Learning representations in Bayesian Confidence Propagation neural networks

Naresh Balaji Ravichandran
Computational Brain Science Lab
KTH Royal Institute of Technology
Stockholm, Sweden
nbrav@kth.se

Anders Lansner
Computational Brain Science Lab
Stockholm University and KTH Royal Institute of Technology
Stockholm, Sweden
ala@kth.se

Pawel Herman
Computational Brain Science Lab
KTH Royal Institute of Technology
Stockholm, Sweden
paherman@kth.se

***Abstract*—Unsupervised learning of hierarchical representations has been one of the most vibrant research directions in deep learning during recent years. In this work we study biologically inspired unsupervised strategies in neural networks based on local Hebbian learning. We propose new mechanisms to extend the Bayesian Confidence Propagating Neural Network (BCPNN) architecture, and demonstrate their capability for unsupervised learning of salient hidden representations when tested on the MNIST dataset.**

Keywords—neural networks, brain-like computing, bio-inspired, unsupervised learning, structural plasticity.

I. Introduction

Artificial neural networks (ANN) have made remarkable progress in supervised pattern recognition in recent years. ANNs achieve this mainly under the umbrella of deep learning by discovering hierarchies of salient data representations [1]. At this stage, it is valuable to study how they compare with the biological neural networks, and explore new opportunities at this intersection [2, 3].

We see at least three fundamental differences between current deep learning approaches and the brain:

Firstly, most deep learning methods rely extensively on labelled samples for extracting and tuning data representations the hierarchy of representations, although biological systems mostly learn in an unsupervised fashion. Recent work in deep learning research has increasingly paid attention to developing unsupervised learning methods [4, 5, 6], and the work we present here will also be in this direction.

Secondly, deep learning methods predominantly make use of error back-propagation (backprop) for learning the weights in the network. Although extremely efficient, backprop has several issues that make it an unlikely candidate model for synaptic plasticity in the brain. The most apparent issue is that the synaptic connection strength between two biological neurons is expected to comply with Hebb's postulate, i.e. to depend only on the available local information provided by the activities of the pre- and postsynaptic neurons. This is violated in backprop, since synaptic weight updates need gradient signals to be communicated from distant output layers. Please refer to [7, 8] for a detailed review of possible biologically plausible implementations of and alternatives to backprop.

Thirdly, an important difference between current deep ANNs and the brain concerns the abundance of recurrent connections in the latter. A typical cortical area receives on the order of 10% of synapses from lower order structures, e.g. thalamus, and the rest from other cortical neurons [9]. In contrast, deep learning networks rely predominantly on feed-forward connectivity. The surplus 90% connections are likely involved in associative memory, constraint-satisfaction, top-down modulation and selective attention [9]. However, we will not consider those important aspects of cortical computation in this work.

These concerns motivate exploring alternative more biologically plausible learning strategies that enable unsupervised learning of representations using local Hebbian rules. The approach we follow here involves framing the update and learning steps of the neural network as probabilistic computations. Probabilistic approaches are widely used in both deep learning models [6] and computational models of brain function [10]. One disadvantage of probabilistic models is that the known methods do not scale well in practice. Also, inference and learning with distributed representations is often intractable and forces approximate approaches [6].

In this work, we examine a modular Bayesian Confidence Propagation Neural Network (BCPNN) architecture, previously used in abstract models of associative memory [11, 12], action selection [13], as well as in applications to brain imaging [14, 15] and data mining [16]. Spiking versions of BCPNN have also been used in biologically detailed models of different forms of cortical associative memory [17–20]. The modules, referred to as hypercolumns (HCs), comprise a number of functional minicolumns (MCs) that compete in a soft-winner-take-all manner. The abstract view of an HC is that it represents some attribute, e.g. edge orientation, in a discrete coded manner. An MC is represented as a unit that conceptually represents one discrete value (a realisation of the given attribute) and, as a biological parallel, it accounts for a local subnetwork of around a hundred recurrently connected neurons with similar receptive field properties [21]. Such an architecture was initially generalized from the primary visual cortex, but today has more support from later experimental work and has been featured in spiking computational models of cortex [22, 23].

Importantly, in this work we introduce additional mechanisms of bias regulation and structural plasticity to the BCPNN framework that conducts unsupervised learning of hidden representations. The bias regulation mechanism ensures that the activities of all units in the hidden layer are maintained near their target activity by adapting their bias parameter. Structural plasticity facilitates learning a set of sparse connections from the input to the hidden layer by greedily maximizing a local information theoretic score. The separability of the extracted representations is evaluated with a BCPNN classification layer equipped with dual pathways inspired by the model of reinforcement learning in the basal ganglia of the brain [13].

