SoftHebb: Bayesian inference in unsupervised Hebbian soft winner-take-all networks

Timoleon Moraitis1,*, Dmitry Toichkin2, Adrien Journé1, Yansong Chua1 and Qinghai Guo1,3

1 Huawei Technologies, Zurich Research Center, Zurich, Switzerland
2 Moscow, Russia
3 Huawei Technologies, Advanced Computing and Storage Lab, Shenzhen, People's Republic of China
* Authors to whom any correspondence should be addressed.
E-mail: timoleon.moraitis@huawei.com and guoqinghai@huawei.com

Keywords: adversarial robustness, local plasticity, alternative to backpropagation, neuromorphic efficiency, Bayesian generative model, online unsupervised Hebbian learning, cortical learning theory

Abstract

Hebbian plasticity in winner-take-all (WTA) networks is highly attractive for neuromorphic on-chip learning, owing to its efficient, local, unsupervised, and on-line nature. Moreover, its biological plausibility may help overcome important limitations of artificial algorithms, such as their susceptibility to adversarial attacks, and their high demands for training-example quantity and repetition. However, Hebbian WTA learning has found little use in machine learning, likely because it has been missing an optimization theory compatible with deep learning (DL). Here we show rigorously that WTA networks constructed by standard DL elements, combined with a Hebbian-like plasticity that we derive, maintain a Bayesian generative model of the data. Importantly, without any supervision, our algorithm, SoftHebb, minimizes cross-entropy, i.e. a common loss function in supervised DL. We show this theoretically and in practice. The key is a 'soft' WTA where there is no absolute 'hard' winner neuron. Strikingly, in shallow-network comparisons with backpropagation, SoftHebb shows advantages beyond its Hebbian efficiency. Namely, it converges in fewer iterations, and is significantly more robust to noise and adversarial attacks. Notably, attacks that maximally confuse SoftHebb are also confusing to the human eye, potentially linking human perceptual robustness, with Hebbian WTA circuits of cortex. Finally, SoftHebb can generate synthetic objects as interpolations of real object classes. All in all, Hebbian efficiency, theoretical underpinning, cross-entropy-minimization, and surprising empirical advantages, suggest that SoftHebb may inspire highly neuromorphic and radically different, but practical and advantageous learning algorithms and hardware accelerators.

1. Introduction

State-of-the-art (SOTA) artificial neural networks (ANNs) achieve impressive results in a variety of machine intelligence tasks (Sejnowski 2020). However, they largely rely on mechanisms that diverge from the original inspiration from biological neural networks (Bengio et al. 2015, Illing et al. 2019). As a result, only a small part of this prolific field also contributes to computational neuroscience fact, biological implausibility is also an important issue for machine intelligence. Despite their impressive performance, ANNs often neglect properties that are present in biological systems, and these properties could offer a path to the next generation of artificial intelligent systems (Zador et al. 2022). Namely, neuromorphic computing has been advancing machine intelligence in energy efficiency, and recent evidence shows that it improves also conventional metrics of SOTA performance such as accuracy, reward, or speed. For example, spike-based models achieve processing speed and energy efficiency through the imitation of biological neuronal activations, without trading off performance (Jeffares et al. 2022), or with minimal trade-offs (Bittar and Garner 2022); short-term plasticity (STP) improves the performance of neural networks in dynamic tasks such as video processing, navigation, robotics, and video games (Moraitis et al. 2020, Garcia Rodriguez et al. 2022).
Schematic: summary of the key properties of SoftHebb, contrasted with backpropagation. (a) Unsupervised: SoftHebb uses no supervision by top-down signals, such as cross-entropy ('X-ent'). (b) X-ent minimization & speed: nevertheless SoftHebb minimizes the cross-entropy loss under certain assumptions. Moreover, it converges faster than backpropagation in number of learning iterations. (c) Classification: SoftHebb’s unsupervised algorithm can be used to cluster an input dataset into classes, either on its own through its Bayesian inference of the classes as hidden causes of the input, or with an added supervised linear classifier. E.g. with an unperturbed image of a handwritten digit ‘4’ (middle row), networks trained with both backprop (top) and SoftHebb (bottom) perform well and recognize the digit correctly (green circles). (d) Noise robustness: when Gaussian noise is added to the input at inference, the backprop-trained network misclassifies the digit (red circle), whereas SoftHebb is robust.

(c) Adversarial attack robustness: with a white-box adversarial attack, the digit’s pixels are perturbed to maximize the loss of each specific network, with its parameters known to the attacker. Note that the images (middle row) have subtle changes compared to the original input. The attack results in a different image targeting each network. The attack is successful for the backprop-trained network, which misclassifies the digit (red circle) as a ‘zero’. SoftHebb on the other hand remains robust. (f) Adversarial attack deflection: an attacker perturbs the input of class ‘four’ targeting a network output of class ‘zero’. The attacker chooses an intensity that suffices to ascertain the attack’s success with a high probability (see figure 5). The attack of the backprop-trained network succeeds while the image still appears as a digit ‘four’ the case of SoftHebb, the attacker must truly generate an image of a digit ‘zero’ (green circle) to succeed, i.e. SoftHebb deflects this adversarial attack attempt.

1.1. Inefficiencies of conventional DL

Several limitations of conventional DL appear to be in contrast with some biological learning processes, and could therefore potentially be addressed by neuromorphic learning algorithms. For instance, ANN training often demands very large and labelled datasets, which are costly to generate. When labels are unavailable, self-supervised learning schemes exist, where supervisory error signals generated by the network itself are exploited and backpropagated from the output towards the input to update the network’s parameters (Goodfellow et al 2014, Devlin et al 2018, Chen et al 2020, Bardes et al 2021, Scherr et al 2022). However, this global propagation of signals in deep networks introduces another limitation. Namely, it prevents the implementation of efficient distributed computing hardware that would be based on only local signals from neighbouring physical nodes in the network, it requires teaching currents to flow throughout the network, and is in contrast to the local synaptic plasticity rules that partly govern biological learning. Several pieces of work have been addressing parts of the biological implausibility and hardware-inefficiency of backpropagation (BP) in ANNs (Crick 1989, Bengio et al 2015, Lillicrap et al 2016, Nokland 2016, Guerguev et al 2017, Pfeiffer and Pfeil 2018, Illing et al 2019, Millidge et al 2020, Pogodin and Latham 2020, Payeur et al 2021, Pogodin et al 2021), such as requirements of exactly symmetric forward and backward weights or the waiting time caused by the network’s forward-backward pass between two training updates in a layer. These are known as the weight–transport (Grossberg 1987, Lillicrap et al 2016) and update-locking (Czarnecki et al 2017, Frenkel et al 2021) problems of BP. Recently, an approximation to BP that is mostly Hebbian, i.e. relies mostly on pre- and post-synaptic activity of each synapse, has been achieved by reducing the global error requirements to 1-bit information (Pogodin and Latham 2020). Two schemes that further localize the signal that is required for a weight update are equilibrium propagation (Scellier and Bengio 2017) and predictive coding (Millidge et al 2020). Both methods approximate BP through Hebbian-like learning, by delegating the global aspect of the computation, from a global error signal, to a global convergence of the network state to
an equilibrium. This equilibrium is reached through several iterative steps of feed-forward and feed-back communication throughout the network, before the ultimate weight update by one training example. The biological plausibility and hardware-efficiency of this added iterative process of signal propagation are open questions that begin to be addressed (Ernoult et al 2020). Therefore, even though there has been significant progress in dealing with some of the inefficiencies and biological implausibilities of BP, this has not been entirely possible, because these approaches aim to approximate BP, rather than learn with a radically different mechanism.

1.2. Adversarial attacks of ANNs. Deflection by humans

Moreover, learning through BP, and presumably also its approximations, has another indication of biological implausibility, which also significantly limits ANN applicability. Namely, it produces networks that are confused by small adversarial perturbations of the input, which are imperceptible by humans. It has recently been proposed that a defence strategy of ‘deflection’ of adversarial attacks may be the ultimate solution to that problem (Qin et al 2020). Through this strategy, to cause confusion in the network’s inferred class, the adversary is forced to generate such a changed input that it really belongs to the distribution mode of a different input class figure 1(f). Intuitively, but also strictly by definition, this deflection is achieved if a human assigns to the perturbed input the same label that the network does. Deflection of adversarial attacks in ANNs has been demonstrated by an elaborate scheme that is based on detecting the attacks (Qin et al 2020). However, the human ability to deflect adversarial perturbations likely does not rely on detecting them, but rather on effectively ignoring them, making the deflecting type of robustness an emergent property of biological computation rather than a defence mechanism. The principles that underlie this biological robustness are unclear, but it might emerge from the distinct algorithms that govern learning in the brain.

1.3. Hebbian winner-take-all (WTA)

Therefore, what is missing is a biologically plausible model that can learn from fewer data points, without labels, through local plasticity, and without feedback from distant layers figure 1(a). This model could then be tested for emergent adversarial robustness figure 1(c), (e) and deflection of adversarial attacks figure 1(f). A good candidate category of biological networks and learning algorithms is that of competitive learning. Neurons that compete for their activation through lateral inhibition are a common connectivity pattern in the superficial layers of the cerebral cortex (Binzegger et al 2004, Douglas and Martin 2004). This pattern is described as WTA, because competition suppresses activity of weakly activated neurons, and emphasizes strong ones. WTA competition is generally categorized into two types, namely hard WTA, where the winning neuron is the only one active, and soft WTA, where the non-winning neurons are not fully suppressed (Binns et al 2014). Combined with Hebbian-like plasticity rules, i.e. update rules based on correlated pre- and post-synaptic activity, WTA connectivity gives rise to competitive-learning algorithms. These networks and learning schemes have been long studied (Von der Malsburg 1973) and a large literature based on simulations and analyses describes their functional properties. A WTA neuronal layer, depending on its specifics, can restore missing input signals (Rutishauser et al 2011, Diehl and Cook 2016), perform decision making i.e. winner selection (Fahnnloser et al 1999, Maass 2000, Rutishauser et al 2011), and generate oscillations such as those that underlie brain rhythms (Cannon et al 2014). Perhaps more importantly, its neurons can learn to become selective to different input patterns, such as orientation of visual bars in models of the primary visual cortex (Von der Malsburg 1973), MNIST handwritten digits (Nessler et al 2013, Diehl and Cook 2015, Krotov and Hopfield 2019), CIFAR-10 objects (Krotov and Hopfield 2019), spatiotemporal spiking patterns (Nessler et al 2013), and can adapt dynamically to model changing objects (Moraits et al 2020). The WTA model is indeed biologically plausible, Hebbian plasticity is local, and learning is input-driven, relying on only feed-forward communication of neurons—properties that seem to address several of the limitations of ANNs. However, the model’s applicability is limited to simple tasks. That is partly because the related theoretical literature remains surprisingly unsettled, despite its long history, and the strong and productive community interest (Földiak and Fidlir 1989, Sanger 1989, Földiak 1990, Linsker 1992, Bell and Sejnowski 1995, Olshausen and Field 1996, 1997, Lee et al 1999, Nessler et al 2013, Hu et al 2014, Pehelevan and Chklovskii 2014, 2015, Pehelevan et al 2017, Isomura and Toyoizumi 2018).

