Cognitive Computation
https://doi.org/10.1007/s12559-022-10085-5

A Developmental Approach for Training Deep Belief Networks

Matteo Zambra[1,2] · Alberto Testolin[1,3] · Marco Zorzi[1,4]

Received: 24 February 2021 / Accepted: 15 November 2022

Abstract
Deep belief networks (DBNs) are stochastic neural networks that can extract rich internal representations of the environment from the sensory data. DBNs had a catalytic effect in triggering the deep learning revolution, demonstrating for the very first time the feasibility of unsupervised learning in networks with many layers of hidden neurons. These hierarchical architectures incorporate plausible biological and cognitive properties, making them particularly appealing as computational models of human perception and cognition. However, learning in DBNs is usually carried out in a greedy, layer-wise fashion, which does not allow to simulate the holistic maturation of cortical circuits and prevents from modeling cognitive development. Here we present *iDBN*, an iterative learning algorithm for DBNs that allows to jointly update the connection weights across all layers of the model. We evaluate the proposed iterative algorithm on two different sets of visual stimuli, measuring the generative capabilities of the learned model and its potential to support supervised downstream tasks. We also track network development in terms of graph theoretical properties and investigate the potential extension of *iDBN* to continual learning scenarios. DBNs trained using our iterative approach achieve a final performance comparable to that of the greedy counterparts, at the same time allowing to accurately analyze the gradual development of internal representations in the deep network and the progressive improvement in task performance. Our work paves the way to the use of *iDBN* for modeling neurocognitive development.

Keywords Unsupervised deep learning · Computational modeling · Cognitive development · Hierarchical generative models · Iterative learning

Introduction

Despite the fact that the most popular approach for training deep neural networks is based on supervised learning [1], the first demonstration of the potential of deep learning stemmed from the discovery of efficient unsupervised learning methods for stochastic neural networks known as Deep Belief Networks (DBNs) [2, 3]. Since their introduction, DBNs have been successfully used in many challenging tasks, ranging from computer vision [4] to acoustic modeling [5], traffic flow prediction [6] and breast cancer classification [7]. These energy-based models have some unique properties compared to other unsupervised deep learning approaches, such as the ability to represent compositional structure [8, 9] and the possibility to be interpreted in terms of well-established theoretical principles rooted in statistical physics [10].

The capability of learning deep generative models using Hebbian-like mechanisms also makes DBNs particularly relevant for cognitive modeling research [11]. Indeed, this class of models offers a principled account for the functional role of top-down processing supported by feedback loops, at the same time providing a bridge to higher-level descriptions of cognition in terms of Bayesian computations [12, 13]. Compared to shallow generative models, hierarchical generative networks allow to study the emergence of increasingly more complex representations of the sensory signal, thus allowing to simulate a wide range

✉ Alberto Testolin
alberto.testolin@unipd.it

✉ Marco Zorzi
marco.zorzi@unipd.it

1 Department of General Psychology and Padova Neuroscience Center, University of Padova, Via Venezia 8, Padua 35131, Italy

2 Department of Electric and Mathematical Engineering, IMT Atlantique, Brest, France

3 Department of Mathematics, University of Padova, Via Trieste, 63, 35121 Padua, Italy

4 IRCCS San Camillo Hospital, via Alberoni 70, Venice Lido, Italy

of high-level perceptual and cognitive functions, such as numerosity perception [14–16], letter perception [17, 18], orthographic processing [19], development of object-action associations [20] and the appearance of visual hallucinations caused by damage in cortical areas [21]. Sparse variants of DBNs have also been used to simulate physiological properties of neurons in the primary and secondary visual cortex [22]. Notably, inference algorithms for DBNs can be implemented using biologically realistic sampling schemes, which explain unique aspects of low-level brain dynamics [23] and can be efficiently reproduced in spiking models [24]. Finally, hierarchical generative models like the DBN offer important insights into the functional role of spontaneous brain activity, both in terms of top-down predictive signals during task execution and in terms of generative replays during rest [25].

However, learning in DBNs has traditionally relied on a greedy, layer-wise training approach: the connection weights of layer n are changed only after the layer $n-1$ has been fully trained. Moreover, the trained weights are frozen and do not further change while learning takes place at higher layers. Although efficient from a computational perspective, this learning modality is clearly implausible from a biological standpoint. Indeed, the greedy approach implies that information is not passed to any higher-order network until learning at the lower level can be stopped because it has reached a (somewhat arbitrary) criterion. Though brain development shows variability across regions and may peak at different times, synaptogenesis begins at about the same time in distant regions such as the visual and the prefrontal cortex [26]. Moreover, a substantial degree of plasticity is preserved in adulthood even in the visual cortex (see [27], for review). Finally, spontaneous brain activity, which is thought to be a manifestation of top-down dynamics of generative models [25], is already structured into distinct cortical networks at birth [28].

Besides these neurobiological considerations, another key limitation of the greedy training approach is that it makes the DBN unsuitable for modeling human development. Neural network models are particularly attractive for understanding developmental phenomena [29] because the trajectories in task performance or in the emergence of internal representations can be examined during learning and compared to human empirical data. Moreover, learning trajectories in neural network models can be analyzed as a function of initial starting conditions to study the emergence of developmental disorders [30]. In the cognitive modeling literature, however, simulations based on DBNs have focused on adult performance (i.e., fully trained models) and developmental investigations based on unsupervised learning have adopted alternative learning paradigms based on deep autoencoders [31] trained with error backpropagation [32] to circumvent this problem.

In this work we propose a novel learning scheme for tuning the entire hierarchy of connections in a DBN iteratively (hereafter, iDBN), using a variant of the original Contrastive Divergence (CD) learning algorithm [33]. Through extensive simulations on two different sets of visual stimuli, we demonstrate that the proposed approach can achieve the same final accuracy of the greedy counterpart, at the same time allowing for a precise tracking of the developmental trajectory of the models. We further show that an alternative developmental scheme based on full-stack propagation of top-down information does not converge to an optimal solution, suggesting that recurrent processing between adjacent layers is a key ingredient to successfully drive learning through local signals. In a first set of simulations we rely on the popular MNIST data set of handwritten digits [34], with the goal of validating the proposed iterative scheme on a well-known benchmark. In this setup, we also show that our developmental learning scheme can be extended to continual learning scenarios [35], where the model exploits interleaved learning to incorporate knowledge from another domain (in our case, handwritten letters) without incurring in catastrophic forgetting [36, 37]. Furthermore, we carry out graph theoretical analyses to investigate how structural properties of the network gradually emerge during learning. We finally consider a more recent data set consisting of images containing a variable number of items, which has been used to investigate the perception of visual numerosity in humans and machines [14, 16], to demonstrate how the iDBN can be used to simulate developmental trajectories in the acquisition of cognitive skills.