II. Related work

A popular unsupervised learning approach is to train a hidden layer to reproduce the input data as, for example, in autoencoders (AE) and restricted Boltzmann machines (RBM). The AE and RBM networks trained with a single hidden layer are relevant here since learning weights of the input-to-hidden-layer connections relies on local gradients, and the representations can be stacked on top of each other to extract hierarchical features. However, stacked AEs and deep belief nets (stacked RBMs) have typically been used for pre-training procedures that are followed by end-to-end supervised fine-tuning (using backprop) [5].

A recently proposed model by Krotov and Hopfield [24] addresses the problem of learning with local gradients by learning hidden representations solely using an unsupervised method. In the network the input-to-hidden connections are trained and additional (non-plastic) lateral inhibition provides competition within the hidden layer. For evaluating the representation, the weights are frozen, and a linear classifier trained with labels is used for the final classification. Our approach shares some common features with this model, e.g. learning hidden representations by unsupervised methods, and evaluating the representations by a separate classifier (refer [3] for an extensive review).

All of the aforementioned models employ either competition within the hidden layer [24] or feedback connections from the hidden to input layer (RBM and AE). The BCPNN uses only the feedforward connections along with an implicit competition via a local softmax operation, corresponding to local lateral inhibition in brain networks.

It has also been observed that sparse connectivity in the feed-forward connections performs better than full connectivity in some classification tasks [3]. The unsupervised learning methods we have discussed so far, however, employ full connectivity [3, 24]. Networks employing supervised learning like convolutional neural networks (CNNs) force a fixed spatial filter to obtain this sparse connectivity [25]. Here we take an alternate approach where along with learning the weights of the feed-forward connections, which is regarded as biological synaptic plasticity, we also simultaneously learn the sparse connectivity between the input and hidden layer, similar to constructive/pruning algorithms [26], and in analogy with the structural plasticity in the brain [27].

III. Bayesian Confidence Propagation neural network

Here, we describe the network architecture and update rules for the Bayesian Confidence Propagation Neural Network (BCPNN). The simplest BCPNN architecture for classification contains two layers, one for data and the other for labels.

A layer consists of a set of HCs, each of which represents a discrete random variable X_i (upper case). Each HC, in turn, is composed of a set of MCs representing a particular instance of the random variable x_i (lower case). The probability of the variable X_i is then a multinomial distribution, defined as $p(X_i = x_i)$, such that $\sum_{x_i} p(X_i = x_i) = 1$. In the neural network, the activity of the MC is interpreted as $p(X_i = x_i)$, and the sum of activities of all the MCs inside a HC sums to one.

Since the network is a probabilistic graphical model, we can compute the posterior of a target HC in the label layer conditioned on all the N source HCs in the input layer. We will use x's and y's for referring the HCs in the input and output layer respectively. Computing the exact posterior $p(Y_j|X_{1:N})$ over the target HC is intractable, since it scales exponentially with the number of units. The assumptions $p(X_1,..,X_N|Y_j) = \prod_{i=1}^{N} p(X_i|Y_j)$ and $p(X_1,..,X_N) = \prod_{i=1}^{N} p(X_i)$ allows us to write the posterior as:

$$\begin{aligned} p(Y_j|X_1,..,X_N) &= p(Y_j)\frac{p(X_1,..,X_N|Y_j)}{p(X_1,..,X_N)} \\ &= p(Y_j)\prod_{i=1}^{N}\frac{p(X_i,Y_j)}{p(X_i)\,p(Y_j)} \quad (1) \end{aligned}$$

When the network is driven by input data $\{X_1,..,X_N\} = \{x_1^D,..,x_N^D\}$, we can write the posterior probabilities of a target MC in terms of the source MCs as:

$$\begin{aligned} p(y_j|x_1^D,..,x_N^D) &= p(y_j)\prod_{i=1}^{N}\frac{p(x_i^D,y_j)}{p(x_i^D)p(y_j)} \\ &= p(y_j)\prod_{i=1}^{N}\prod_{x_i}\left(\frac{p(x_i,y_j)}{p(x_i)p(y_j)}\right)^{I(x_i=x_i^D)} \quad (2) \end{aligned}$$

where $I(\cdot)$ is the indicator function that equals 1 if its argument is true, and zero otherwise. We have written the posterior of the target MC as a function of all the source MCs (all x_i's). The log posterior can be written as:

$$\begin{aligned} \log p(y_j|x_1^D,..,x_N^D) &= \log p(y_j) \\ &+ \sum_{i=1}^{N}\sum_{x_i} I(x_i = x_i^D)\,\log\frac{p(x_i,y_j)}{p(x_i)p(y_j)} \quad (3) \end{aligned}$$

Since the posterior is linear in the indicator function of data sample, $I(x_i = x_i^D)$ can be approximated by its expected value, that is, $p(x_i^D)$. Except for $p(x_i^D)$, all the terms in the posterior are functions of the marginals $p(x_i)$,

$p(y_j)$, and $p(x_i, y_j)$. We define the terms bias $\beta(y_j) = log\, p(y_j)$ and weight $w(x_i, y_j) = log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$ in analogy with artificial neural networks.