1.4. Necessity for new theoretical foundation of Hebbian WTA learning

A very relevant theory in this direction was described by Nessler et al (2009, 2013). That work showed that WTA circuits implement Bayesian computation and, combined with local plasticity, they implement a type of expectation-maximization. The specific plasticity rule for a synapse connecting a presynaptic neuron $i$ with activation $x_i$ to a postsynaptic neuron $k$ with WTA output $y_k$ in that case was
The key steps in our derivations are the following. First we define the assumptions about the input and we define a parametrized Bayesian model that will form the backbone of the neural network. Then we derive the theory underpinning SoftHebb. The resulting ML-theoretic probabilistic model and the equivalent neural network are summarized in figure 2.1, whereas a succinct description of the learning algorithm is provided in algorithm 1.

**2. Theoretical results**

We will now derive the theory underpinning SoftHebb. The resulting ML-theoretic probabilistic model and the equivalent neural network are summarized in figure 2, whereas a succinct description of the learning algorithm is provided in algorithm 1.

### 2.1. Overview of the derivation

The goal of the derivation is to find a WTA neural network and a plasticity rule that optimizes the probabilistic model represented by the network, for a given input distribution. The aimed optimization is specifically the minimization of the KL-divergence between the model distribution and the input distribution, so we aim to derive a plasticity rule that can learn the parameters that achieve this minimum. The key steps in our derivations are the following. First we define the assumptions about the input and we define a parametrized Bayesian model that will form the backbone of the neural network. Then we

\[
\Delta w_{ik}^{(\text{Nessler})} = \begin{cases} 
\eta (e^{-w_k} - 1), & \text{if } x_i = 1 \text{ and } y_k = 1 \\
-\eta, & \text{if } x_i = 0 \text{ and } y_k = 1 \\
0, & \text{if } y_k = 0. 
\end{cases}
\]

(1)
Algorithm 1. SoftHebb learning.

1: for all neurons $k \in \{1, 2, \ldots, K\}$ in the layer, do
2:   initialize random weights and biases
3: end for
4: for all training examples $x$ do
5:   for all neurons $k$ do
6:     calculate preactivation $u_k = w_k x$
7:   end for
8: for all neurons $k$ do
9:   Optional: calculate activation $q'_k = h(u_k + w_{0k})$ [e.g. $h(x) = \exp(x)$]
10:  calculate posterior (i.e. normalized activation) $y_k$ [e.g. Softmax]
11:  end for
12: for all neurons $k$ do
13:   for all synapses $i$ do
14:     calculate weight change $\Delta w_{ik}^{\text{SoftHebb}} = \eta \cdot y_k \cdot (x_i - u_i w_k)$
15:     update weight $w_{ik} = w_{ik} + \Delta w_{ik}^{\text{SoftHebb}}$
16:   end for
17: calculate bias change $\Delta w_{0k}^{\text{SoftHebb}} = \eta y_k (y_k - e^{w_{0k}})$
18: update bias $w_{0k} = w_{0k} + \Delta w_{0k}^{\text{SoftHebb}}$
19: end for
20: end for

determine the optimal parameters of the probabilistic model, i.e. those that imply minimum KL divergence from the input. Subsequently, we describe how the model is equivalent to a soft WTA network. Then we describe SoftHebb’s plasticity rule, and we show that the parameters that we found as optimal are the plasticity rule’s equilibrium, which shows that the plasticity rule updates the network to maintain its Bayesian model optimally. Finally, we describe how given certain assumptions the network minimizes cross-entropy from the input labels, despite the absence of the labels or other supervision. The detailed proofs of the theorems are provided in section 4.

Definition 2.1 (The input assumptions). Each observation $x \in \mathbb{R}^n$ is generated by a hidden ‘cause’ $C$ from a finite set of $K$ possible such causes: $C \in \{C_k, \forall k \leq K \in \mathbb{N}\}$. Therefore, the data is generated by a mixture of the probability distributions attributed to each of the $K$ classes $C_k$:

$$p(x) = \sum_{k=1}^{K} p(x|C_k)P(C_k).$$  

$x$ is a vector quantity, and its dimensions, i.e. components $x_i$, are conditionally independent from each other, i.e. $p(x) = \prod_{i=1}^{n} p(x_i)$. The number $K$ of the true causes or classes of the data is assumed to be known.
The term ‘cause’ is used here in the sense of causal inference. It is important to emphasize that the true cause of each input is hidden, i.e. not known the case of a labelled dataset, labels commonly correspond to causes, and the labels are deleted before presenting the training data to the model. We choose a mixture model that corresponds to the data assumptions but is also interpretable in neural terms (section 2.2):  

**Defination 2.2 (The generative probabilistic mixture model).** We consider a mixture model distribution $q(x) = \sum_{k=1}^{K} q(x|C_k) Q(C_k)$, approximating the data distribution $p$. We choose specifically a mixture of exponentials and we parametrize $Q(C_k; w_{0k})$ also as an exponential, specifically:

$$q(x|C_k; w_{0k}) = e^{w_k \cdot x} \frac{1}{\Gamma(K)}, \forall k$$

$$Q(C_k; w_{0k}) = e^{w_k}, \forall k.$$  

In addition, the parameter vectors are subject to the normalization constraints: $||w_k|| = 1, \forall k$, and $\sum_{k=1}^{K} e^{w_k} = 1$.

The model we have chosen is a reasonable choice because it factorizes similarly to the data of definition 2.1:

$$q_k := q(x|C_k; w_k) = \prod_{i=1}^{n} q(x_i|C_k; w_k)$$

$$= e^{\sum_{i=1}^{n} w_k \cdot x_{ik}} = e^{w_k},$$

where $w_k = \frac{w_k \cdot x}{||w_k|| \cdot ||x||}$, i.e. the cosine similarity of the two vectors. A similar probabilistic model was used in related previous theoretical work (Nessler et al 2009, 2013, Moraitis et al 2020), but for different data assumptions, and with certain further constraints to the model. Namely, Nessler et al (2009, 2013) considered data that was binary, and created by a population code, while the model was stochastic. These works provide the foundation of our derivation, but here we consider the more generic scenario where data are continuous-valued and input directly into the model, which is deterministic and, as we will show, more compatible with standard ANNs (Moraitis et al 2020), data had particular short-term temporal dependencies, whereas here we consider the distinct case of independent and identically distributed (i.i.d.) input samples. The Bayes-optimal parameters of a model mixture of exponentials can be found analytically as functions of the input distribution's parameters, and the model is equivalent to a soft WTA neural network (Moraitis et al 2020). After describing this, we will prove here that Hebbian plasticity of synapses combined with local plasticity of the neuronal biases sets the parameters to their optimal values.

**Theorem 2.1 (The optimal parameters of the model).** The parameters that minimize the KL divergence of such a mixture model from the data are, for every $k$,

$$\text{opt} w_{0k} = \ln P(C_k)$$

$$\text{and} \quad \text{opt} w_k^* = \frac{\text{opt} w_k}{||\text{opt} w_k||} = \frac{\mu_{p_k}(x)}{||\mu_{p_k}(x)||},$$

where $c \in \mathbb{R}^+$, opt $w_k = c \cdot \mu_{p_k}(x), \mu_{p_k}(x)$ is the mean of the distribution $p_k$, and $p_k := p(x|C_k)$.

In other words, the optimal parameter vector of each component $k$ in this mixture is proportional to the mean of the corresponding component of the input distribution, i.e. it is a centroid of the component addition, the optimal parameter of the model’s prior $Q(C_k)$ is the logarithm of the corresponding component’s prior probability. This theorem’s proof was provided in the supplementary material of Moraitis et al (2020), but for completeness we also provide it in our section 4. These centroids and priors of the input’s component distributions, as well as the method of their estimation, however, are different for different input assumptions, and we will derive a learning rule that provably sets the parameters to their maximum likelihood estimate for the inputs addressed here. The learning rule is a Hebbian type of synaptic plasticity combined with a plasticity for neuronal biases. Before providing the rule and the related proof, we will describe how our mixture model is equivalent to a WTA neural network.

**2.2. Equivalence of the probabilistic model to a WTA neural network**

The cosine similarity between the input vector and each centroid’s parameters underpins the model (equation 5). This similarity is precisely computed by a linear neuron that receives normalized inputs $x^* := \frac{x}{||x||}$ and normalizes its vector of synaptic weights: $w_k^* := \frac{w_k}{||w_k||}$. Specifically, the neuron’s summed weighted input $u_k = w_k^* \cdot x^*$ then determines the cosine similarity of an input sample to the weight vector,
thus computing the likelihood function of each component of the input mixture (equation (3)). It should be noted that even though $u_k$ depends on the weights of all input synapses, the weight values of other synapses do not need to be known to each updated synapse. Therefore, in the SoftHebb plasticity rule that we will present (equation (9)), the term $u_k$ is a local, postsynaptic variable that does not undermine the locality of the plasticity. The bias term of each neuron can store the parameter $w_0$ of the prior $Q(C_k; w_0)$. Based on these, it can also be shown that a set of $K$ such neurons can actually compute the Bayesian posterior, if the neurons are connected in a configuration that implements softmax. Softmax has a biologically-plausible implementation through lateral inhibition (divisive normalization) between neurons (Nessler et al. 2009, 2013, Moraitis et al. 2020). Specifically, based on the model of definition 2.2, the posterior probability is

$$Q(C_k|x; w) = \frac{e^{u_k + w_{0k}}}{\sum_{i=1}^{K} e^{u_i + w_{0i}}}. \tag{8}$$

But in the neural description, $u_k + w_{0k}$ is the activation of the $k$th linear neuron. That is, equation (8) shows that the result of Bayesian inference of the hidden cause from the input $Q(C_k|x)$ is found by a softmax operation on the linear neural activations this equivalence, we will be using $y_k := Q(C_k|x; w)$ to symbolize the softmax output of the $k$th neuron, i.e. the output after the WTA operation, interchangeably with $Q(C_k|x)$. It can be seen in equation (8) that the probabilistic model has one more, alternative, but equivalent neural interpretation. Specifically, $Q(C_k|x)$ can be described as the output of a neuron with exponential activation function (numerator in equation (8)) that is normalized by its layer’s total output (denominator). This is equally accurate, and more directly analogous to the biological description (Nessler et al. 2009, 2013, Moraitis et al. 2020). This shows that the exponential activation of each individual neuron $k$ directly equals the $k$th exponential component distribution of the generative mixture model (equation (5)). Therefore, the softmax-configured linear neurons, or equivalently, the normalized exponential neurons, fully implement the generative model of definition 2.2, and also infer the Bayesian posterior probability given an input and the model parameters. However, the problem of calculating the model’s parameters from data samples is a difficult one, if the input distribution’s parameters are unknown the next sections we will show that this neural network can find these optimal parameters through Bayesian inference, in an unsupervised and on-line manner, based on only local Hebbian plasticity. It should be noted that the number of neurons $K$ must be chosen in advance, an aspect that may be regarded as human supervision.