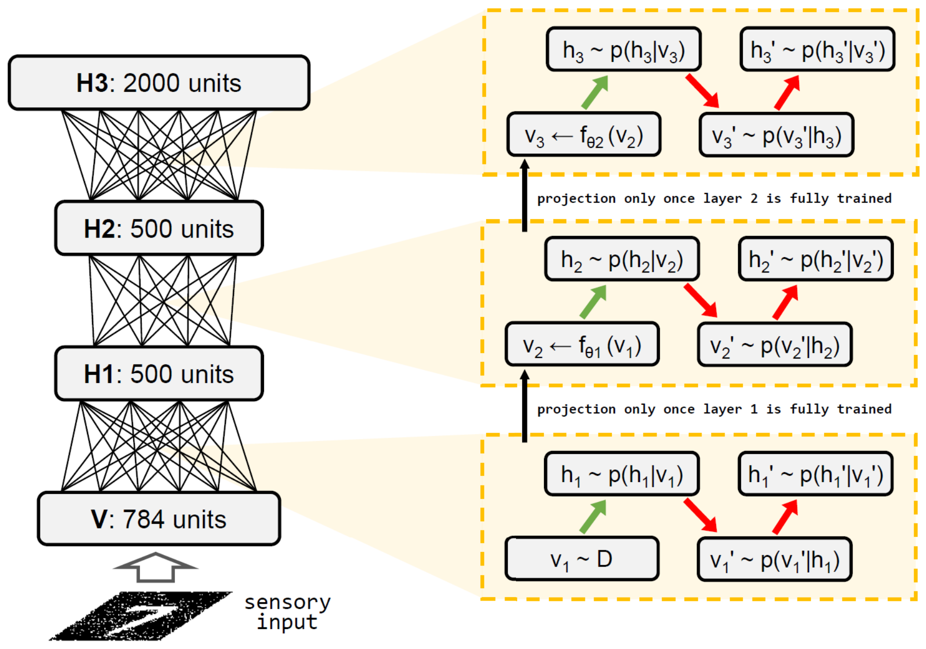
Methods

In this section we will briefly review the theoretical foundations of deep belief networks and describe their classical, greedy learning algorithm. We will then introduce our iterative learning approach and describe the materials and methods used in our simulations.

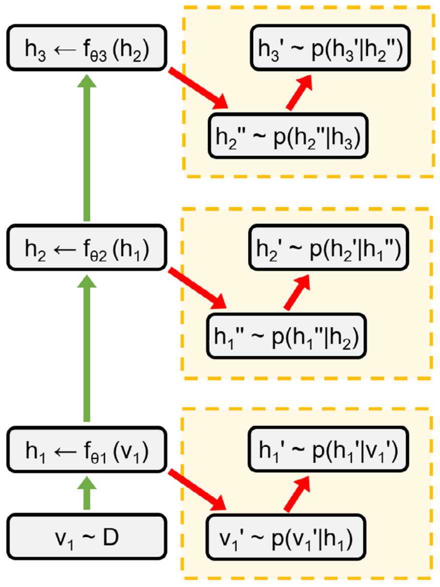
Deep Belief Networks

The building block of a DBN is the Restricted Boltzmann Machine (RBM [38]; for a recent overview see [39]), which is a bipartite network composed by two separate sets of neurons: *visible* neurons, which constitute the interface with the sensory environment and are thus usually clamped to the input data, and *hidden* (equivalently called 'latent') neurons that allow to capture high-order correlations in the data distribution. Learning in RBMs consists in discovering a set of latent features that can be used to compactly describe the statistical regularities in the data distribution, by creating an internal model of the environment that can

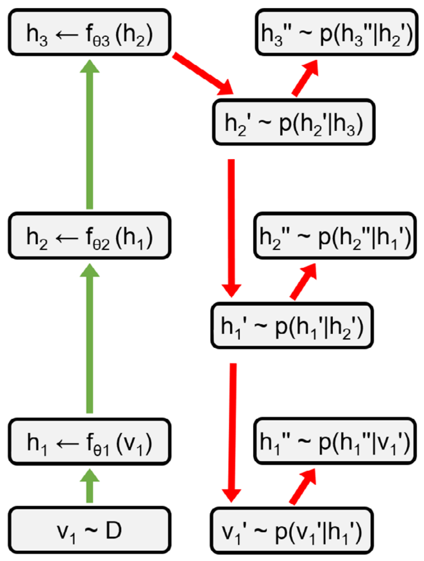
Fig. 1 Graphical representation of the architecture of a 3-layer Deep Belief Network and the learning schemes implemented in the present work. Green arrows represent bottom-up recognition connections, while red arrows represent top-down generative processing. Yellow boxes enclose local computations. We consider the case of CD1, CD-k can be recovered by repeating the sampling steps k times. $v \sim D$ identifies a data instance sampled from the training set and $\boldsymbol{h}_i$ represents the hidden activities of layer i. In the greedy scheme **a** hidden layers are trained sequentially, from bottom to top, and input signals are never projected into layer l unless learning at layer l - 1 is completed. In the iterative scheme **b** input signals are immediately propagated through the entire deep network, and top-down processing is performed locally at each layer to jointly learn all connection weights. In the full-stack scheme **c** both feed-forward propagation and top-down processing occur over the entire deep network



(a) Schematic illustration of the DBN architecture and the greedy, layer-wise learning scheme.



(b) Iterative learning scheme.



(c) Full-stack learning scheme.

be used to generate plausible activation patterns in the visible neurons. The lack of connections between neurons in the same layer makes it easy to compute the data-dependent and model-dependent statistics used in the Constrastive Divergence (CD) algorithm [33], because units in the same layer are conditionally independent given the activation of the other layer. Deep belief networks are created by stacking together several RBMs [2] (see Fig. 1a), thus allowing to exploit hierarchical composition of the features learned by the individual RBMs [40].

Greedy Layer-Wise Learning

For the sake of our argument, it is useful to make an explicit distinction between bottom-up *recognition* connections and top-down *generative* processing (represented by green and red arrows, respectively, in Fig. 1). During the recognition (also called "inference") phase, the sensory pattern is clamped on the visible neurons, and the hidden neurons are activated in a bottom-up fashion in order to infer the most likely configuration (i.e., activation pattern over the hidden neurons) that could have produced the observed data. During the generation phase, the visible neurons are not clamped to the data, and all neurons are activated in a top-down fashion in order to produce a plausible activation pattern, that is, to generate a sample from the internal model.

In the classical approach [2], the RBM constituting the bottom layer of the DBN is initially trained using the input data. Once this first layer has been *fully* trained, the weights of the first RBM are "frozen" and the input data set is projected into the activation space of the hidden neurons, thus creating a new training set that is used as input for the second RBM (black arrows in Fig. 1a). Once training of the second layer is completed, the data projection is made likewise to create the training set for the third layer; the procedure is repeated for all the layers constituting the DBN, as shown in Fig. 1a. Note that the data patterns projected into the deeper layers are created by computing the conditional probability distribution of the hidden neurons given the activation of the neurons observed in the layer below [41].

Iterative Joint Learning

In contrast to the greedy approach, the proposed iDBN learning algorithm attempts to update at once all the parameters of the hierarchical generative model, regardless of the depth of the respective layer (see Fig. 1b).