The inference step to calculate the posterior probabilities of the target MCs conditioned on the input sample is given by the activity update equations:

$$h(y_j) = \beta(y_j) + \sum_{i=1}^{N} \sum_{x_i} p(x_i^D)\, w(x_i, y_j),$$
$$\pi(y_j) = \frac{exp(\gamma h(y_j))}{\sum_k exp(\gamma h(y_k))} \qquad (4)$$

where $h(y_j)$ is the total input received by each target MC from which the posterior activity $\pi(y_j)$ is recovered by softmax normalization (with gain γ) within the HC.

The learning step involves incrementally updating all the marginals as input samples are presented. The marginals $p(x_i), p(y_j)$, and $p(x_i, y_j)$ are computed from the probabilities $\pi(x_i)$ and $\pi(y_j)$ using exponentially weighted running averages, and the bias and weight parameters are, in turn, computed from the marginals as follows:

$$\tau_p \frac{dp(x_i)}{dt} = \pi(x_i) - p(x_i),$$
$$\tau_p \frac{dp(x_i, y_j)}{dt} = \pi(x_i)\pi(y_j) - p(x_i, y_j),$$
$$\tau_p \frac{dp(y_j)}{dt} = \pi(y_j) - p(y_j),$$
$$\beta(y_j) = k_\beta\, log\, p(y_j),$$
$$w(x_i, y_j) = k_w\, log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}. \qquad (5)$$

The terms k_β, k_w, and τ_p are the bias gain, weight gain, and learning time constant, respectively. Equations 4 and 5 define the set of update and learning equations of the BCPNN architecture. The scope of the work is limited to this abstract model of BCPNN where MCs are the fundamental computational unit.

IV. UNSUPERVISED REPRESENTATION LEARNING

The network for unsupervised learning is similar to the two-layer network, except we now have more than one HC, each of which can contain an arbitrary number of MCs (see Fig. 2). On top of the regular BCPNN equations, we introduce additional mechanisms that enable learning representations. We will use z to refer to this layer, and differentiate it from the layer with supervised learning (y in the previous section).

A. Bias regulation

The BCPNN update rule implements Bayesian inference if the parameters are learnt with the source and target layer probabilities available as observations. When the target layer is hidden, we are learning the representations, and we cannot expect the update rule to follow Bayesian inference. In fact, we can see that performing learning and inference simultaneously is counter-productive in this case. If a hidden representation with random initialization assigned some MCs with slightly higher marginal probability $p(z_j)$ than others, learning would then amplify this difference. In consequence, the learnt parameters would make the network associate more input samples with the MCs with high $p(z_j)$, hence causing the marginals to increase further. One way to circumvent this undesirable effect is to promote MCs with low $p(z_j)$ to be more active in the future in the spirit of an activity dependent homeostasis process in biological terms [28].

To this end, we use a bias regulation mechanism, where the bias gain k_β for each MC (equal to 1 if only Bayesian inference is performed) depends on $p(z_j)$. One motivation for choosing the bias gain is to influence the marginal $p(z_j)$ alone without affecting the weight parameters that are responsible for learning the input to hidden mapping. The value of $p(z_j)$ is compared with respect to the maximum entropy probability, $p_{MaxEnt} = 1/N_{MC}$, where N_{MC} is the number of MCs per HC. It is worth noting that the maximum entropy is the ideal representation without the input layer since all the MCs have equal marginal probability, and hence the uniform distribution acts as the reference for bias regulation. The dynamic update of k_β with the time constant τ_k follows Eq. 6:

$$\tau_k \frac{dk_\beta}{dt} = 1 + (k_{half} - 1)\frac{(p_{MaxEnt}/4)^2}{(p(y_j) - p_{MaxEnt}/4)^2} - k_\beta \qquad (6)$$

The mechanism maintains the value of gain k_β at around 1 when $p(z_j) \gg p_{MaxEnt}$, and drops sharply to negative values when $p(z_j)$ is below p_{MaxEnt} (see Fig. 1). The rate of this drop is controlled using the metaparameter k_{half}, defined as the value of gain $k_\beta = k_{half}$ at $p(z_j) = 1/2\, p_{MaxEnt}$. For learning representations, this dynamically updating version of bias gain k_β substitutes the constant in Equation 5.