### 2.3. A Hebbian rule that optimizes the weights

Several Hebbian-like rules exist and have been combined with WTA networks. For example, in the case of stochastic binary neurons and binary population-coded inputs, it has been shown that weight updates with an exponential weight-dependence find the optimal weights (Nessler et al. 2009, 2013). Oja’s rule is another candidate (Oja 1982). An individual linear neuron equipped with this learning rule finds the first principal component of the input data (Oja 1982). A variation of Oja’s rule combined with hard-WTA networks and additional mechanisms has achieved good experimental results performance on classification tasks (Krotov and Hopfield 2019), but lacks the theoretical underpinning that we aim for. Here we propose a Hebbian-like rule for which we will show it optimizes the soft WTA’s generative model. The rule is similar to Oja’s rule, but considers, for each neuron $k$, both its linear weighted summation of the inputs $u_k$, and its nonlinear output of the WTA $y_k$:

$$\Delta w_{ik}^{\text{SoftHebb}} := \eta \cdot y_k \cdot (x_i - u_k w_{0k}), \tag{9}$$

where $w_{ik}$ is the synaptic weight from the $i$th input to the $k$th neuron, and $\eta$ is the learning rate hyperparameter. As can be seen, all involved variables are local to the synapse, i.e. only indices $i$ and $k$ are relevant. No signals from distant layers, from non-perisynaptic neurons, or from other synapses are involved. By solving the equation $E[\Delta w_{ik}] = 0$ where $E[\cdot]$ is the expected value over the input distribution, we can show that, with this rule, there exists a stable equilibrium value of the weights, and this equilibrium value is an optimal value according to theorem 2.3:

**Theorem 2.2.** The equilibrium weights of the SoftHebb synaptic plasticity rule are

$$w_{ik}^{\text{SoftHebb}} = c \cdot \mu_k (x) \equiv \mu_k w_{0k}, \tag{10}$$

where $c = \frac{1}{||\mu_k (x)||}$. The proof is provided in section 4. Therefore, our update rule (equation (9)) optimizes the neuronal weights. Moreover, the following normalization theorem is proven in section 4.
The equilibrium weights of the SoftHebb synaptic plasticity rule of equation (9) are implicitly normalized by the rule to a vector of length 1.

This then constrains the convergence of SoftHebb to a unique solution. Moreover, it can be used as a proxy for measuring the progress of convergence (see section 3.2 and figure 4(d)).

2.4. Local learning of neuronal biases as Bayesian priors
For the complete optimization of the model, the neuronal biases \( w_{0k} \) must also be optimized to satisfy equation (6), i.e. to optimize the Bayesian prior belief for the probability distribution over the \( K \) input causes. For the biases, we define the following rate-based rule inspired from the spike-based bias rule of Nessler et al (2013):

\[
\Delta w_{0k}^{\text{SoftHebb}} = \eta e^{-w_{0k}} (y_k - e^{w_{0k}}).
\] (11)

With the same technique we used for theorem 2.2, we also provide proof in section 4 that the equilibrium of the bias with this rule matches the optimal value \( w_{0k}^{\text{opt}} = \ln P(C_k) \) of theorem 2.3:

Theorem 2.4. The equilibrium biases of the SoftHebb bias learning rule are

\[
w_{0k}^{\text{SoftHebb}} = \ln P(C_k) = w_{0k}^{\text{opt}}.
\] (12)

2.5. Alternate activation functions. Relation to Hard WTA
The model of definition 2.2 uses for each component \( p(x|C_k) \) an exponential probability distribution with a base of Euler’s e, equivalent to a model using similarly exponential neurons (section 2.2). Depending on the task, different probability distribution shapes, i.e. different neuronal activation functions, may be better models. This is compatible with our theory (see section 4.2). Firstly, the base of the exponential activation function can be chosen differently, resulting in a softmax function with a different base, such that equation (8) becomes more generally

\[
Q^{(\text{alt})}(C_k|x) = \frac{b^{u_k+w_{0k}}}{\sum_{l=1}^{K} b^{u_l+w_{0l}}}.\]

This is equivalent to temperature scaling (Hinton et al 2015), a mechanism that also maintains the probabilistic interpretation of the softmax output. The alternate-base version can also be implemented by a normalized layer of exponential neurons, which are compatible with our theoretical derivations and the optimization by the plasticity rule of equation (9). Interestingly, this integrates the hard WTA into the SoftHebb framework. Specifically, hard WTA is a special case of SoftHebb with an infinite base of e, under the assumption of an exponential probability distribution with an exponential activation function.

This understanding and comparison is important because hard WTA is often chosen to underlie Hebbian learning (Amato et al 2019, Grinberg et al 2019, Krotov and Hopfield 2019, Lagani et al 2021).

Moreover, we show in section 4 that other activation functions than exponential or softmax are also supported. Soft WTA models can be constructed by rectified linear units (RelUs) or in general by neurons with any non-negative monotonically increasing activation function, and their weights are also optimized by the same plasticity rule.

2.6. Cross-entropy minimization without supervision
2.6.1. SoftHebb as a discriminator
Even though SoftHebb’s soft WTA is a generative and not a discriminative model, i.e. it models the distribution \( p(x) \) as \( q(x|w) \), it can also be used for discrimination of the input classes \( C_k \), i.e. classification, using Bayes’ theorem:

\[
Q(C_k|x;w) = \frac{q(x|C_k;w)}{q(x;w)}.
\] (14)
2.6.2. SoftHebb minimizes cross-entropy of the true causes

It can be shown that while the generative model is optimized by SoftHebb, its discriminative aspect is also optimized. Specifically, the algorithm minimizes in expectation $H_C^D := H(P(C|x), Q(C|x))$, i.e. the cross-entropy of the causes $Q(C_k|x)$ that it infers, from the true causes of the data $P(C_k|x)$:

$$w^{\text{SoftHebb}} = \arg\min_w H_C^D.$$  \hspace{1cm} (15)

That corollary follows from the proof of theorem 2.3. The theorem's proof involved showing that SoftHebb minimizes the KL divergence $D_{KL}(p(x)||q(x;w))$ of the model $q(x;w)$ from the data $p(x)$. But KL divergence is the difference of cross-entropy $H(p(x), q(x;w))$ minus the entropy $S_p$ of the data: $D_{KL}(p(x)||q(x;w)) = H(p(x), q(x;w)) - S_p$.

Therefore, since the entropy $S_p$ of the data distribution does not depend on the optimized model parameters, we conclude that minimizing KL divergence through SoftHebb implies also minimizing cross-entropy $H(p(x), q(x;w))$.

2.6.3. Disparity between direct cause and label

So far we have shown that beyond the generative model, SoftHebb also optimizes the discriminative ability of the Bayesian model, but specifically pertaining to discriminating among the direct causes of the data. However, ML practice is frequently interested in categorizations that may not correspond to the true causes that directly relate to the data other words, in tasks of classification into a set of class labels, the label set—let us denote this by $L$—chosen by a human supervisor may not correspond exactly to the true and single cause $C$ that generates the data points, which is what SoftHebb's unsupervised process learns to infer. This difference is depicted in figure 3(a). Nevertheless, SoftHebb's process does minimize cross-entropy with respect to the labels too, as long as the label set is reasonable—which we will now formalize. For example, consider the commonly used benchmark dataset MNIST. The ten labels indicating the ten decimal digits do not correspond exactly to the true cause of each example image reality, the direct cause $C$ generating each MNIST example in the sense implied by causal inference is not the digit cause on its own, which corresponds to the MNIST label, but rather it is a combination of the digit $L$ with one of many handwriting styles $S$. That is, the probabilistic model is such that the direct cause $C$ of each sample is dual, i.e. there exists a digit $L_i (i \in \{0,1,\ldots,9\})$ and a style $S_i$ that jointly compose the direct cause (see also figure 3):

$$P(C_k) := P(C = C_k) = P(L_i)P(S_i).$$ \hspace{1cm} (16)

2.6.4. SoftHebb minimizes cross-entropy of the labels

This relationship between the labels $L$ and the direct causes $C$ (see figure 3(a), red arrow) can be written as

$$P(L_i) = \sum_k P(L_i|C_k)P(C_k).$$ \hspace{1cm} (17)

Therefore, SoftHebb's $Q(C)$ implicitly defines a model $Q(L)$:

$$Q(L_i) := \sum_k P(L_i|C_k)Q(C_k).$$ \hspace{1cm} (18)

In equations (17) and (18), the term $P(L_i|C_k)$ is fixed by the data-generation process. As a consequence, to minimize cross-entropy between $P(L)$ and $Q(L)$ is to minimize cross-entropy between $P(C)$ and $Q(C)$, which SoftHebb does (see equation (15)).

---

**Figure 3.** (a) A causal graph where the single direct hidden cause $C$ that generates the observed data $x$ is itself affected by two root causes $L$ and $S$. (b) The same graph annotated, with root causes as they correspond to the MNIST dataset.
Therefore, SoftHebb minimizes cross-entropy $H_Q^C$ between its implicit model of the labels and the true labels $L$:

\[
\text{Cross-entropy minimization: } w_{\text{SoftHebb}} = \arg \min_w H_Q^C = \arg \min_w H_Q^C.
\] (19)

This is remarkable, given that SoftHebb never accesses the labels or any other supervisory signal.

2.6.5. Measuring loss minimization: post-hoc cross-entropy

It has not been obvious how to correctly measure the loss of an unsupervised WTA network during the learning process, since the ground truth for causes $C$ is often not available, even if labels $L$ are available, as we reasoned above. Using our theoretical result about cross-entropy minimization, here we propose a method for measuring the loss minimization during training in spite of this cause-label mismatch. To obtain $Q(L|x)$ of equation (18) and measure the cross-entropy, the causal structure $P(L|C)$ (see figure 3, red) is missing, but it can be represented by a supervised classifier $Q_L(L|Q(C|x))$ of SoftHebb’s outputs, trained using the labels $L$. Therefore, by (a) an unsupervised training of SoftHebb, then (b) training a supervised classifier on top, and finally (c1) repeating the training of SoftHebb with the same initial weights and ordering of the training inputs as in step (a), while (c2) measuring the trained classifier’s loss, we can observe the cross-entropy loss $H_Q^C$ of SoftHebb while it is being minimized, and infer that $H_Q^C$ is also minimized (equation (19)). We call this the post-hoc cross-entropy method, and it enables evaluation of the learning process in a theoretically sound manner during experimentation (see section 3.2 and figure 4(c)).

3. Experimental results

We implemented the theoretical SoftHebb model in simulations and tested it in the task of learning to classify MNIST handwritten digits. The network received the MNIST frames normalized by their Euclidean norm, while the plasticity rule that we derived updated its weights and biases in an unsupervised manner. We used $K = 2000$ neurons. First we trained the network for 100 epochs, i.e. randomly ordered presentations of the 60,000 training digits. Each training experiment was repeated five times with varying random initializations and input order. We report the mean and standard deviation of the resulting accuracies.

Inference of the input labels by the WTA network of 2000 neurons was performed in two different ways. The first approach is single-layer, where, after training the network, we assigned a label to each of the 2000 neurons, in a standard approach that is used in unsupervised clustering. Namely, for each neuron, we found the label of the training set that makes it win the WTA competition most often. In this single layer approach, this is the only time when labels were used, and at no point were weights updated using labels. The second approach was two-layer and based on supervised training of a perceptron, i.e. linear classifier on top of the WTA layer. The classifier layer was trained with the Adam optimizer (Kingma and Ba 2015) and cross-entropy loss for 100 epochs, while the previously-trained WTA parameters were frozen. The hyperparameters that we used are provided in table 1.