As noted before, in the greedy algorithm deeper layers are trained sequentially, using as input the projection of the data through the weights resulting from learning in the previous layers. In our novel iterative algorithm, instead, the training patterns for the deeper layers are immediately created following each sensory experience, by propagating the input across the entire processing hierarchy (green arrows in Fig. 1b). This process mimics the *fast feed-forward sweep* observed in cortical circuits, where neuronal activity is rapidly routed to a large number of visual areas after stimulus presentation [42, 43]. It should be noted that in our algorithm the complete feed-forward sweep occurs even during the initial learning phase, where all the connection weights are randomly initialized, which makes learning of the subsequent layers challenging since the data distribution (i.e., hidden unit activation) is non-stationary. Concurrently with the fast feed-forward sweep, top-down generative connections are locally used to reconstruct the data representations at each level of the hierarchy (red arrows in Fig. 1b), mimicking the kind of processing supported by recurrent and horizontal connections within cortical areas [42, 44][1].

A schematic pseudo-code that illustrates the implementation details of our iterative algorithm is reported below2.

Algorithm 1 Iterative learning in iDBN

```
1: Define model architecture and initialize the model parameters $\boldsymbol{\theta} = \{W_i, \boldsymbol{a}, \boldsymbol{b}_i\}$, $W_i$ are the weights matrices for layers $i = 1, \ldots, N_L$, $\boldsymbol{a}$ is the bias of the visible neurons, $\boldsymbol{b}_i$ are the hidden neurons biases for the layers $i = 1, \ldots, N_L$. For recommendations on a good choice of the parameters please refer to Reference [41]
2: Prepare the training set $\boldsymbol{X}_0 \leftarrow Data$
3: while not convergence do
4:     for layer $i = 1, \ldots, N_L$ do
5:         Store the input activations in a temporary variable $\boldsymbol{X}_{\text{tmp}} \leftarrow \boldsymbol{X}_i$
6:         for each $\boldsymbol{v}_i$ in $\boldsymbol{X}_i$ do
7:             $\boldsymbol{h}_i \sim p(\boldsymbol{h}_i|\boldsymbol{v}_i) = \sigma(W_i \boldsymbol{v}_i + \boldsymbol{b}_i)$, data-driven hidden activations
8:             $\boldsymbol{v}'_i \sim p(\boldsymbol{v}'_i|\boldsymbol{h}_i) = \sigma(W_i^t \boldsymbol{h}_i + \boldsymbol{a}_i)$, model-driven visible activations
9:             $\boldsymbol{h}'_i \sim p(\boldsymbol{h}'_i|\boldsymbol{v}'_i) = \sigma(W_i \boldsymbol{v}'_i + \boldsymbol{b}_i)$, model-driven hidden activations
10:            $\Delta W^+ = \langle \boldsymbol{v}_i \cdot \boldsymbol{h}_i \rangle$ and $\Delta W^- = \langle \boldsymbol{v}'_i \cdot \boldsymbol{h}'_i \rangle$
11:            $\Delta \boldsymbol{a}^+ = \langle \boldsymbol{v}_i \rangle$ and $\Delta \boldsymbol{a}^- = \langle \boldsymbol{v}'_i \rangle$
12:            $\Delta \boldsymbol{b}^+ = \langle \boldsymbol{b}_i \rangle$ and $\Delta \boldsymbol{b}^- = \langle \boldsymbol{h}'_i \rangle$
13:            $\Delta \boldsymbol{\theta}_i = \Delta \boldsymbol{\theta}^+ - \Delta \boldsymbol{\theta}^-$, with $\boldsymbol{\theta} = \{W, \boldsymbol{a}, \boldsymbol{b}\}$
14:            $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \Delta \boldsymbol{\theta}_i$
15:        end for
16:        Reset the input data to $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_{\text{tmp}}$
17:    end for
18: end while
```

Full-Stack Joint Learning

An alternative way to jointly learn all weights of a DBN could be to first propagate the input across the entire processing hierarchy (as in the first phase of iDBN) and then produce top-down reconstructions starting from the deepest layer back to the sensory layer (see Fig. 1c). This processing scheme is simpler to implement, but suffers from the vanishing gradient problem encountered in standard deep learning settings [45]. We use this full-stack developmental scheme as a benchmark for our iterative developmental scheme.

Simulations

MNIST Data Set

The same DBN structure is used and held fixed during the simulations for both greedy and iterative learning. In order to allow for a fair comparison, we maintained the same architecture and hyper-parameters of the original model [2], which was composed of one visible layer with 784 neurons, two hidden layers with 500 neurons each and a final hidden layer with 2000 neurons (see Fig. 1a). Connection weights are initialized with random values sampled from a Normal distribution $N(0, 0.01)$, while biases are intialized to zero. We also tested an alternative initialization scheme known as the Glorot initialization [47], in order to evaluate learning convergence under more advanced weight initialization strategies. The weights matrices initialized with the Glorot

[1] It should be noted that, following the initial feed-forward sweep, the generative phases at subsequent levels of the hierarchy still occur sequentially, because the model-driven activation of hidden neurons at layer t should not interfere with model-driven activation of the

Footnote 1 (continued)

same neurons at layer $t + 1$. Other families of generative models, such as the Deep Boltzmann Machine [46], resolve this potential interference by incorporating top-down influences during learning, but nevertheless require a greedy training strategy.

scheme are multiplied by a factor of 0.1 to make them compatible with the range observed in the Normal initialization. The learning rate is set to $\lambda = 0.01$ and the weight decay coefficient is set to $\alpha = 0.0001$. The momentum parameter ν is set to 0.5 in the initial learning stage and then updated to 0.9 after 5 epochs of training. The loss is minimized using standard stochastic gradient descent, implemented through CD1 learning. The model is trained for 50 epochs. Further tests, using both Normal and Glorot initialization schemes, also included dropout regularization [48] with the probability of unit presence set to $p = 0.1$.

The first quantity inspected to evaluate the quality of the learned models is the accuracy of a linear readout at each hidden layer: since in DBNs the input patterns are non-linearly transformed from one layer to the next one, the internal representations are supposed to become more linearly separable as we move up in the hierarchy [11]. The accuracy of a simple Ridge classifier is used as a measure of separability.

The performance of the trained models is then assessed in three image generation tasks: 1) reproduction of clean images, 2) completion of partially occluded images and 3) denoising of images corrupted by noise. In the second case, an arbitrary number of subsequent rows in the image matrix are set to zero (i.e., turned to black). In the latter case, all values $\{X_{ij}\}_{i,j=1}^{28}$ (being 28 the number of side-pixel of the images) are spoiled by adding Gaussian noise, that is $X_{ij} \leftarrow X_{ij} + \epsilon$, $\epsilon \sim N(0, 0.5)$. Images are fed to the visible layer of the DBN, propagated through all its hidden layers and then fed back to the visible layer through feedback connections. In this way the noisy / corrupted samples can be adjusted according to the internal model learned by the DBN. Original samples and the corresponding reconstructions are quantitatively compared using mean squared error (hereafter MSE) between input patterns and reconstruction. For each case, the error is computed as $\|X^0 - X^r\|$, where the superscript 0 denotes the original sample and r means that the image has been reconstructed (reproduced, recreated or denoised) and $\| \cdot \|$ denotes an L^2 norm. Model performance is averaged over 10 model runs with different random initialization in order to assess the robustness of the analyses. We also visualize the receptive fields of neurons at different hidden layers to qualitatively assess the type of features (i.e., internal representations) learned by the DBNs.