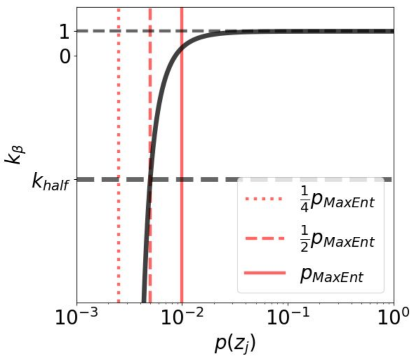

Fig. 1: Bias regulation mechanism. For generating the figure, $k_{half} = -5$ and $p_{MaxEnt} = 0.01$ was used.

B. Structural plasticity

Structural plasticity aims to iteratively improve the receptive fields for the hidden HCs from the input layer by greedily maximising the information content transferred from the input MCs. We define a boolean variable M_{x_i, Z_j} denoting the connection from an input MC x_i to an hidden HC Z_j as either active, $M_{x_i, Z_j} = 1$, or silent, $M_{x_i, Z_j} = 0$. Each M is initialized randomly with probability

$p_M = p(M=1)$. Once initialized, the number of active incoming connections to each hidden HC is fixed whereas the outgoing connections from a source MC can be changed (see Fig. 2). The mutual information corresponding to a connection is determined based on the probability distributions over $\{x_i, \neg x_i\}$ and Z_j using the BCPNN weights:

$$I_{x_i,Z_j} = \sum_{x \in \{x_i, \neg x_i\}} \sum_{z_j} p(x, z_j) \log \frac{p(x,z_j)}{p(x)p(z_j)} \quad (7)$$

The information I_{x_i,Z_j} is then normalized by the number of active outgoing connections for each input MC, i.e. its fan-out (with an additional 1 for numerical stability):

$$\widehat{I}_{x_i,Z_j} = I_{x_i,Z_j} \,/\, (1 + \sum_k M_{x_i,Z_k}) \quad (8)$$

Since the total number of active incoming connections is fixed, each hidden HC greedily maximizes the $\widehat{I}$ it receives by removing the active connection with the lowest $\widehat{I}$ (set M from 1 to 0) and adding the inactive connection with the highest $\widehat{I}$ (set M from 0 to 1). We call this operation a *flip*, and use a parameter N_{flips} to set the number of flips made per training epoch.

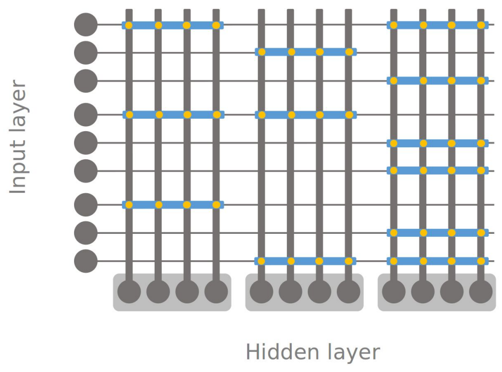

Fig. 2: The schematic of the network used for unsupervised learning. In this network, the input layer contains 9 binary MCs (grey circles on the left), and the hidden layer contains 3 HCs (grey boxes), each of which contains 4 MCs (grey circles inside the boxes). The existence of a connection between an input MC and hidden HC is shown as a blue strip, i.e., $M = 1$. The input-hidden weights are shown as yellow dots and are present only when a connection already exists.

V. BCPNN Classification

After learning the input-hidden connection, we freeze the weights, biases, and receptive fields of this connection, and treat the hidden layer representations as input to train a BCPNN classifier with an output (label) layer. This way we evaluate the separability of the learned representations. To this end, we add another BCPNN projection from hidden to output layer with a negative gain $k_w = -1$ (in contrast to the existing projection with $k_w = 1$). This is analogous to a network architecture used to model reinforcement learning in the basal ganglia [13]. Using systems neuroscience nomenclature, we call the projections *Go* ($k_w = 1$) and *No-Go* ($k_w = -1$), as they are intended to increase the probability of correct labels and reduce the probability of wrong labels, respectively. The classification layer training procedure is as follows: we first drive the pre-trained network from input samples and check the predicted label. If the classification is wrong, we either train the Go projection (by setting the output activations to the true label), or train the No-Go projection (by setting the output activations to the predicted label), or both. We run this procedure for N_{sup} epochs.