3.1. Indicative accuracies in standard setting

SoftHebb achieved an accuracy of $96.31 \pm 0.06\%$ and $97.80 \pm 0.02\%$ in its one- and two-layer form respectively. To test the strengths of the soft-WTA approach combined with training the priors through biases, which makes the network Bayesian, we also trained the weights of a hard-WTA network, i.e. a model equivalent to SoftHebb with an infinite base in the softmax. The SoftHebb model slightly outperformed the special case of hard WTA (figure 4(a)), especially in the one-layer case where the supervised 2nd layer cannot compensate for the drop in accuracy. Speed comparisons reveal that this is due to a faster convergence by SoftHebb in terms of learning examples (see section 3.2 and figure 4(d)). As an indicative baseline, we also trained a multi-layer perceptron (MLP) with one hidden layer of also 2000 neurons, exhaustively with end-to-end BP and tested it. This was expected to perform significantly better and indeed it reached an accuracy of $98.65 \pm 0.06\%$ (figure 4(a), horizontal dashed line). This is not surprising, due to end-to-end training, supervision, and the MLP being a discriminative model as opposed to a generative model merely applied to a classification task, as SoftHebb is. If the Bayesian and generative aspects that follow from our theory were not required, several mechanisms exist to enhance the discriminative power of WTA networks (Krotov and Hopfield 2019), and even an untrained, random projection layer in place of a trained WTA performs well (Illing et al 2019). Our approach however does have surprising advantages even in a discriminative task, and we report these in the next sections.

3.2. Speed advantages of SoftHebb. Cross-entropy minimization

3.2.1. Learning speed: SoftHebb outperforms hard WTA and BP in the first epoch

Next, we evaluated SoftHebb’s data-efficiency and speed by comparing it to other models during the first training epoch the common, ‘greedy’ training of such networks, layer $L + 1$ is trained only after layer $L$ is
trained on the full dataset and its weights frozen. We trained in this manner a second layer as a supervised classifier for 100 epochs after a single unsupervised learning epoch in the first layer. SoftHebb with a non-zero temperature again slightly outperformed its hard WTA version, showing it extracts superior features (figure 4(b), light-coloured bars). More importantly, we tested a truly single-epoch scenario, without longer training for the second layer. SoftHebb further outperforms hard WTA (figure 4(b), dark coloured bars). Strikingly, it even outperforms end-to-end (e2e) BP in one-epoch accuracy. For the one-epoch experiments, SoftHebb was trained in the fully on-line setting, where each iteration includes a single training example, i.e. a ‘mini-batch’ of size 1, whereas for BP we tuned the batch size and learning rate to the single-epoch setting and found that stochastic gradient descent with a batch size of 4 was best (see section 4.3), which we used.

3.2.2. SoftHebb minimizes cross-entropy, and faster than BP
To empirically confirm the theoretical result about unsupervised minimization of cross-entropy by SoftHebb, we measured the post-hoc cross-entropy loss as derived in section 2.6.5. The resulting training curve is depicted in figure 4(c), for a representative example of a training run. The first observation is that the experiment confirms our theoretical prediction that SoftHebb minimizes the cross-entropy between the model and the labels, which is remarkable, considering that the labels were hidden from the model comparison to the baselines, SoftHebb is faster to converge than its hard-WTA special case and than BP, as measured by number of training iterations/examples. All three algorithms were trained in the fully on-line setting. That is consistent with the accuracy results for the first epoch (section 3.1). This specific manifestation of SoftHebb’s speed advantage is particularly interesting, as it concerns a loss function that backprop minimizes explicitly and has access to it. One might have intuitively expected then that backprop’s supervised minimization would be faster. SoftHebb’s speed advantage might be attributed to its Bayesian nature, which takes account of each data point of evidence optimally.

3.3. SoftHebb improves representation learning over hard WTA
As a further insight into the differences between SoftHebb and hard-WTA learning, we measured throughout learning the number of learned features that lie on a hypersphere with a radius of $1 \pm 0.01$ (R1 features), according to their Euclidean norm. The SoftHebb learning algorithm converges to such a normalization in theory (end of section 4.1, theorem 2.3), and figure 4(d) validates that it does, but also that it does so faster than its hard-WTA special case. This demonstrates SoftHebb’s superiority in unsupervised representation learning and speed, from the perspective of its convergence to the optimal generative Bayesian model, rather than from its discriminative ability.

3.4. Update unlocking
In the simultaneous experiments, the Hebbian networks has an additional important advantage. By using the delta rule for the 2nd layer, each individual training example updates both layers contrast to end-to-end
Figure 5. Noise and adversarial attack robustness of SoftHebb and of backpropagation-trained MLP on MNIST and Fashion-MNIST. The insets show one example from the testing set and its perturbed versions, for increasing perturbations.

(a) SoftHebb is highly robust to noise, even in very noisy settings, in contrast to backprop. (b) MLP’s MNIST accuracy drops to ∼50% by hardly perceptible perturbations (ε = 16/255), while SoftHebb requires visually noticeable perturbations (ε = 64/255) for similar drop in performance. At that degree of perturbation, the MLP has already dropped to zero. SoftHebb deflects the attack: it forces the attacker to produce examples of truly different classes—the original digit ‘4’ is perturbed to look like a ‘0’ (see also figure 8). The attack of the backprop-trained network does not confuse a human observer even at ε = 64/255.

3.5. Robustness to noise and adversarial attacks

3.5.1. Robustness comparison with BP

Based on the Bayesian, generative, and purely input-driven learning nature of the algorithm (as opposed to driven by top-down signals), we hypothesized that SoftHebb may be more robust to input perturbations. Indeed, we tested the trained SoftHebb and MLP models for robustness, and found that SoftHebb is significantly more robust than the backprop-trained MLP, both to added Gaussian noise and to PGD adversarial attacks (see figure 5). PGD (Madry et al 2017) produces perturbations in a direction that ascends the loss of each targeted network, and in size controlled by a parameter ε. It is a white-box attack, that has access to the values of the weights, and is considered one of the strongest adversarial attacks. Strikingly, the Hebbian WTA model has a visible tendency to deflect the attacks, i.e. its most confusing examples actually belong to a perceptually different class (figures 5(b) and 8). This effectively nullifies the attack and was previously shown in elaborate SOTA adversarial-defence models (Qin et al 2020). The attack’s parameters were tuned systematically (see section 4.4).

3.5.2. Robustness comparison with K-means and principal component analysis (PCA)

It is possible that the observed robustness is not unique to SoftHebb and can be reproduced by other unsupervised learning rules. To test this possibility, we compared SoftHebb with PCA and K-means. We used 100 neurons, principal components, or centroids respectively PCA and K-means we then treated the learned coefficients as weight vectors of neurons and applied an activation function, i.e. non-linearity, to then train a supervised classifier on top. First we attempted softmax as in SoftHebb. However the unperturbed test accuracy achieved at convergence was much lower. For example, on MNIST, K-means only reached an accuracy of 53.64% and PCA 28.55%, whereas SoftHebb reached 91.06%. Therefore, we performed the experiment again, but with ReLU activation for K-means and PCA, reaching 90.61% and 82.74% respectively. Then we tested for robustness, revealing that SoftHebb’s learned features are in fact more robust than those of the other two unsupervised algorithms (figure 6). For K-means, the centroids were initialized at positions equal to K, i.e. 100 randomly sampled data points from the MNIST training set. We also experimented with random initialization from a uniform distribution, which did not produce a significantly different behaviour.
3.5.3. Effect of softmax on adversarial robustness

It is possible that the observed robustness of SoftHebb is due to the use of softmax as an activation function. To test this, we compared the SoftHebb network from figure 5 with a same-size backprop-trained two-layer network, but in this case the hidden layer’s summed weighted input was passed through a softmax instead of ReLU before forwarding to the 2nd layer. First, we observed that at convergence, the backprop-trained network did not achieve SoftHebb’s accuracy on either MNIST (94.38%) or Fashion-MNIST (75.90%). Increasing the training time to 300 epochs did not help. Second, as can be seen in figure 7, BP remains significantly less robust than SoftHebb to the input perturbations. This, together with the previous control experiments, suggests that, rather than its activation function, it is SoftHebb’s learned representations that are responsible for the network’s robustness.

3.6. SoftHebb’s generative adversarial properties

The pair of the adversarial attacker with the generative SoftHebb model essentially composes a generative adversarial network (GAN), even though the term is usually reserved for pairs trained in tandem (Goodfellow et al 2014, Creswell et al 2018). As a result, the model could inherit certain properties of GANs. It can be seen that it is able to generate interpolations between input classes (figure 8). The parameter $\epsilon$ of the adversarial attack can control the balance between the interpolated objects. Similar functionality has existed in the realm of GANs (Radford et al 2015), autoencoders (Berthelot et al 2018), and other deep neural networks (Bojanowski et al 2017), but was not known for simple biologically-plausible models.

3.7. Extensibility of SoftHebb: F-MNIST, CIFAR-10

Finally, we performed preliminary tests on two more difficult datasets, namely Fashion-MNIST (Xiao et al 2017), which contains grey-scale images of clothing products, and CIFAR-10 (Krizhevsky et al 2009), which contains RGB images of animals and vehicles. We did not tune the Hebbian networks’ hyper-parameters extensively, so accuracies on these tasks are not definitive but do give a good indication. Future experiments could use for example a recent Bayesian hyperparameter-optimization scheme (Cowen-Rivers et al 2022). On F-MNIST, the SoftHebb model achieved a top accuracy of 87.46% whereas a hard WTA reached a similar
accuracy of 87.49%. A supervised MLP of the same size achieved a test accuracy of 90.55%. SoftHebb’s generative interpolations (figure 8(b)) are reconfirmed on the F-MNIST dataset, as is its robustness to attacks, whereas, with very small adversarial perturbations, the MLP drops to an accuracy lower than the SoftHebb model (dashed lines in figure 5). On CIFAR-10’s preliminary results, the hard WTA and SoftHebb achieved an accuracy of 49.78% and 50.27% respectively every tested dataset, it became clear that SoftHebb learns in fewer iterations than either BP or a hard WTA, by observing the loss and the learned features as in figures 4(c) and (d). The fully-connected SoftHebb layer is applicable to MNIST because the data classes are well-clustered directly in the feature-space of pixels. That is, SoftHebb’s probabilistic model’s assumptions (definition 2.1) are quite valid for this feature space, and increasing the number of neurons for discovering more refined sub-clusters does help. However, for more complex datasets, this approach alone has diminishing returns and multilayer networks will be needed. Towards this, we believe that a convolutional version of SoftHebb will be a key aspect to enable a distributed feature-representation despite the concentrated activation of WTA networks. However, we leave multilayer networks and further experiments for future work (Journé et al 2022), and in the present article we focus on the theoretical foundation and properties of the individual SoftHebb layer.