Continual Learning

To further support the cognitive validity of our approach, we investigate whether the proposed developmental algorithm could effectively deal with a challenging continual learning scenario, where the DBN should learn a generative model of data distributions provided incrementally, without forgetting knowledge obtained during the preceding stages. In such scenario it is well known that neural networks suffer from catastrophic forgetting, whereby knowledge learned during the subsequent stages completely disrupts previously learned information (for review, see [37]). Several solutions have been proposed to mitigate this issue, and here we specifically consider one that can be readily implemented within our framework: *interleaved learning* [49]. We consider a somewhat simplified version of interleaved learning, where unsupervised learning during a second training phase takes advantage of both patterns from the new target distribution and patterns belonging to the previous data set. More advanced implementations could exploit deep generative replay [50] to directly sample previously learned patterns from the hierarchical generative model, allowing to build the mixed data set in a more data-efficient way.

Following the first stage of unsupervised learning on handwritten digits, the DBN is exposed to a subset of handwritten letters from the EMNIST data set [51]. In one setup the DBN is trained sequentially, which means that only letter patterns are used to train the network during the second stage. In the interleaved setup, instead, during the second stage the unsupervised training set includes both digits and letters. In order to balance the number of patterns and output classes, we randomly sample 20,000 digits from the MNIST data set and we evenly sample 20,000 uppercase letters from the first 10 EMNIST classes. Continual learning performance is probed by monitoring both digit recognition accuracy (using the readout classifier trained during the first learning stage) and letter recognition accuracy (training a new readout classifier on the EMNIST patterns).

Graph Analysis

Recent work has shown that graph theory can be successfully used to study deep learning models from a network science perspective [52, 53]. In order to investigate how topological properties of the graph derived from the DBN might emerge during the course of unsupervised learning, we thus performed a graph analysis on the structure of the deep network during learning over the MNIST data set.

From the trained DBN we extract the weights matrices and define a graph having the same architecture and connections weights as the DBN. A methodological difficulty is posed by the continuity of the connection weights: in order to determine the nodes degrees, we thus prune the network by *binarizing* the connections according to a suitably chosen cutoff threshold. A sensible choice of the cutoff threshold is a value that allows to remove redundant connections and to keep those that contribute most to the signal propagation through the network. Note that the concept of "redundant" and "important" connections is largely arbitrary and not obvious to assess. For the sake of the structural analysis, however, this choice is based on the numerical magnitude of the connections strengths. A range of such thresholds has been set to $\{0.2, 0.4, 0.6, 0.8, 1, 1.25, 1.5\}$, and structural analysis has been performed for each of these values. A cutoff threshold of c discards the weights in the interval $[-c, c]$, while those outside the interval are kept for the network analysis.

As customary in network science, the structural properties of the "real" networks (DBNs) are compared with the same properties observed in a random counterpart. A random "replica" of the network is generated according to its architectural characteristics (such an number of nodes) and its local properties (e.g., mean node degree or edge probability). The process is composed of two main steps: 1) generate a random replica of the real network and 2) perform the structural analyses on both the instances. In our case, the real network was compared with an analogous binomial graph generated using the probability of edge existence, computed as the ratio between number of effective edges and the maximum number of potential edges (which depends of the number of node couples). This random replica is generated in such a way to have the same architecture of the DBN, in particular it is a stack of bipartite binomial graphs with the same number of nodes as the DBN layers. Due to this architecture, the node connectivity is constrained by the number of nearby layers. For example, one node of the first layer is not allowed to be connected with one node of the third layer or superior. Such graph structure poses a problem in the characterization of the nodes degree distribution, which is used in network science to evaluate whether a graph is random or derived from a real network. These latter (might them be natural, biological, technological or social networks) typically have a degree distribution well described by a power-law [54], while random graphs have a degree distribution that depends on the method they are generated with. For example, binomial random graphs have a degree distribution that follows, by design, the Binomial distribution. In our setup, the constrained nodes connectivity would lead the normalization of the degree distribution to have such architectural bias. To mitigate this effect, we chose to weight the degree of each node according to its potential maximum degree, as described in detail in Appendix C.

Numerosity Data Set

The "numerosity" data set was first introduced by Stoianov and Zorzi [14], who demonstrated that the approximate number of objects in a visual scene can be estimated by a hierarchical processing architecture that learns to extract increasingly more abstract representations from the sensory input in a completely unsupervised way (sample images are provided in Fig. 7 in Appendix A). Here we focus on the development of "number acuity" in the network, which has been recently investigated using a developmental approach based on deep autoencoders [31]. Number acuity can be measured using a numerosity discrimination task, where the network is asked to classify any image in terms of containing a larger or smaller number of objects with respect to a given reference number. Also in this case, to ensure a fair comparison with the original model [14], we considered the same model architecture and task. The DBN is composed by a visible layer composed of 900 visible neurons, while the first and the second hidden layers have 80 and 400 neurons, respectively. We adopted a Normal initialization scheme, where weights are initialized with random values sampled from $N(0, 0.1)$. The learning rate and weight decay are set to $\lambda = 0.1$ and $\alpha = 0.0002$, respectively, and the initial and final momentum are set to $\nu = 0.5$ and 0.9, again with a momentum switch at epoch 5. The model is trained for 100 epochs.

To assess the number acuity of the network (also known as "internal Weber fraction"), a linear classifier is applied to the deepest layer of the model, with the goal of establishing whether the hidden neurons' activation correspond to an input image containing a numerosity larger than a reference number (i.e., 8 or 16). This task becomes trivial in the limit of the difference between the given numerosity n_i and the reference number N_{ref} being large. For example, it is easier to tell which numerosity is smaller among 4 and 16, but it is harder for 15 and 16. For this reason, each reference numerosity has an associated window of numerosities used for the comparison: $\{5, \dots, 12\}$ for $N_{\mathrm{ref}} = 8$ and $\{10, \dots, 24\}$ for $N_{\mathrm{ref}} = 16$, so that the ratios $r_i = n_i / N_{\mathrm{ref}}$ yield the range [0.65, 1.25]. The percentage of correct classifications is analyzed as a function of these numerical ratios: each value r_i has associated the percentage of correct classifications y_i. This ensemble of points (r, y) is used to fit a psychometric function corresponding to a logistic curve, defined by

$$y = 1 - \Phi(\mu = r, \sigma = \sqrt{2}\, w) \quad (1)$$

being Φ the cumulative distribution function of the Normal distribution and w the Weber fraction.

Our interest is to compare the progressive refinement of the Weber fraction during the unsupervised learning phase. The final performance can be directly compared with the Weber fraction achieved by a DBN trained using the greedy scheme [14], while the developmental trajectories can be compared to the learning curves recently reported for deep autoencoders [31].