VI. Results

We evaluate the model using the MNIST hand-written digits database [20]. MNIST contains 60000 training and 10000 test images of 28x28 handwritten digits. The images were flattened to 784 dimensions and the grey-scale intensities were normalized to the range [0,1] and interpreted as probabilities. For each of the following subsections, we used N_{train}=50000 random training samples for training, report on the other N_{val}=10000 in the validation set, and at the end of this section, report the test accuracy of N_{test}=10000 samples for the best set of model parameters.

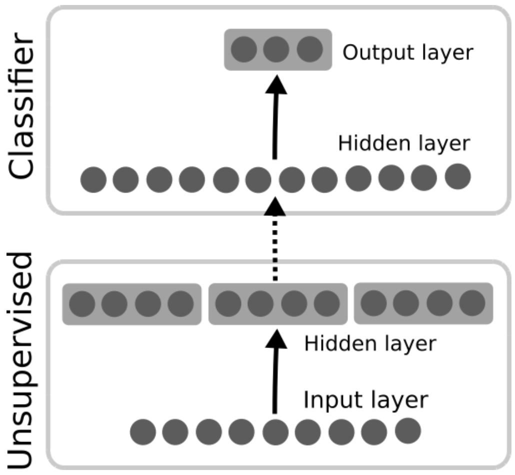

Fig. 3: Schematic of the unsupervised learning and classifier network employing BCPNN architecture. Here, for illustrative purposes, the input layer has 9 MCs, the hidden layer has 3 HCs and 4 MCs per HC (collectively 12 units), and the output layer has 3 MCs (label units). The dotted lines imply we use the representations of the hidden layer as input for the classifier.

The network had 784 input MCs, the hidden layer had 30 HCs and 100 MCs per HC, and the output layer had one HC with 10 MCs corresponding to digit labels (see Fig. 3). The dynamical update equations are implemented in discrete steps using the forward Euler method. Each sample was clamped to the input layer for N_{sample} iterations of time-step Δt, and the training was performed for N_{usup} epochs of the training set. The time constants τ_k and τ_p were scaled by the total training time per epoch, that is, $\tau_k = \tau_k^o N_{train} N_{sample} \Delta t$ and $\tau_p = \tau_p^o N_{train} N_{sample} \Delta t$. The parameters used in the simulation are listed in Table I. All the results presented here are the mean and standard deviation of the mean squared validation error over 10 random runs of the network, unless stated otherwise. The simulations were performed on code parallelized using MPI on 2.3 GHz Xeon E5 processors and the training process took approximately two hours per run.

TABLE I. Model Parameters

Symbol	Value	Description
N_{HC}	30	Number of HCs in hidden layer
N_{MC}	100	Number of MCs per HC in hidden layer
Δt	0.01	Computation time step
μ	10	Mean of Poisson distribution for initializing MCs
γ	1	Softmax gain
k_{half}	-100	Bias gain when marginal is $1/2\, p_{MaxEnt}$
τ_p^o	0.5	Multiplier for learning time-constant
τ_k^o	0.1	Multiplier for bias gain time-constant
p_M	0.1	Probability of connections from input to hidden layer
N_{usup}	5	Number of epochs of training for unsupervised learning
N_{sup}	25	Number of epochs for training the BCPNN classifier

A. *Bias regulation*

We evaluate the bias regulation mechanism by measuring the accuracy while varying the relevant parameters: k_{half} is changed from -10 to -100 in steps of -10, and τ_k^o from 10^{-2} to 10^{1} in exponential steps of 10. Results are shown in Fig. 4a. The validation accuracy improves consistently as k_{half} is lowered and converges at around $k_{half} < -50$ to 96.7. This suggests that our bias regulation mechanism effectively improves the representations.

To quantitatively assess the effect of k_{half} on the marginals $p(z_j)$, we compute the marginal entropy of each HC, $H(p(Z_j)) = \sum_{z_j} p(z_j) \log p(z_j)$, and plot the histogram of this entropy over the 30 HCs (Fig. 4c). Note that for the marginals, higher entropy is preferred since it indicates all the MCs in a HC are utilized evenly, and p_{MaxEnt} was our target while designing the bias regulation mechanism. Even though the number of samples used in plotting this histogram is low (N_{HC}), it clearly shows that lowering the k_{half} increases the entropy of all the HCs.

The marginals give the overall utilization of the MCs over the training set, and we have evaluated it by measuring the entropy of this marginal distribution with respect to the parameter that regulates the bias (k_{half}). However, the marginals by themselves cannot give the complete picture of the representations since they do not take into account how well the representations differentiate between samples. For example, if all the MCs had a posterior of p_{MaxEnt} for all the input samples, the resulting high marginal entropy would not reflect a desirable scenario.