4. Methods

4.1. Proofs of theoretical results

Proof of Theorem 2.3. A shorter version of this theorem’s proof was provided in the supplementary material of Moraitis et al (2020), but for completeness we also provide it here. The parameters of model \( q \) are optimal \( w = \text{opt}_w \) if they minimize the model’s KL divergence with the data distribution \( p. D_{KL}(p(x)||q(x; w)) \). Because \( p_k := p(x|C_k) \) is independent from \( p_l \) and \( q_k := q(x|C_k; w_k) \) is independent from \( w_l \) for every \( l \neq k \), we can find the set of parameters that minimize the KL divergence of the mixtures, by minimizing the KL divergence of each component \( k \): \( \min D_{KL}(p_k||q_k), \forall k \), and simultaneously setting

\[
P(C_k) = Q(C_k; w_{0k}), \forall k.
\]

(20)

From equation (4) and this last condition, equation (6) of the theorem is proven:

\[
\text{opt}_w w_{0k} = \ln P(C_k).
\]

Further,

\[
\text{opt}_w w_k := \arg \min_w D_{KL}(p_k||q_k)
\]

\[
= \arg \min_w \int p_k \ln \frac{p_k}{q_k} dx
\]
\[
= \arg \min_{w_k} \int_x p(x) \ln p_k(x) - p_k(x) \ln q_k(x) \, dx \\
= \arg \min_{w_k} \int_x -p_k(x) \ln q_k(x) \, dx \\
= \arg \max_{w_k} \int_x p_k(x) \ln q_k(x) \, dx \\
= \arg \max_{w_k} \int_x p_k(x) \ln e^{\mu_k} \, dx \\
= \arg \max_{w_k} \int_x p_k(x) \delta_k \, dx \\
= \arg \max_{w_k} \mu_{p_k}(\cos(w_k, x)) .
\]

Where we used for equation (21) the fact that \( \int_x p_k(x) \ln p_k(x) \, dx \) is a constant because it is determined by the environment’s data and not by the model’s parametrization on \( w \). Equation (22) follows from the definition of \( q_k \).

The result in equation (23) is the mean value of the cosine similarity \( u_k \).

Due to the symmetry of the cosine similarity, and as we prove formally at the end of this theorem’s proof, it follows that

\[
\text{opt } w_k = \arg \max_{w_k} \mu_{p_k}(\cos(w_k, x)) = \arg \max_{w_k} \{ w_k, \mu_{p_k}(x) \} \\
= c \cdot \mu_{p_k}(x), \, c \in \mathbb{R}^+.
\]

Equivalently to equation (24), we must prove that \( \arg \max_{w_k} \mu_{p_k}(\cos(w_k, x)) = w_k(0) \), i.e. that

\[
\arg \max_{w_k} \mu_{p_k}(\cos(w_k, x)) = \mu_{p_k}(\cos(w_k(0), x)) .
\]

For this, we will show that, if the cosine is measured with respect to a weight vector shifted by some \( \delta x \) instead of with respect to \( w_k(0) \), then the mean value decreases, i.e. that

\[
\mu_{p_k}(\cos(w_k(0), x)) \geq \mu_{p_k}(\cos(w_k(0) + \delta x, x)) , \quad \forall \delta x
\]

\[
\iff \mu_{p_k}(\cos(w_k(0), x)) - \mu_{p_k}(\cos(w_k(0) + \delta x, x)) \geq 0 , \quad \forall \delta x.
\]

Let \( \cos_0(x) := \cos(w_k(0), x) \) and the shifted cosine function \( \cos_1(x) := \cos(w_k(0) + \delta x, x) \). Then, equivalently to inequality (28), we must show that

\[
\mu_{p_k}(\cos_0(x)) - \mu_{p_k}(\cos_1(x)) \geq 0
\]

\[
\iff \int_{-\infty}^{\infty} \cos_0(x) p_k(x) \, dx - \int_{-\infty}^{\infty} \cos_1(x) p_k(x) \, dx \geq 0
\]

\[
\iff \int_{-\infty}^{\infty} \Delta \cos(x) p_k(x) \, dx \geq 0
\]

\[
\iff \int_{-\infty}^{\mu_{p_k}(x) + \frac{\pi}{2}} \frac{\Delta \cos p_k(x)}{\sin \frac{\pi}{2}} \, dx + \int_{\mu_{p_k}(x) + \frac{\pi}{2}}^{\infty} \frac{\Delta \cos p_k(x)}{\sin \frac{\pi}{2}} \, dx \geq 0
\]
\[ \Leftrightarrow \int_{-\infty}^{\nu} (\Delta \cos \circ p_k) (x) \, dx - \int_{\nu}^{\infty} (-\Delta \cos \circ p_k) (x) \, dx \geq 0 \]  
\[ \Leftrightarrow \int_{-\infty}^{\nu} (\Delta \cos \circ p_k) (x) \, dx \geq \int_{\nu}^{\infty} (-\Delta \cos \circ p_k) (x) \, dx, \]  

where we have defined \( \Delta \cos := (\cos \circ \nu) (x) \), and \( \nu := \mu_{p_k} (x) + \frac{\delta x}{2} \).

To show inequality (34), it suffices to show that

\[
\begin{cases}
\rho_k (\nu - \eta) \geq \rho_k (\nu + \eta) \\
\text{and } \Delta \cos (\nu - \eta) = -\Delta \cos (\nu + \eta),
\end{cases}
\]

\( \forall \eta : \delta x \cdot \eta \geq 0. \)

The probability density function \( \rho_k (x) \) is symmetrically decreasing, i.e. it is decreasing in the directions that point away from its mean \( \mu_{p_k} (x) \), with respect to any reference point. Therefore, taking as a reference the point \( \nu \), it is indeed true that

\[ \rho_k (\nu - \eta) \geq \rho_k (\nu + \eta), \quad \forall \eta : \delta x \cdot \eta \geq 0. \]  

We will now complete the proof by also showing the second equality that we seek, i.e. \( \Delta \cos (\nu - \eta) = -\Delta \cos (\nu + \eta) \)

\[ \Delta \cos (\nu - \eta) = \cos_0 (\nu - \eta) - \cos_1 (\nu - \eta) \]  
\[ = \cos_0 \left( \mu_{p_k} (x) + \frac{\delta x}{2} - \eta \right) - \cos_1 \left( \mu_{p_k} (x) + \frac{\delta x}{2} - \eta \right). \]  

From the definition of \( w_k^{(0)} \), it is \( w_k^{(0)} = c \mu_{p_k} (x), c \in \mathbb{R}^{+} \) addition, based on the definitions of \( \cos_0 \) and \( \cos_1 \), it is \( \cos_0 (w_k^{(0)} + \epsilon) = \cos_1 (w_k^{(0)} + \delta x + \epsilon), \forall \epsilon \). Moreover, \( \cos_0 \) and \( \cos_1 \) are symmetric around \( w_k^{(0)} \) and \( w_k^{(0)} + \delta x \) respectively by their definition. Therefore, by using these facts and by also choosing \( c = 1 \), it is

\[ \Delta \cos (\nu - \eta) = \cos_0 \left( w_k^{(0)} + \frac{\delta x}{2} - \eta \right) - \cos_1 \left( w_k^{(0)} + \frac{\delta x}{2} - \eta \right) \]  
\[ = \cos_1 \left( w_k^{(0)} + \delta x + \frac{\delta x}{2} - \eta \right) - \cos_1 \left( w_k^{(0)} + \delta x - \frac{\delta x}{2} - \eta \right) \]  
\[ = \cos_1 \left( w_k^{(0)} + \delta x - \frac{\delta x}{2} - \eta \right) - \cos_1 \left( w_k^{(0)} + \delta x + \frac{\delta x}{2} - \eta \right) \]  
\[ = \cos_1 \left( w_k^{(0)} + \delta x + \frac{\delta x}{2} + \eta \right) - \cos_0 \left( w_k^{(0)} - \frac{\delta x}{2} - \eta \right) \]  
\[ = \cos_1 \left( w_k^{(0)} + \frac{\delta x}{2} + \eta \right) - \cos_0 \left( w_k^{(0)} + \frac{\delta x}{2} + \eta \right). \]

Therefore,

\[ \Delta \cos (\nu - \eta) = -\Delta \cos \left( w_k^{(0)} + \frac{\delta x}{2} + \eta \right) \]  
\[ = -\Delta \cos (\nu + \eta), \]

which completes the proof. \( \square \)
Proof of Theorem 2.2. We will find the equilibrium point of the SoftHebb plasticity rule, i.e. the weight $w_k$ that implies $E[\Delta w_k^{\text{SoftHebb}}] = 0$.

We will expand this expected value based on the plasticity rule itself, and on the probability distribution of the input $x$

$$E[\Delta w_k^{\text{SoftHebb}}] = \eta \int_x y_k(x) \cdot (x_i - u_k(x)w_{ik})p(x)dx$$

$$= \eta \int_x y_k(x)(x_i - w_{ik}) \left( \sum_{l=1}^{K} p_l(x)P(C_l) \right) dx$$

$$= \eta \left[ \sum_{l=1}^{K} \int_x x_i y_k(x)p_l(x)P(C_l)dx - \sum_{l=1}^{K} w_{ik} \int_x x y_k(x)p_l(x)P(C_l)dx \right].$$

Based on this, we will now show that

$$w_k = \text{opt} w_k^* = \frac{\mu_{p_l}(x)}{||\mu_{p_l}(x)||} \text{ and } w_{0k} = \text{opt} w_{0k}, \forall k$$

$$\implies E[\Delta w_k^{\text{SoftHebb}}] = 0 \quad \forall i, k.$$ (51)

Using the premise (50), we can take the following steps, where steps (c), (d), and (f) are the main ones, and steps (a), (b), and (e), as well as theorem 4.1 support those.

(a) The cosine similarity function $u_k(x) = xw_k$, as determined by the total weighted input to the neuron $k$, and appropriately normalized, defines a probability distribution centred symmetrically around the vector $w_k$, i.e. $\mu_{u_k}(x) = w_k$ and $u_k(\mu_{u_k} - x) = u_k(\mu_{u_k} + x)$. $w_k$, as premised, is equal to the normalized $\mu_{p_l}(x)$, i.e. the mean of the distribution $p_k(x) = p(x|C_k)$, therefore: $\mu_{u_k}(x) = \mu_{p_l}(x)$.

(b) The soft-WTA version of the neuronal transformation, i.e. the softmax version of the model’s inference is

$$y_k(x) = \frac{\exp(u_k(x))\exp(w_{0k})}{\sum_{i=1}^{K} \exp(u_l(x))\exp(w_{0l})}.$$ (52)

But because of the premise (50) that the parameters of the model $u_k$ are set to their optimal value, it follows that $\exp(u_k(x)) = p_k(x)$ and $\exp(w_{0k}) = P(C_k)$, $\forall k$ (see also theorem 2.4), therefore

$$y_k(x) = \frac{p_k(x)p(C_k)}{\sum_{l=1}^{K} p_l(x)P(C_l)}.$$ (53)

(c) Equation (49) involves twice the function $y_k(x)p_l(x) = \sum_{l=1}^{K} y_k(x)p_l(x)P(C_l)$. Using the above two points, we will now show that this is approximately equal to $y_k(x)p_l(x)P(C_k)$.

1. We assume that the ‘support’ $O_k^p$ of the component $p_k(x)$, i.e. the region where $p_k$ is not negligible, is not fully overlapping with that of other components $p_l$.

   In addition, $O_k^p$ is narrow relative to the input space $\forall k$, because, first, the cosine similarity $u_k(x)$ diminishes fast from its maximum at $\arg\max u_k = w_k$ in case of non-trivial dimensionality of the input space, and, second, $p_k(x) = \exp(u_k(x))$ applies a further exponential decrease.

   Therefore, the overlap $O_k^p \cap O_k^l$ is small, or none, $\forall l \neq k$.

   If $O_k^l$ is the ‘support’ of $y_k$, then this is even narrower than $O_k^p$, due to the softmax.

   As a result, the overlap $O_k^p \cap O_k^l$ of $y_k$ and $p_l$ is even smaller than the overlap $O_k^p \cap O_k^l$ of $p_k$ and $p_l$, $\forall l \neq k$.