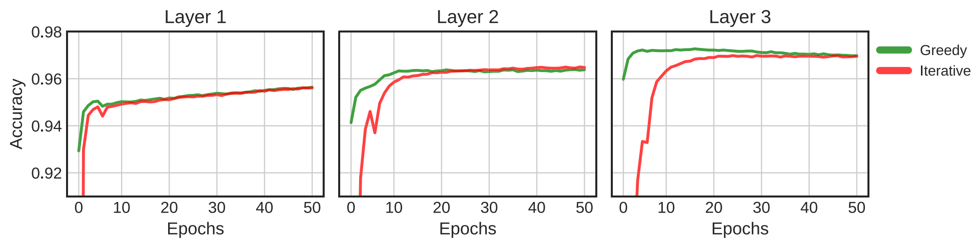
Results

MNIST Data Set

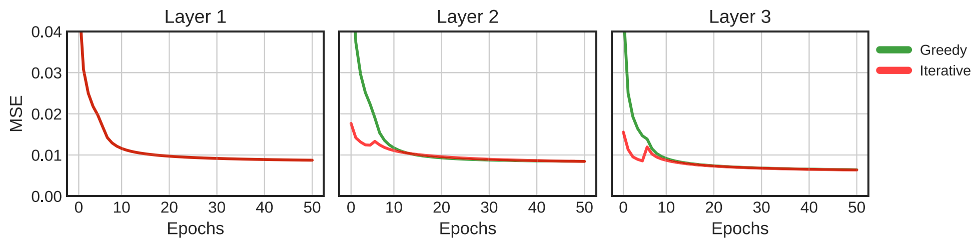
Readout Accuracy and Reconstruction Error Trends

Results discussed in this section only refer to the DBN configuration with Normal initialization and no dropout, since it turned out that differences with Glorot initialization and

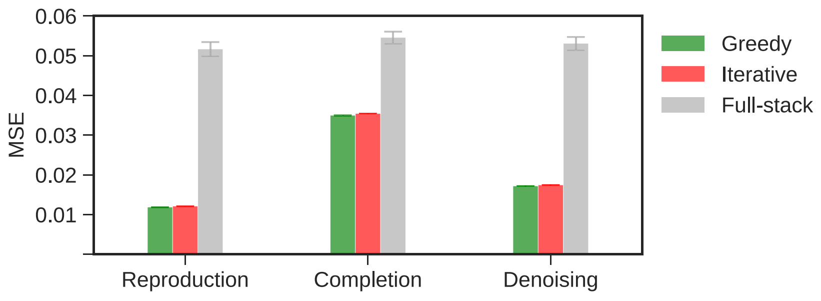
Fig. 2 Performance of the greedy vs. iterative learning schemes during learning (top and middle panels) and at the end of the unsupervised learning phase (bottom panel). For the latter case we also report the generation capabilities of the alternative developmental scheme based on full-stack propagation



(a) Readout profiles at different layers of the DBN hierarchy during training.



(b) Reconstruction errors at different layers of the DBN hierarchy during training.



(c) Average error on the image generation tasks (error bars represent standard error).

inclusion of dropout are negligible. The reader may refer to Appendix B for the complete results.

Figure 2a shows that the readout accuracy increases with depth, suggesting that during the course of unsupervised learning the internal representations became more disentangled (i.e., linearly separable). The reconstruction errors (MSE at each hidden layer) keep decreasing monotonically and eventually converge (Fig. 2b). Not surprisingly, results related to the first layer are almost perfectly overlapping for the greedy and iterative learning schemes: the activation of the visible layer corresponds to the raw data in both cases, thus the learned weights do not depend on the training modality. For the second and third layers, instead, the greedy scheme achieves higher readout accuracy in fewer epochs: however, this effect is due to the fact that the previous layers have been already completely trained, thus providing a head-start for the upper layers. At the end of training, the final accuracy is indistinguishable. Notably, the alternative developmental variant based on full-stack propagation does not exhibit the same optimal convergence, as highlighted by the poor reconstruction error measured in the first hidden layer (see Fig. 8 in Appendix A).

Generative Capabilities

As shown in Fig. 2c, at the end of the learning phase the greedy and iterative DBNs achieve equivalent generative capabilities (samples of generated images are provided in Fig. 9 in Appendix A). The completion task appears as the more challenging, probably because when entire regions of the images are corrupted it is difficult to generate plausible completions. The full-stack developmental version does not converge to a satisfactory generative model, as highlighted by its poor capability in all generation tasks.

Emergent Internal Representations

Receptive fields are useful to qualitatively inspect what kind of features are learned by the different layers of the hierarchical model during training. Such inspection is done by visualizing the connection weights of a given neuron in the input space [11]. The first hidden layer is easy to inspect, since we just need to plot the weights matrix of the first layer W_1. When it comes to neurons of the second hidden layer it is necessary to compose the weights matrices, so that it is still possible to represent the visualization in terms of the

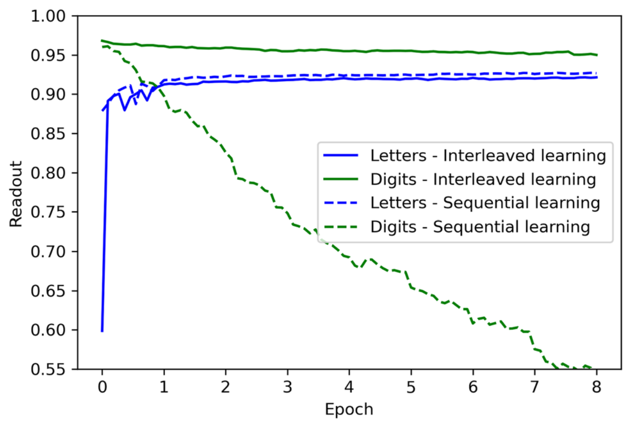


Fig. 3 Readout accuracy in the continual learning scenario. The sequential learning regimen is strongly affected by catastrophic forgetting, while interleaved learning incorporates information from the new distribution (Letters) while preserving previous knowledge (Digits). Classifiers are evaluated at 10 regularly spaced intervals during each unsupervised learning epoch

visible layer dimension. The weights matrices are simply multiplied, thus producing a linear combination: we can simply plot some chosen rows of the product $W_2 W_1$ to look at the receptive fields of the second layer neurons, $W_3 W_2 W_1$ for the third layer, and so forth. As shown in Fig. 10 in Appendix A, the greedy and iterative learning schemes developed qualitatively similar receptive fields across the entire processing hierarchy.

Resilience to Catastrophic Interference

As clearly shown in Fig. 3, the sequential learning setup (dashed lines) is dramatically affected by catastrophic interference: while the readout accuracy on the new letter data set increases, the performance of the classifier trained on the previous digits data set steadily drops as learning proceeds. On the contrary, the interleaved learning setup (solid lines) allows to easily incorporate knowledge from the new letter data set, achieving the same accuracy of the sequential setup, at the same time allowing to maintain previously learned knowledge, as demonstrated by the preservation of the readout accuracy for the digit data set.