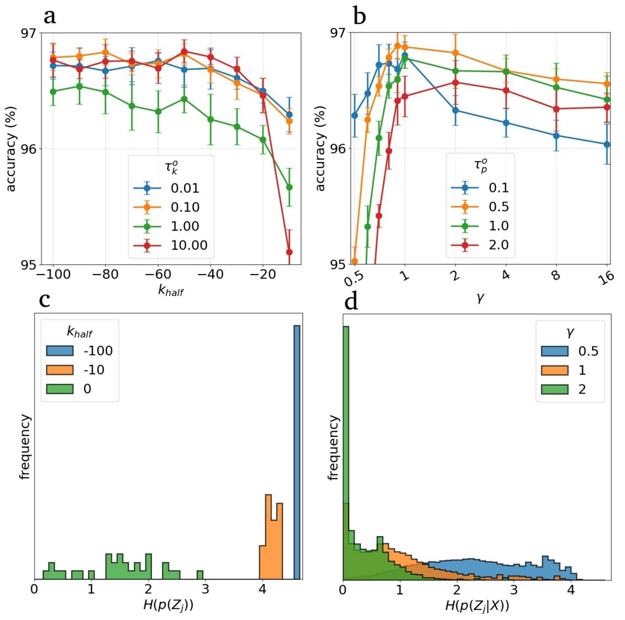

Fig. 4: (a) Accuracy results (mean and standard deviation for 10 random runs) on the MNIST validation set as a function of k_{half} for different values of time constant of bias gain τ_k^o; (b) Accuracy results as a function of softmax gain γ for different values of learning time constant; (c) Histogram of marginal entropy $p(Z_j)$ of hidden layer HCs for different k_{half}; (d) Histogram of conditional entropy $p(Z_j|X)$ for different values of softmax gain γ.

We measured the entropy of the posterior distribution of the MCs conditioned on each input sample, that is, $H(p(Z_j|X))$. Contrary to the entropy of the marginals, we expect this entropy to be as low as possible as we want the posteriors in the hidden layer to be certain about the conditioned input sample. The hyperparameter that controls this is the softmax gain γ. We computed the conditional entropy of all HCs per sample, and plotted the histogram over all samples in Fig. 4d. The histogram shows that the entropy predominantly has values < 2, whereas the maximum entropy is around $log(p_{MaxEnt}) \approx 4.6$ for $\gamma = 1$. This confirms that the bias regulation does not force the representations to have high marginal entropy at the cost of making all posterior per sample have high entropy. Fig. 4b shows an interesting relationship between accuracy of the representations and the softmax gain γ. One would expect low values of γ to have poor performance since we "flatten" the posteriors to be equal in value, and thereby, losing information about the input sample. However, high values of γ (>1) also worsen the performance, that is, having "winner-take-all" like activity regulation does not necessarily imply better representations.

B. *Structural plasticity*

We first report the performance of the network without structural plasticity, that is, the connections M are initialized randomly with probability p_M and left unchanged during the course of learning. The train and validation accuracies over 10 random runs of the network are 95.33 ± 0.03 and 93.83 ± 0.04 respectively.

Adding structural plasticity produced a meaningful set of receptive fields, which indicate the regions of the input space that drive hidden units. In Fig. 5 we visualize receptive fields, for four randomly chosen HCs and a subset of the corresponding MCs in the hidden layer. Notice that we obtain rather contiguous patches even though no spatial structure of images were presented. The receptive fields of MCs also seem to capture diverse features such as lines and strokes.

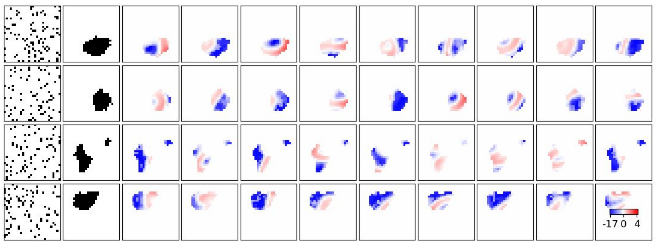

Fig. 5: Receptive fields. Each row corresponds to a randomly chosen HC of the hidden layer and the constituent MCs. First column shows the receptive field of the HC before training and second column after training (black means $M = 1$). The remaining columns show the receptive field of nine randomly chosen MCs in the HC.

