inference: thresholds and algorithms *Adv. Phys.* **65** 453–552
Zenke F, Poole B and Ganguli S 2017 Continual learning through synaptic intelligence *Int. Conf. on Machine Learning* (PMLR) pp 3987–95
Zou Q, Zhang H and Yang H 2021 Multi-layer bilinear generalized approximate message passing *IEEE Trans. Signal Process.* **69** 4529–43