Emergence of Structural Properties During Learning

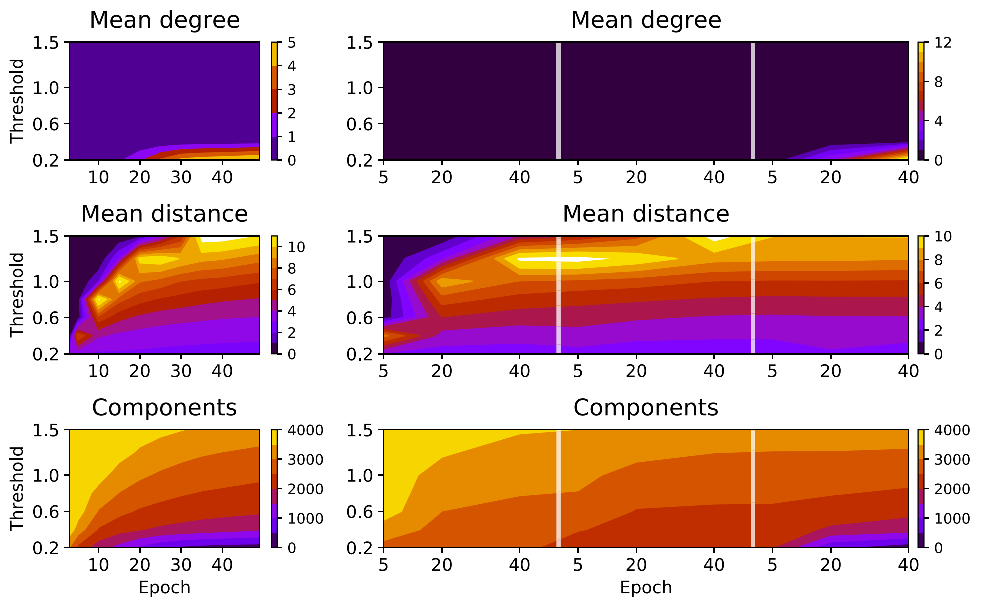
Typically, inspecting the degrees distribution of a given network gives a first idea of the nature of the system, at least to determine whether the network is random or real. Here, our main focus is on how the structural properties (among the other, also the mean degree) are affected by the learning dynamics. The degree distribution itself is not informative about the structural evolution that the network experiences during training but still could give useful insights about the structural differences of each network. We chose to track other global properties, e.g., mean degree, mean geodesic distance and number of connected components. Figure 4 displays these quantities, while Appendix C provides a broad explanation on the degrees distribution and how to approach its evaluation in this constrained-architecture setup.

Results suggest that the deep network undergoes a substantial transformation during unsupervised learning. A major difference lays in the mean degree evolution during training. Referring to Fig. 4a, non-trivial network properties tend to emerge in later stages of learning, especially for the greedy network. This suggests that the iterative implementation favors the development of complex circuits within the network connections and eventually the emergence of larger components. This applies both to the mean degree and to the number of connected components (recall that an isolated node itself is considered a component). The visualization of the mean geodesic distance shows that the evolution of this property is similar in both the greedy and iterative cases, in particular we can observe a phase transition between an initial state in which all nodes are isolated (indeed an isolated node is considered a component in which the geodesic distance is zero) to the emergence of some components. Once the different component connect to each other and the connections strengthen, the mean geodesic distance decreases.

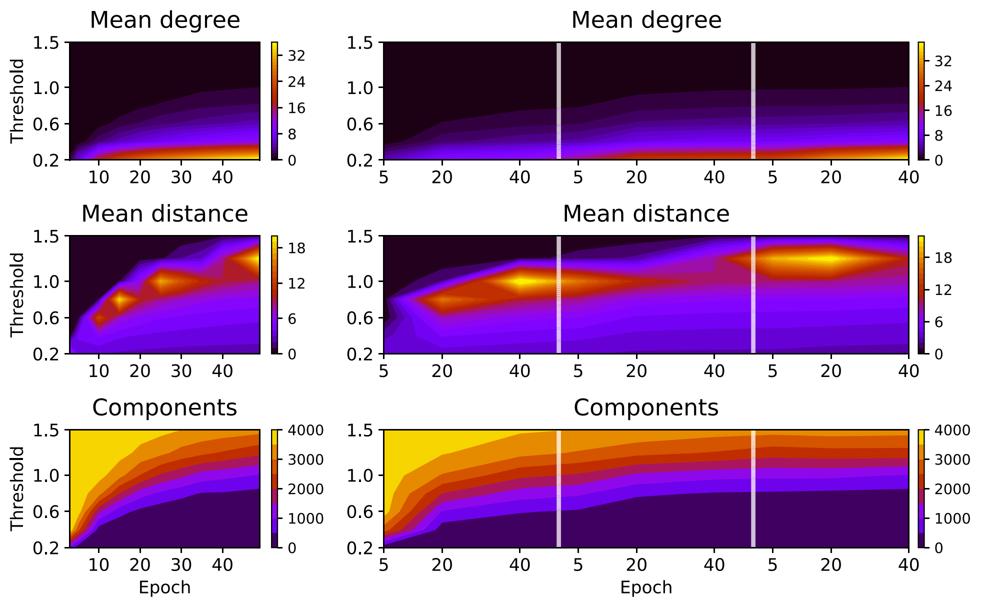
The second set of results, in Fig. 4b, displays the evolution of the same properties in the binomial replicas. The names “binomial greedy” and “binomial iterative” mean that the binomial counterparts are obtained using the probabilities of edge existence derived from the real greedy and iterative networks, in the epochs considered. The main difference is that both the greedy and iterative instances display the same overall behavior. Unsurprisingly, the visualizations of mean degree and connected components practically overlap. This expected results are given by the fact that there is no such thing as the effect of a learning dynamics that shapes the networks internal structure. In addition, the bottom panels (connected components) show that in random networks the formation of only one giant component is strongly encouraged. Note further that mean degree and mean geodesic distances attain larger values in binomial networks. This is indeed what one could expect: random networks do not have *hubs*, i.e., nodes with a larger degree with respect to the vast majority of the other nodes, which are instead a characteristic of real scale-free networks. The absence of few super-connected nodes implies a larger mean degree.

The results discussed above refer to the evolution of global network properties. As mentioned before, in network science it is well known that real networks exhibit a power-law degree distribution. As a further comparison between the real networks and their binomial counterparts, we thus choose to also inspect the degree distribution in both the greedy and iterative cases. For this test,

Fig. 4 Contour plots of mean degree, mean geodesic distance and number of connected components for the case of Normal initialization without dropout. Note that while the iterative iDBN algorithm allows to analyze the entire network since earliest learning stages, in the greedy case the upper layers remain untouched by the update rule until the lower layers are fully trained. This motivates the visualization choice of the right column: the dashed line represents the subdivision between layers, so that to display the trend of change in the whole network during the true learning time-span



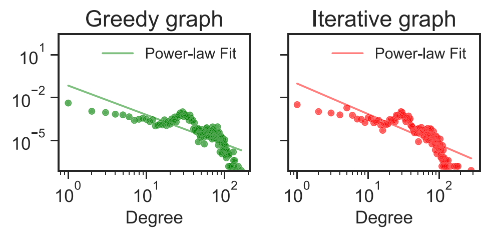
(a) Real network. Left column: Iterative DBN; Right column: Greedy DBN.