2. Because of the numerator in equation (53), the overlap $O_k^p \cap O_k^l$ of $y_k$ and $p_k$ is large.

   Based on these two points, the overlaps $O_k^p \cap O_k^l$ can be neglected for $l \neq k$, and it follows that

$$\sum_{l=1}^{K} y_k(x)p_l(x)P(C_l) \approx y_k(x)p_l(x)P(C_k).$$ (54)
Therefore, we can write equation (49) as

\[ E(\Delta w_k^{(\text{Soft+lebb})}) \approx \eta \left[ \int x y_k(x) p_k(x) P(C_k) \, dx - w_k w_{ik} \int x y_k(x) p_k(x) P(C_k) \, dx \right]. \] (55)

Next, we aim to show that the integrals \( \int x y_k(x) p_k(x) \, dx \) and \( \int x y_k(x) p_k(x) \, dx \) involved in that equation equal the mean value of \( x \) and \( x \) respectively according to the distribution \( p_k \).

To show this, we observe that the integrals are indeed mean values of \( x \) and \( x \) according to a probability distribution, and specifically the distribution \( y_k p_k \). We will first show that the distribution \( p_k \) is symmetric around its mean \( \mu_{p_k}(x) \). Then we will show that \( y_k(x) \) is also symmetric around the same mean. Then we will use the fact that the product of two such symmetric distributions with common mean, such as \( y_k p_k \), is a distribution with the same mean, which is a fact that we will prove in theorem A.1.

(d) Because the cosine similarity function \( u_k(x) \) is symmetric around the mean value \( \mu_{u_k}(x) = w_k: u_k(\mu_{u_k} - x) = u_k(\mu_{u_k} + x) \), it follows that \( p_k(\mu_{u_k} - x) = \exp(u_k(\mu_{u_k} - x)) = \exp(u_k(\mu_{u_k} + x)) = p_k(\mu_{u_k} + x) \).

Therefore, \( p_k(x) \) does have the sought property of symmetry, around \( \mu_{p_k}(x) \).

In point (a) of this list we have also shown that \( \mu_{u_k}(x) = \mu_{p_k}(x) \), thus \( p_k \) is symmetric around its own mean \( \mu_{p_k}(x) \).

(e) The reason why the softmax output \( y_k \) is symmetric around the same mean as \( p_k \) consists in the following arguments:

1. The numerator \( p_k \) of the \( y_k \) softmax in equation (53) is symmetric around \( \mu_{p_k}(x) \), as was shown in the preceding point.
2. The denominator of equation (53), i.e. \( \sum_{l=1}^{K} p_l(x) P(C_l) \) is also symmetric around \( \mu_{p_k}(x) \) in \( \Omega_k^p \), where \( \Omega_k^p \) is the 'support' of the \( p_k \) distribution, i.e. the region where the numerator \( p_k \) is not negligible.

This is because:

(i) We assume that the data is distributed on the unit hypersphere of the input space according to \( p(x) \) without a bias. Therefore the total contribution of components \( \sum_{l \neq k} p_l(x) P(C_l) \) to \( p(x) \) in that neighbourhood \( \Omega_k^p \) is approximately symmetric around \( \mu_{p_k}(x) \).

(ii) \( \sum_{l \neq k} p_l(x) P(C_l) \) is not only approximately symmetric, but also its remaining asymmetry has a negligible contribution to \( p(x) \) in \( \Omega_k^p \), because \( p(x) \) in \( \Omega_k^p \) is mostly determined by \( p_k(x) \).

This is true, because as we showed in point (C1), \( \Omega_k^p \cap \Omega_l^p \) is a narrow overlap \( \forall l \neq k \).

(iii) \( p_k \) is also symmetric, therefore, the total sum \( \sum_{l=1}^{K} p_l(x) P(C_l) \) is symmetric around \( \mu_{p_k}(x) \).

3. The inverse fraction \( \frac{1}{f} \) of a symmetric function \( f \) is also symmetric around the same mean, therefore the inverse of the denominator \( \frac{1}{p(x)} = \frac{1}{\sum_{l=1}^{K} p_l(x) P(C_l)} \) is also symmetric around the mean value \( \mu_{p_k}(x) \).

4. The normalized product of two distributions that are symmetric around the same mean is a probability distribution with the same mean. We prove this formally in theorem 4.1 and in its proof at the end of the present section 4.1.

Therefore \( y_k = p_k \frac{1}{p(x)} \) is indeed symmetric around \( \mu_{p_k}(x) = \mu_{u_k}(x) = w_k = \mu_{p_k}(x) \).

In summary,

\[ \mu_{p_k}(x) = \mu_{y_k}(x), \] (56)

and, due to the symmetry,

\[ p_k(\mu_{p_k} + x) = p_k(\mu_{p_k} - x), \] (57)

\[ y_k(\mu_{p_k} + x) = y_k(\mu_{p_k} - x). \] (58)

(f) Because the means of \( y_k(x) \) and of \( p_k(x) \) are both equal to \( \mu_{p_k}(x) \), and because both distributions are symmetric around that mean, the probability distribution \( y_k(x) p_k(x) P(C_k) / I_k \), where \( I_k \) is the normalization constant, also has a mean \( \mu_{y p_k}(x) \) equal to \( \mu_{p_k}(x) \):

\[ \int x y_k(x) p_k(x) P(C_k) / I_k \, dx = \mu_{p_k}(x). \] (59)

We prove this formally in theorem 4.1 and in its proof at the end of the present section 4.1.
Therefore, the first component of the sum in equation (55) is

\[
\int_x x_i y_k(x)p_k(x)P(C_k)\,dx = I_k \cdot \mu_{p_k}(x)
\]  

and, similarly, the second component is

\[
-w_{ik}w_k \int_x x_i y_k(x)p_k(x)P(C_k)\,dx = -I_k w_{ik} \cdot \mu_{p_k}(x).
\]

From the above conclusions about the two components of the sum in equation (55), it follows that

\[
E[\Delta w_{ik}^{(\text{SoftHebb})}] = I_k \cdot \mu_{p_k}(x_i) - I_k w_{ik} \cdot \mu_{p_k}(x)
\]

\[
= I_k \cdot (\mu_{p_k}(x_i) - w_{ik}\mu_{p_k}(x) \cdot w_{ik})
\]

\[
= I_k \cdot (\mu_{p_k}(x_i) - ||\mu_{p_k}(x)|| \cdot w_{ik})
\]

\[
= I_k \cdot \left( \mu_{p_k}(x_i) - ||\mu_{p_k}(x)|| \frac{\mu_{p_k}(x_i)}{||\mu_{p_k}(x)||} \right)
\]

\[
= 0.
\]

Therefore, it is indeed true that \( \left[ w_k = \text{opt} w_k^i = \frac{\mu_{p_k}(x)}{||\mu_{p_k}(x)||} \forall k \right] \implies E[\Delta w_{ik}^{(\text{SoftHebb})}] = 0 \forall i, k. \)

Thus, the optimal weights of the model \( \text{opt} w_k^i = \frac{\mu_{p_k}(x)}{||\mu_{p_k}(x)||} \forall k \) are equilibrium weights of the SoftHebb plasticity rule and network.

However, it is not yet clear that the weights that are normalized to a unit vector are those that the rule converges to, and that other norms of the vector are unstable. We will now give an intuition, and then prove that this is the case.

The multiplicative factor \( u_k \) is common between our rule and Oja’s rule (Oja 1982). The effect of this factor is known to normalize the weight vector of each neuron to a length of one (Oja 1982), as also shown in similar rules with this multiplicative factor (Krotov and Hopfield 2019). We prove that this is the effect of the factor also in the SoftHebb rule, separately in theorem 2.3 and its proof, provided at the end of the present section 4.1.

This proves theorem 2.2, and satisfies the optimality condition derived in theorem 2.3. \( \square \)

Proof of Theorem 2.3. Using a technique similar to Krotov and Hopfield (2019), we write the SoftHebb plasticity rule as a differential equation

\[
\tau \frac{dw_{ik}^{(\text{SoftHebb})}}{dt} = \Delta w_{ik}^{(\text{SoftHebb})}(t) = \eta \cdot y_k(t) \cdot (x_i(t) - u_k(t)w_{ik}(t))
\]

In this formulation, synapses undergo continuous changes, with an instantaneous rate \( \Delta w_{ik}^{(\text{SoftHebb})}(t)/\tau \). The time constant of the plasticity dynamics is \( \tau \).

The derivative of the norm of the weight vector is

\[
\frac{d||w_k||}{dt} = \frac{d(w_kw_k)}{dt} = 2w_k \frac{dw_k}{dt}
\]

Replacing \( \frac{dw}{dt} \) in this equation with the SoftHebb rule of equation (67), it is

\[
\frac{d||w_k^{(\text{SoftHebb})}||}{dt} = 2 \frac{\eta}{\tau} w_k \cdot y_k \cdot (x - u_kw_k) = 2 \frac{\eta}{\tau} w_k \cdot y_k \cdot (x - w_kxw_k) = 2 \frac{\eta}{\tau} w_ky_k \cdot (1 - ||w_k||).
\]

This differential equation shows that the derivative of the norm of the weight vector increases if \( ||w_k|| < 1 \) and decreases if \( ||w_k|| > 1 \), such that the weight vector tends to a sphere of radius 1, which proves the theorem. \( \square \)
Proof of Theorem 2.4. Similarly to the proof of theorem 2.2, we find the equilibrium parameter \( w_{0k} \) of the SoftHebb plasticity rule

\[
E[\Delta w_{0k}^{(\text{SoftHebb})}] = \eta \int_x (y_k e^{-w_{0k}} - 1) p(x)\,dx
\]

(70)

\[
= \eta \int_x (e^{-w_{0k}} \int_x y_k(x) p(x)\,dx - 1)
\]

(71)

\[
= \eta \left( e^{-w_{0k}} \int_x p(x|C_k)P(C_k)p(x)\,dx - 1 \right)
\]

(72)

\[
= \eta \left( e^{-w_{0k}} P(C_k) \int_x p(x|C_k)dx - 1 \right)
\]

(73)

\[
= \eta \left( e^{-w_{0k}} P(C_k) - 1 \right).
\]

(74)

In the above, we have replaced \( y_k \) by its definition, i.e. \( y_k(x) = p(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)} \).

Therefore, using this form of \( E[\Delta w_{0k}^{(\text{SoftHebb})}] \), and setting this expectation to zero as a condition for equilibrium, we find the equilibrium value of \( w_{0k} \):

\[
E[\Delta w_{0k}^{(\text{SoftHebb})}] = 0 \implies w_{0k} = \ln P(C_k),
\]

(76)

which proves theorem 2.4 and shows the SoftHebb plasticity rule of the neuronal bias finds the optimal parameter of the Bayesian generative model as defined by equation (6) of theorem 2.3.