Additionally, structural plasticity involved computing the mutual information between each input MC and hidden HC (Equation 7) and normalizing by the fan-out of input MC (Equation 8). This normalization ensured that all the input MCs have approximately equal fan-outs. We also separately ran the network without this normalization, which resulted in poor classification performance, with train and validation accuracy of 91.7 ± 0.22 and 90.4 ± 0.31 respectively for 10 random runs of the network.

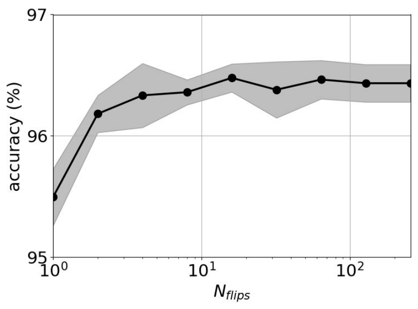

Fig. 6: Validation accuracy (mean and standard deviation for 10 random runs) results as a function of number of receptive field flips per epoch.

In order to examine the effect of the number of flips per epoch, N_{flip}, during learning of the receptive fields on the quality of hidden representations, we measured the validation accuracy while varying from 1 to 258 exponentially in steps of 2. Fig. 6 shows that the accuracy converges for $N_{flip} \geq 16$.

C. *Classification with Go and No-Go pathways*

Table II shows the accuracy when learning with the following strategies: (i) Go, (ii) No-Go and, (iii) Go + No-Go. Go and No-Go strategies perform well individually, but Go + No-Go performs slightly better.

TABLE II: Classification Results

Architecture	Accuracy (train)	Accuracy (validation)
Go	98.03 ± 0.28	96.40 ± 0.10
No-Go	97.21 ± 0.13	96.23 ±0.13
Go + No-Go	99.10 ±0.63	96.52 ± 0.08

Using the Go + No-Go classifier and the parameter set we found best suited based on validation (Table I), we report the train and test accuracies as 99.10 ± 0.63 % and 96.49 ± 0.12%, respectively.

VII. Discussion

We have demonstrated that the proposed network model can perform unsupervised representation learning using biologically plausible local learning rules. We made our assessment relying on the assumption that the saliency of representations is reflected in their class dependent separability, which can be quantified by classification performance (similar to [3, 24]). The performance on MNIST is significantly lower than the "superhuman" deep learning methods. Learning representations without supervised fine-tuning is a harder task compared to similar networks with end-to-end backprop training, since the information about the label corresponding to each sample cannot be utilised. Consequently, representations learnt with unsupervised methods cannot be expected to offer better class separability than the classification performance reported by supervised end-to-end approaches. We show that the BCPNN method scored around 96.5%, which is slightly worse compared to the 98.5% accuracy of networks with one hidden layer trained with end-to-end backprop [29]. However, we consider the modular architecture, sparse connectivity, lower complexity of our correlation based brain-like learning approach has a potential for high robustness, good scaling and low-power hardware implementations (see [30] for VLSI design of BCPNN).

It is important to note that the unsupervised learning methods introduced here are proof-of-concept designs and not meant to directly model some specific biological system or structure. Yet, they may shed some light on the hierarchical functional organization of e.g. sensory processing streams in the brain.

Further work will focus on comparing functionality and performance to other popular unsupervised learning networks such as AEs, RBMs, and the network by Krotov and Hopfield [31], as well as extending our architecture with a brain-like deep structure and recurrent connectivity.

Acknowledgment

Funding for the work is partially received from Swedish e-Science Research Centre (SeRC) and the European Commission H2020 programs, Grant Agreement No. 800999 (SAGE2) and No. 801039 (EPiGRAM- HS). The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the PDC Center for High Performance Computing, KTH Royal Institute of Technology.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] C. Pehlevan and D. B. Chklovskii, "Neuroscience-Inspired Online Unsupervised Learning Algorithms: Artificial Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6. pp. 88–96, 2019, doi: 10.1109/msp.2019.2933846.

[3] B. Illing, W. Gerstner, and J. Brea, "Biologically plausible deep learning — But how far can we go with shallow networks?," *Neural Networks*, vol. 118. pp. 90–101, 2019, doi: 10.1016/j.neunet.2019.06.001.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *J. Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[7] J. C. R. Whittington and R. Bogacz, "Theories of Error Back-Propagation in the Brain," *Trends Cogn. Sci.*, vol. 23, no. 3, pp. 235–250, Mar. 2019.

[8] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nat. Rev. Neurosci.*, Apr. 2020, doi: 10.1038/s41583-020-0277-3.

[9] R. J. Douglas and K. A. C. Martin, "Recurrent neuronal circuits in the neocortex," *Current Biology*, vol. 17, no. 13. pp. R496–R500, 2007, doi: 10.1016/j.cub.2007.04.024.