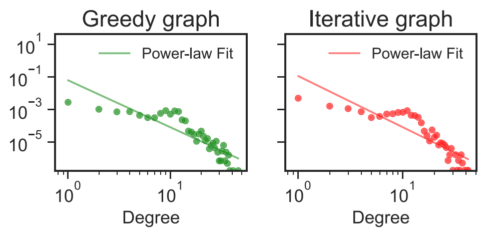
(b) Binomial replica. Left column: Iterative DBN; Right column: Greedy DBN.

we analyze the networks at the end of training and we also include the results obtained with the implementation of dropout, to see if the sparsity induced by dropout changes the network structure. Figure 5 shows the visualization of the degree distributions for real and binomial graphs for the case of Normal initialization, both with and without dropout. The effect of the architectural bias discussed above is particularly clear on the two leftmost upper and lower panels, despite the implementation of the modified degree distribution that accounts for the maximal potential connectivity for each node. This result suggests that the modified degree distribution should be further refined in order to model more accurately the distribution of nodes degrees; still, the degrees distributions show qualitatively different shapes, suggesting that learning dynamics in DBNs characterizes the derived network as a non-random graph. Interestingly, the effect of dropout is to make random replicas more structurally similar to real networks. The top and down right panels of Fig. 5 show that a larger connectivity for a small number of nodes is tolerated also in binomial graphs.

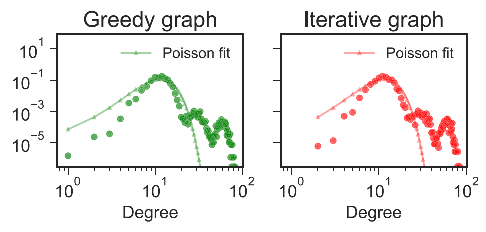
Fig. 5 Degrees distributions with cutoff threshold set to 0.4. The networks analyzed for this figure have been initialized with the Normal distribution; the case of Glorot initialization does not display significant differences



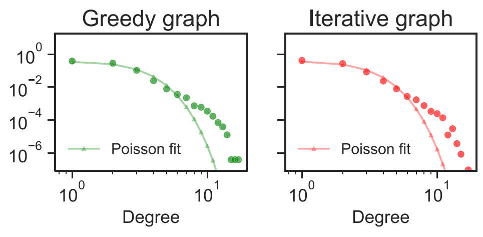
(a) Real DBN, no dropout.



(b) Real DBN, dropout with $p = 0.1$.



(c) Binomial replica, no dropout.



(d) Binomial replica, dropout with $p = 0.1$.

Numerosity Data Set

As shown in Fig. 6a the psychometric function observed at the end of the unsupervised learning phase is well aligned with the results obtained by Stoianov and Zorzi using the greedy training scheme [14]. The authors reported a Weber fraction value of 0.15, while we obtained a value of 0.17.

Concerning the developmental trajectory, as shown in Fig. 6b the Weber fraction trend displays a significant decrease during the learning period, especially at the early stages of development. This trend is similar to that observed during human development [55, 56]. Note that our curve has been obtained by averaging 20 model runs, training 5 different classifiers for each data point to collect more reliable statistics. Thus, the average values account for a population of 100 data points for each sample epoch. It is interesting to note that in the early stages the network might yield a worsening of the performance, and hence highly varying values of w. The behavior stabilizes to an asymptotic value in more advanced learning stages.

Figure 6c and d display the same data points, along with a power-law fit obtained using the method proposed by Testolin et al. [31]. Due to the initial zero value of the epoch time stamp, the basic functional form is actually a modified power-law [21]:

$$y = a(1 + sx)^b \tag{2}$$

The parameters a, b and s are fitted to the data points describing the progressive development of the Weber fraction. The resulting parameters are reported in Table 1: the fit closely follows the trajectory of Weber fraction, hence describing satisfactorily well the development of the number sense across the learning period of our networks.

Discussion

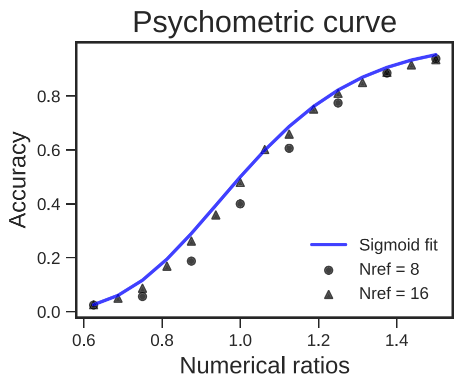
Our simulations demonstrate that deep belief networks can be trained iteratively, by jointly adjusting all the weights of the model hierarchy following observation of each sensory pattern (or minibatch of patterns). This innovative learning algorithm can be used in place of the traditional greedy, layer-wise learning algorithm in order to accurately track the developmental trajectory of the model. This allows to study how global properties of the network can gradually emerge during the course of learning, at the same time enabling a systematic comparison with biological developmental trajectories observed in empirical studies.

The proposed algorithm was first evaluated on the popular MNIST benchmark, where DBNs trained using the greedy algorithm have traditionally achieved very good performance. We showed that the DBN trained using our iterative algorithm was able to achieve a performance comparable to the greedy counterpart, both in terms of readout accuracy from the internal representations and in terms of reconstruction capabilities. We also probed the final models on a variety of generative tasks, which assessed the DBN ability to reproduce, complete and denoise corrupted input images. Also in this case, we did not observe significant differences between the greedy and iterative versions of the learning algorithm. On the contrary, the attempt to implement joint training of all weights by simply propagating signals across the full stack both in the bottom-up

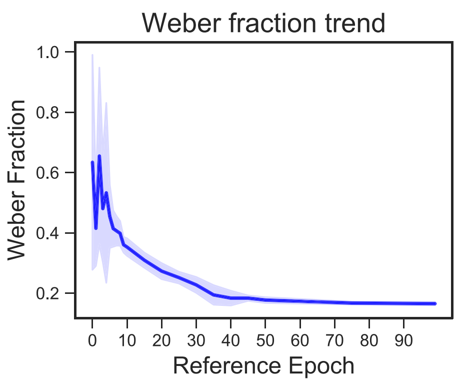
Table 1 Fitted parameters as in Eq. 2 for both the w values scaled, respectively, according to the method of S & Z [14] and TZM [31]

	a	b	s	R^2
S & Z	0.61	0.59	0.13	0.9
TZM	0.86	0.59	0.13	0.9

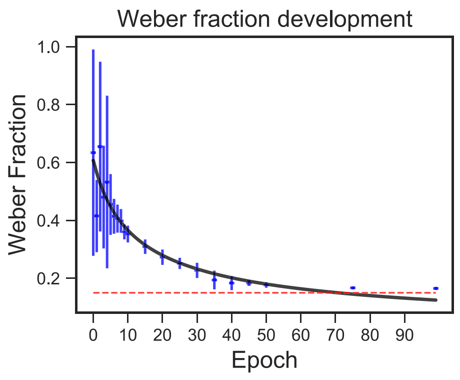
Fig. 6 Final number acuity and developmental trajectories of the DBN. Panel 6a reports the psychometric curve obtained from the numerosity discrimination task at the end of unsupervised iDBN learning. The Weber fraction measures the steepness of the curve, here $w = 0.17$. Panel 6b reports the trend of decay of the Weber fraction during unsupervised learning. The solid curve represents the average w values obtained in different simulation runs, while the shaded area represents the standard deviation. Panels 6c and 6d show the power-law fit according to Eq. 2. The dashed lines represent reference values for the final w from [14] and [31] respectively