Theorem 4.1. Given two probability density functions (PDFs) \( y(x) \) and \( p(x) \) that are both centred symmetrically around the same mean value \( \mu \), their product \( y(x)p(x) \), normalized appropriately, is a PDF with the same mean, i.e.

\[
\begin{align*}
\int_x xy(x)dx &= \mu \\
\int_x xp(x)dx &= \mu \\
p(\mu + x) &= p(\mu - x) \\
y(\mu + x) &= y(\mu - x)
\end{align*}
\]

(77)

\[
\implies \frac{1}{\pi} \int_x xy(x)p(x)\,dx = \mu.
\]

Proof of Theorem 4.1.

\[
\int_{-\infty}^{+\infty} xy(x)p(x)\,dx = \int_{-\infty}^{\mu} xy(x)p(x)\,dx + \int_{\mu}^{+\infty} xy(x)p(x)\,dx.
\]

(78)

We will derive a different form for each of these two integrals,

\[
I_1 := \int_{x=-\infty}^{\mu} xy(x)p(x)\,dx
\]

(79)

\[
= \int_{x=-\infty}^{\mu} (\mu - u)y(\mu - u)p(\mu - u)\,d(\mu - u).
\]

(80)

We defined \( x = \mu - u \),

\[
= \int_{u=+\infty}^{0} (\mu - u)y(\mu - u)p(\mu - u)\,d(\mu - u).
\]

(81)

We substituted the integration limits accordingly,

\[
= - \int_{+\infty}^{0} (\mu - u)y(\mu - u)p(\mu - u)\,du.
\]

(82)
Because \( d(\mu - u) = -du \),

\[
I_2 := \int_{\mu}^{+\infty} xy(x)p(x)dx
\]

\[
= \int_{0}^{+\infty} (\mu + u)y(\mu + u)p(\mu + u)d(\mu + u)
\]

\[
= \int_{0}^{+\infty} \mu y(\mu + u)p(\mu + u)du + \int_{0}^{+\infty} uy(\mu + u)p(\mu + u)du
\]

\[
= \mu \int_{\mu}^{+\infty} y(x)p(x)dx + \int_{0}^{+\infty} uy(\mu + u)p(\mu + u)du.
\]

Therefore,

\[
\int_{-\infty}^{+\infty} xy(x)p(x)dx = I_1 + I_2
\]

\[
= \mu \int_{\mu}^{+\infty} y(x)p(x)dx - \int_{0}^{+\infty} uy(\mu + u)p(\mu + u)du
\]

\[
+ \mu \int_{\mu}^{+\infty} y(x)p(x)dx + \int_{0}^{+\infty} uy(\mu + u)p(\mu + u)du = \mu \cdot I,
\]

where \( I = \int_{-\infty}^{+\infty} y(x)p(x)dx \).
4.2. Details to theoretical support of alternate activation functions (section 2.5)

Theorem 2.3, which concerns the synaptic plasticity rule in equation (9), was proven for the model of definition 2.2, which uses a mixture of natural exponential component distributions, i.e. with base e (equation (5)): 

\[ q_k := q(x|C_k; w_k) = e^{wh}. \]  

(96)

This implied an equivalence to a WTA neural network with natural exponential activation functions (section 2.2). However, it is simple to show that these results can be extended to other model probability distributions, and thus other neuronal activations.

Firstly, in the simplest of the alternatives, the base of the exponential function can be chosen differently that case, the posterior probabilities that are produced by the model’s Bayesian inference, i.e. the network outputs, \( Q(C_k|x; w) = y_k(x; w) \) are given by a softmax with a different base. If the base of the exponential is \( b \), then

\[ Q(C_k|x; w) = y_k = \frac{b^{u_k + w_k}}{\sum_{l=1}^{R} b^{u_l + w_l}}. \]  

(97)

It is obvious in the proof of theorem 2.3 in section 4.1 that the same proof also applies to the changed base, if we use the appropriate logarithm for describing KL divergence. Therefore, the optimal parameter vector does not change, and the SoftHebb plasticity rule also applies to the SoftHebb model with a different exponential base. This change of the base in the softmax bears similarities to the change of its exponent, in a technique that is called temperature scaling and has been proven useful in classification (Hinton et al 2015).

Secondly, the more conventional type of temperature scaling, i.e. that which scales exponent, is also possible in our model, while maintaining a Bayesian probabilistic interpretation of the outputs, a neural interpretation of the model, and the optimality of the plasticity rule this case, the model becomes

\[ Q(C_k|x; w) = y_k = \frac{e^{(u_k + w_k)/T}}{\sum_{l=1}^{R} e^{(u_l + w_l)/T}}. \]  

(98)

The proof of theorem 2.3 in section 4.1 also applies in this case, with a change in equation (22), but resulting in the same solution. Therefore, the SoftHebb synaptic plasticity rule is applicable in this case too. The solution for the neuronal biases, i.e. the parameters of the prior in the theorem (equation (6)), also remains the same, but with a factor of \( T: \ opt_{w_k} = T \ln P(C_k) \).

Finally, and most generally, the model can be generalized to use any non-negative and monotonically increasing function \( h(x) \) for the component distributions, i.e. for the activation function of the neurons, assuming \( h(x) \) is appropriately normalized to be interpretable as a PDF this case, the model becomes

\[ Q(C_k|x; w) = y_k = \frac{h(u_k) \cdot w_k}{\sum_{l=1}^{R} h(u_l) \cdot w_l}. \]  

(99)

Note that there is a change in the parametrization of the priors into a multiplicative bias \( w_k \), compared to the additive bias in the previous versions above. This change is necessary in this general case, because not all functions have the property \( e^{a+b} = e^a \cdot e^b \) that we used in the exponential case. We can show that the optimal weight parameters remain the same as in the previous case of an exponential activation function, also for this more general case of activation \( h \). It can be seen in the proof of theorem 2.3, that for a more general function \( h(x) \) than the exponential, equation (22) would instead become:

\[ \text{opt}_{w_k} = \arg \min_{w_k} D_{KL}(p_k||q_k) = \arg \max_{w_k} \int p_k \ln h(u_k) dx \]

\[ = \arg \max_{w_k} \int p_k \ln h(\cos(w_k,x)) dx \]

\[ = \arg \max_{w_k} \mu_{p_k} \ln h(\cos(w_k,x)) \]

\[ = \arg \max_{w_k} \mu_{p_k} g(\cos(w_k,x)), \]  

(100)

where \( g(x) = \ln h(x) \). We have assumed that \( h \) is an increasing function, therefore \( g \) is also increasing. The cosine similarity is symmetrically decreasing as a function of \( x \) around \( w_k \). Therefore, the function \( g'(x) = g(\cos(w_k,x)) \) also decreases symmetrically around \( w_k \). Thus, the mean of that function \( g \) under the probability distribution \( p_k \) is maximum when \( \mu_{p_k} = w_k \). As a result, equation (100) implies that in this more
Table 1. Training hyperparameters.

|                        | Hard WTA | Soft WTA | Backpropagation |
|------------------------|----------|----------|-----------------|
| **100 epochs**         |          |          |                 |
| Optimizer              | —        | —        | Adam            |
| Softmax base           | —        | 1000     | —               |
| Initial learning rate  | 0.05     | 0.03     | 0.001           |
| Learning rate decay    | Linear   | Linear   | —               |
| Minibatch size         | 128      | 128      | 64              |
| **1 epoch**            |          |          |                 |
| Optimizer              | —        | —        | SGD             |
| Softmax base           | —        | 200      | —               |
| Initial learning rate  | 0.55     | 0.55     | 0.2             |
| Learning rate decay    | Exponential | Exponential | —       |
| Minimum learning rate  | 0.0055   | 0.0055   | —               |
| Minibatch size         | 1        | 1        | 4               |

Figure 9. Adversarial-attack hyperparameter tuning. Accuracy as a function of number of PGD iterations, and of step size for different attack strength size $\epsilon$.

4.3. Hyperparameters

We tuned the hyperparameters of the learning algorithms for each experiment. The tuned values that we found for MNIST are shown in the table below.

For Fashion-MNIST, the same values are best, except that an initial learning rate value of 0.065 performed better for the WTA networks.

4.4. Details to adversarial attack experiments

We used ‘Foolbox’, a Python library for adversarial attacks (Rauber et al 2017). PGD has a few parameters that influence the effectiveness of the attack. Namely, $\epsilon$ which is a parameter determining the size $\epsilon$ of the perturbation, the number of iterations for the attack’s gradient ascent, the step size per iteration, and a number of possible random restarts per attacked sample. Here we chose five random restarts. Then we found that 200 iterations are sufficient for both MNIST and F-MNIST (figures 9(a) and (b)). Then, using 200 iterations, and different $\epsilon$ values, we searched for a sufficiently good step size. We found that relative to $\epsilon$ a step size value of $0.01\epsilon \approx 0.33\epsilon$ (which is also the default value of the toolbox that we used) is a good value.

An example curve for $\epsilon = 32/255$ is shown in figure 9(c) for MNIST and F-MNIST.

5. Discussion

We have described SoftHebb, a highly biologically plausible neural algorithm that is founded on a Bayesian ML-theoretic framework. The model consists of elements fully compatible with conventional ANNs. It was previously not known which plasticity rule should be used to learn a Bayesian generative model of the input general model too, the optimal weight vector is $\text{opt} \mathbf{w}_k = c \cdot \mathbf{H}_{P_k}(\mathbf{x})$, $c \in \mathbb{R}$, and, consequently, it is also optimized by the same SoftHebb plasticity rule.

The implication of this is that the SoftHebb WTA neural network can use activation functions such as ReLUs, or other non-negative and increasing activations, such as rectified polynomials (Krotov and Hopfield 2019) etc and maintain its generative properties, its Bayesian computation, and the theoretical optimality of the plasticity rule. A more complex derivation of the optimal weight vector for alternative activation functions, which was specific to ReLU only, and did not also derive the associated long-term plasticity rule for our problem category (definition 2.1), was provided by Moraitis et al (2020).
distribution in ANN WTA networks. Here we derived this rigorously. Moreover, we showed that Hard-WTA networks, and neurons with other activation functions can be described within the same framework as variations of the probabilistic model. This theory could provide a new foundation for normative Hebbian ANN designs with practical significance.

SoftHebb’s properties are highly sought-after by efficient neuromorphic learning chips. It is unsupervised, local, and requires no error or other feedback currents from upper layers, thus solving hardware-inefficiencies and bio-implausibilities of BP such as weight-transport and update-locking. However, a limitation of our study is that we have not experimented with deeper networks that could enable true applicability in complex datasets. This will require further experiments that are beyond the single-layer theoretical scope of the present work. A convolutional implementation could become the foundation of such experiments and could provide insights into the role of WTA microcircuits in larger cortical networks with localized receptive fields (Pogodin et al 2021), similar to area V1 of cortex (Hubel and Wiesel 1962). This would be a radically different approach from BF and its approximations, circumventing their key limitations by not relying on any feedback signals. The possibility that this may reach competitive accuracies in multilayer datasets becomes more realistic as a consequence of the present work. The ability of SoftHebb to minimize cross-entropy without supervision, and its fast and robust learning, are properties that may conceivably support a multilayer learning algorithm without any feedback signals (Journé et al 2022). In addition to its future potential, surprisingly, SoftHebb already has practical advantages beyond its efficiency. Specifically, it surpasses BP in accuracy, when training time and network size are limited. Moreover, in a demonstration that goes beyond the common greedy-training approach to Hebbian networks, SoftHebb achieves update-unlocked operation in practice, by updating the first layer before the input’s processing by the next layer. It is intriguing that, through its biological plausibility, emerge properties commonly associated with biological intelligence, such as speed of learning, and substantially increased robustness to noise and adversarial attacks. Importantly, robustness emerges without specialized defences. Furthermore, SoftHebb tends to not merely be robust to attacks, but actually deflect them as specialized SOTA defences aim to do (Qin et al 2020). We also demonstrated the ability of SoftHebb to generate synthetic objects as interpolations of true object classes.