[10] K. Doya, *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2007.

[11] A. Sandberg, A. Lansner, K. M. Petersson, and O. Ekeberg, "A Bayesian attractor network with incremental learning," *Network*, vol. 13, no. 2, pp. 179–194, May 2002.

[12] A. Lansner, S. Benjaminsson, and C. Johansson, "From ANN to Biomimetic Information Processing," *Biologically Inspired Signal Processing for Chemical Sensing*. pp. 33–43, 2009, doi: 10.1007/978-3-642-00176-5_2.

[13] P. Berthet, J. Hellgren-Kotaleski, and A. Lansner, "Action selection performance of a reconfigurable basal ganglia inspired model with Hebbian-Bayesian Go-NoGo connectivity," *Front. Behav. Neurosci.*, vol. 6, p. 65, Oct. 2012.

[14] S. Benjaminsson, P. Fransson, and A. Lansner, "A novel model-free fMRI data analysis technique based on clustering in a mutual information space," *Frontiers in Neuroinformatics*, vol. 3. 2009, doi: 10.3389/conf.neuro.11.2009.08.028.

[15] M. Schain *et al.*, "Arterial input function derived from pairwise correlations between PET-image voxels," *J. Cereb. Blood Flow Metab.*, vol. 33, no. 7, pp. 1058–1065, Jul. 2013.

[16] R. Orre, A. Lansner, A. Bate, and M. Lindquist, "Bayesian neural networks with confidence estimations applied to data mining," *Computational Statistics & Data Analysis*, vol. 34, no. 4. pp. 473–493, 2000, doi: 10.1016/s0167-9473(99)00114-0.

[17] M. Lundqvist, P. Herman, and A. Lansner, "Theta and Gamma Power Increases and Alpha/Beta Power Decreases with Memory Load in an Attractor Network Model," *Journal of Cognitive Neuroscience*, vol. 23, no. 10. pp. 3008–3020, 2011, doi: 10.1162/jocn_a_00029.

[18] F. Fiebig and A. Lansner, "A Spiking Working Memory Model Based on Hebbian Short-Term Potentiation," *The Journal of Neuroscience*, vol. 37, no. 1. pp. 83–96, 2017, doi: 10.1523/jneurosci.1989-16.2016.

[19] P. J. Tully, M. H. Hennig, and A. Lansner, "Synaptic and nonsynaptic plasticity approximating probabilistic inference," *Frontiers in Synaptic Neuroscience*, vol. 6. 2014, doi: 10.3389/fnsyn.2014.00008.

[20] F. Fiebig, P. Herman, and A. Lansner, "An Indexing Theory for Working Memory Based on Fast Hebbian Plasticity," *eneuro*, vol. 7, no. 2. pp. ENEURO.0374–19.2020, 2020, doi: 10.1523/eneuro.0374-19.2020.

[21] V. Mountcastle, "The columnar organization of the neocortex," *Brain*, vol. 120, no. 4. pp. 701–722, 1997, doi: 10.1093/brain/120.4.701.

[22] K. Rockland, "Five points on columns," *Frontiers in Neuroanatomy*. 2010, doi: 10.3389/fnana.2010.00022.

[23] A. Lansner, "Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations," *Trends in Neurosciences*, vol. 32, no. 3. pp. 178–186, 2009, doi: 10.1016/j.tins.2008.12.002.

[24] D. Krotov and J. J. Hopfield, "Unsupervised learning by competing hidden units," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 116, no. 16, pp. 7723–7731, Apr. 2019.

[25] G. Lindsay, "Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future," *J. Cogn. Neurosci.*, pp. 1–15, Feb. 2020.

[26] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks," *Artificial Intelligence Review*, vol. 49, no. 2. pp. 281–299, 2018, doi: 10.1007/s10462-016-9526-2.

[27] M. Butz, F. Wörgötter, and A. van Ooyen, "Activity-dependent structural plasticity," *Brain Research Reviews*, vol. 60, no. 2. pp. 287–305, 2009, doi: 10.1016/j.brainresrev.2008.12.023.

[28] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," *Nat. Rev. Neurosci.*, vol. 5, no. 2, pp. 97–107, Feb. 2004.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11. pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[30] D. Stathis *et al.*, "eBrainII: A 3 kW Realtime Custom 3D DRAM integrated ASIC implementation of a Biologically Plausible Model of a Human Scale Cortex," *arXiv:1911.00889*, 2019.

[31] N. B. Ravichandran, A. Lansner, and P. Herman, "Brain-like approaches to unsupervised learning of hidden representations - a comparative study," *arXiv:2005.03476*, 2020.