All in all, the algorithm has several properties that are individually interesting, novel, and worth future exploration. Combined, however, SoftHebb’s properties already enable certain applications that are small-scale but infeasible with BP-based learning. For example, fast, on-line, unsupervised learning of simple tasks by edge-sensing neuromorphic devices, operating in noisy conditions, with a small battery and only local processing, requires those algorithmic properties that we have demonstrated here.

Data availability statement

No new data were created or analysed in this study.

ORCID iD

Timoleon Moraitis © https://orcid.org/0000-0002-6521-0717

References

Amato G, Carrara F, Falchi F, Gennaro C and Lagani G 2019 Hebbian learning meets deep convolutional neural networks Int. Conf. on Image Analysis and Processing (Springer) pp 324–34
Bardes A, Ponce J and LeCun Y 2021 VICReg: variance-invariance-covariance regularization for self-supervised learning (arXiv:2105.04906)
Bell A J and Sejnowski T J 1995 An information-maximization approach to blind separation and blind deconvolution Neural Comput. 7 1129–59
Bengio Y, Lee D-H, Bornschein J, Mesnard T and Lin Z 2015 Towards biologically plausible deep learning (arXiv:1502.04156)
Berthelot D, Raffel C, Roy A and Goodfellow I 2018 Understanding and improving interpolation in autoencoders via an adversarial regularizer (arXiv:1807.07543)
Binas J, Rutishauser U, Indiveri G and Pfister M 2014 Learning and stabilization of winner-take-all dynamics through interacting excitatory and inhibitory plasticity Front. Comput. Neurosci. 8 68
Binzegger T, Douglas R J and Martin K A 2004 A quantitative map of the circuit of cat primary visual cortex J. Neurosci. 24 8441–53
Bittar A and Garner P N 2022 A surrogate gradient spiking baseline for speech command recognition Front. Neurosci. 16
Bojanowski P, Joulin A, Lopez-Paz D and Szlam A 2017 Optimizing the latent space of generative networks (arXiv:1707.05776)
Cannon J, McCarthy M M, Lee S, Lee J, Bölcskei H, Whittington M A and Kopell N 2014 Neurosystems: brain rhythms and cognitive processing Eur. J. Neurosci. 39 705–19
Chen T, Kornblith S, Norouzi M and Hinton G 2020 A simple framework for contrastive learning of visual representations Int. Conf. on Machine Learning (PMLR) pp 597–607
Cowen-Rivers A I, Lyu W, Tutunov R, Wang Z, Grosnut A, Griffiths R R, Maraval A M, Jianye H, Wang J and Peters J et al 2022 HEBO: pushing the limits of sample-efficient hyper-parameter optimisation J. Artif. Intell. Res. 74 1269–349
Cresswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B and Bharath A A 2018 Generative adversarial networks: an overview

IEEE Signal Process. Mag. 35 53–65

Crick F 1989 The recent excitement about neural networks Nature 337 129–32

Czarnecki W M, Świrszcz G, Jaderberg M, Osindero S, Vinyals O and Kavukcuoglu K 2017 Understanding synthetic gradients and decoupled neural interfaces Int. Conf. on Machine Learning (PMLR) pp 904–12

Devlin J, Chang M-W, Lee K and Toutanova K B 2018 Pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805)

Diehl P U and Cook M 2015 Unsupervised learning of digit recognition using spike-timing-dependent plasticity Front. Comput. Neurosci. 9 99

Diehl P U and Cook M 2016 Learning and inferring relations in cortical networks (arXiv:1608.08267)

Douglas R J and Martin K A 2004 Neuronal circuits of the neocortex Annu. Rev. Neurosci. 27 419–51

Ernoult M, Grollier J, Quevillon D, Bengio Y and Scellier B 2020 Equilibrium propagation with continual weight updates (arXiv:2005.04168)

Földiák P 1990 Forming sparse representations by local anti-Hebbian learning Biol. Cybern. 64 165–70

Földiák P and Dfldr P 1989 Adaptive network for optimal linear feature extraction 1989 International Joint Conference on Neural Networks 1 401–5

Frenkel G, Lefebvre M and Bol D 2021 Learning without feedback: fixed random learning signals allow for feedforward training of deep neural networks Front. Neurosci. 15 629892

García Rodríguez H, Guo Q and Moraitis T 2022 Short-term plasticity neurons learning to learn and forget Int. Conf. on Machine Learning (PMLR) pp 18704–22

Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial networks (arXiv:1406.2661)

Grinberg L, Hopfield J and Krotov D 2019 Local unsupervised learning for image analysis (arXiv:1908.08993)

Grossberg S 1987 Competitive learning: from interactive activation to adaptive resonance Cogn. Sci. 11 23–63

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

Hahnloser R, Douglas R J, Mahowald M and Sepp K 1999 Feedback interactions between neuronal pointers and maps for attentional processing Nat. Neurosci. 2 746–52

Hinton G, Vinyals O and Dean J 2015 Distilling the knowledge in a neural network (arXiv:1503.02531)

Hu T, Pechenov C and Chklovskii D B 2014 A Hebbian/anti-Hebbian network for online sparse dictionary learning derived from symmetric matrix factorization 2014 48th Asilomar Conf. on Signals, Systems and Computers (IEEE) pp 613–19

Hubel D H and Wiesel T N 1962 Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex J. Physiol. 160 106–154

Illing B, Gerstner W and Brea J 2019 Biologically plausible deep learning—but how far can we go with shallow networks? Neural Netw. 118 90–101

Isomura T and Toyozumi T 2018 Error-gated Hebbian rule: a local learning rule for principal and independent component analysis Sci. Rep. 8 1–11

Jeffares A, Guo Q, Stenotorp P and Moraitis T 2022 Spike-inspired rank coding for fast and accurate recurrent neural networks Int. Conf. on Learning Representations (available at: https://openreview.net/forum?id=iMH1e5k7n3L)

Journe A, Rodriguez, Guo Q and Moraitis T 2022 Hebbian deep learning without feedback (arXiv:2209.11883)

Kingma D P and Ba J 2015 Adam: a method for stochastic optimization 3rd Int. Conf. on Learning Representations, ICLR 2015 (San Diego, CA, USA, 7–9 May 2015) (Conf. Track Proc.) ed Y Bengio and Y LeCun (arXiv:1412.6980)

Krizhevsky A and Hinton G 2009

Krotov D and Hopfield J J 2019 Unsupervised learning by competing hidden units Proc. Natl Acad. Sci. 116 7723–31

Lagani G, Falchi F, Gennaro C and Amato G 2021 Hebbian semi-supervised learning in a sample efficiency setting Neural Netw. 143 719–31

Lee T-W, Girolami M and Sejnowski T J 2019 Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources Neural Comput. 11 417–47

Lillicrap T P, Cownden D, Tweed D B and Akerman C J 2016 Random synaptic feedback weights support error backpropagation for deep learning Nat. Commun. 7 1–10

Linsker R 1992 Local synaptic learning rules suffice to maximize mutual information in a linear network Neural Comput. 4 691–702

Maass W 2000 On the computational power of winner-take-all Neural Comput. 12 2359–35

Maday A, Makelov A, Schmidt L, Tsipras D and Vladu A 2017 Towards deep learning models resistant to adversarial attacks (arXiv:1706.06083)

Millidge B, Tschantsz A and Buckley C I L 2020 Predictive coding approximates backprop along arbitrary computation graphs (arXiv:2006.04182)

Moraitis T, Sebastian A and Eleftheriou E 2020 Optimality of short-term synaptic plasticity in modelling certain dynamic environments (arXiv:2009.08608)

Nessler B, Pfeiffer M, Buesing L and Maass W 2013 Bayesian computation emerges in generic cortical microcircuits by spike-timing-dependent plasticity PLoS Comput. Biol. 9 e1003037

Nessler B, Pfeiffer M and Maass W 2009 STDP enables spiking neurons to detect hidden causes of their inputs Advances in Neural Information Processing Systems vol 22 pp 1357–65

Nokland A 2016 Direct feedback alignment provides learning in deep neural networks Advances in Neural Information Processing Systems vol 29

Oja E 1982 Simplified neuron model as a principal component analyzer J. Math. Biol. 15 267–73

Olshausen B A and Field D J 1996 Emergence of simple-cell receptive field properties by learning a sparse code for natural images Nature 381 607–9

Olshausen B A and Field D J 1997 Sparse coding with an overcomplete basis set: a strategy employed by V1? Vis. Res. 37 3311–25

Payeur A, Guerguiev J, Zenke F, Richards B A and Naud R 2021 Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits Nat. Neurosci. 24 1010–19

Pechenov C and Chklovskii D B 2014 A Hebbian/anti-Hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features 2014 48th Asilomar Conf. on Signals, Systems and Computers (IEEE) pp 769–75
Pehlevan C and Chklovskii D 2015 A normative theory of adaptive dimensionality reduction in neural networks Advances in Neural Information Processing Systems vol 28, ed C Cortes, N Lawrence, D Lee, M Sugiyama and R Garnett (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2015/file/861dc9bd7f4e7dd3cccd534d0ae2a2e9-Paper.pdf)

Pehlevan C, Genkin A and Chklovskii D B 2017 A clustering neural network model of insect olfaction 2017 51st Asilomar Conf. on Signals, Systems and Computers (IEEE) pp 593–600

Pfeiffer M and Pfeil T 2018 Deep learning with spiking neurons: opportunities and challenges Front. Neurosci. 12

Pogodin R and Latham P E 2020 Kernelized information bottleneck leads to biologically plausible 3-factor Hebbian learning in deep networks (arXiv:2006.07123)

Pogodin R, Mehta Y, Lillicrap T P and Latham P E 2021 Towards biologically plausible convolutional networks (arXiv:2106.13031)

Poirazi P and Papoutsis A 2020 Illuminating dendritic function with computational models Nat. Rev. Neurosci. 21 303–21

Qin Y, Frost N, Raffel C, Cottrell G and Hinton G 2020 Deflecting adversarial attacks (arXiv:2002.07405)

Radford A, Metz L and Chintala S 2015 Unsupervised representation learning with deep convolutional generative adversarial networks (arXiv:1511.06434)

Rabler J, Brendel W and Bethge M 2017 Foolbox: a python toolbox to benchmark the robustness of machine learning models (arXiv:1707.04151)

Rutishauser U, Douglas R J and Slotine J-J 2011 Collective stability of networks of winner-take-all circuits Neural Comput. 23 735–73

Sanger T D 1989 Optimal unsupervised learning in a single-layer linear feedforward neural network Neural Netw. 2 459–73

Sarwat S G, Moraitis T, Wright C D and Bhaskaran H 2022 Chalcogenide optomemristors for multi-factor neuromorphic computation Nat. Commun. 13 1–9

Scellier B and Bengio Y 2017 Equilibrium propagation: bridging the gap between energy-based models and backpropagation Front. Comput. Neurosci. 11 24

Scherr F, Guo Q and Moraitis T 2022 Self-supervised learning through efference copies (arXiv:2210.09224)

Sejnowski T J 2020 The unreasonable effectiveness of deep learning in artificial intelligence Proc. Natl Acad. Sci. 117 0033–8

Von der Malsburg C 1973 Self-organization of orientation sensitive cells in the striate cortex Kybernetik 14 85–100

Xiao H, Rasul K and Vollgraf R 2017 Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms (arXiv:1708.07747)

Zador A et al 2022 Toward next-generation artificial intelligence: catalyzing the NeuroAI revolution (arXiv:2210.08340)