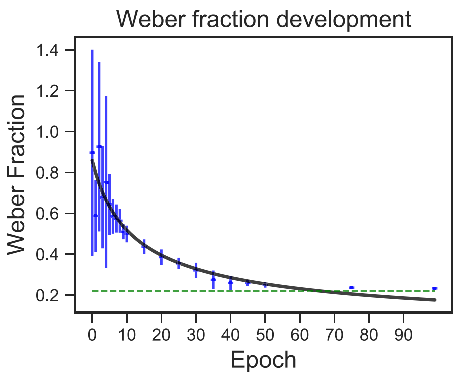
(a) Sigmoidal fit of the psychometric function.



(b) Trend of change of the Weber fraction.



(c) Values of w as in Stoianov and Zorzi [14], fitted according to a power function.



(d) Values of w rescaled as in Testolin et al. [31], fitted according to a power function.

and top-down learning phases led to poor convergence and unsatisfactory results. This reveals that, subsequently to the feed-forward propagation across the entire DBN hierarchy (which resembles the fast feed-forward sweep observed in cortical circuits [42, 43]), neurons' activation at the deepest layer cannot be fed all the way back to the visible layer in a (symmetrical) fast feedback sweep but need to be locally processed at each layer to compute the learning signals (as implemented in the iDBN). Notably, the latter scheme is also consistent with local recurrent processing supported by recurrent and horizontal connections within cortical areas [42, 44].

To further support the cognitive plausibility of the proposed developmental scheme, we also implemented a straightforward interleaved training approach to tackle continual learning tasks, demonstrating that the iDBN can be also extended to challenging scenarios that require to incrementally incorporate new knowledge in the deep network. This paves the way to the investigation of more plausible continual learning schemes that exploit the top-down generative properties of the DBN to sample the stimuli that are needed for interleaved training [57]. Moreover, we carried out an extensive analysis on the progressive development of structural properties in the DBN, by investigating the emergence of a variety of graph theoretical properties, such as degree, geodesic distance and connected components. Our iterative learning approach allowed to emphasize the gradual refinement of these properties at the global level, thus opening the possibility to more systematically study the topological development of such complex hierarchical systems.

We finally evaluated our iterative approach on a data set that has been recently exploited in a variety of cognitive models to simulate the perception of visual numerosity. Also in this case, the iDBN achieved a final accuracy comparable to the greedy counterpart, at the same time allowing for a precise tracking of the progressive development of the internal representations of the model. Behavioral performance, supported by a linear readout from the internal representations, can also be continuously tracked to monitor skill acquisition. Indeed, the learning curves resulting from our approach closely resemble the learning curves reported in experimental studies with human children, and also overlap with those reported in a recent developmental model based on deep autoencoders.

Conclusion

The scope of the present work was twofold. On the one hand, we presented a novel unsupervised learning algorithm for deep belief networks that allows to accurately track the progressive development of the internal representations of the model. We validated our algorithm on two prototypical benchmark domains, achieving results comparable to the state-of-the-art. On the other hand, we demonstrated that our iterative learning algorithm can be extended to more realistic learning scenarios, at the same time supporting the psychometric analysis of progressive changes in behavioral performance and the study of the gradual development of network properties from the point of view of network theory.

Future studies might take advantage of the proposed iterative algorithm to investigate how the emergence of developmental disorders can be related to impairments in both the initial conditions of the system and the subsequent learning phases. This would be particularly relevant for the study of widespread learning disabilities such as dyscalculia, for which we are still lacking a computational characterization [30]. It would also be useful to explore whether the proposed iDBN approach could be applied in other cognitive domains involving unsupervised deep learning. This would be valuable not only from a machine learning standpoint, but also from a cognitive modeling perspective: Indeed, our iterative algorithm allows to create developmental models that can be quantitatively validated against empirical data collected on humans. Similarly, the proposed graph analysis could be applied in developmental physiology to better understand how structural and functional properties of self-organizing networks might gradually emerge from unsupervised learning dynamics.

Appendix A: Supplementary Figures

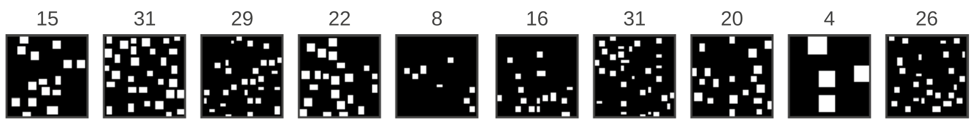


Fig. 7 Samples from the Numerosity data set, which contains 51200 images featuring a variable number of white rectangles drawn on a black background. Numerosity ranges from 1 to 32 and objects have variable position and dimension (see [14] for further details). The corresponding numerosity is reported on top of each image

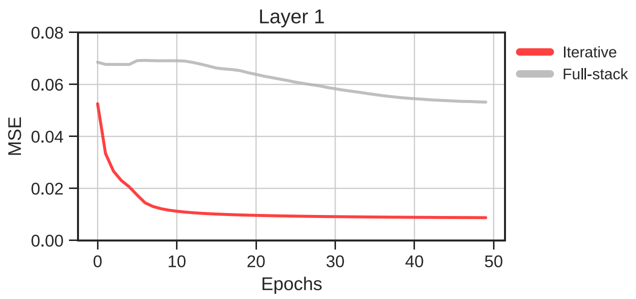
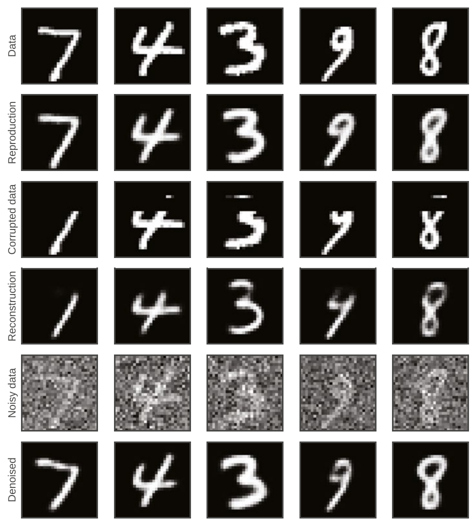
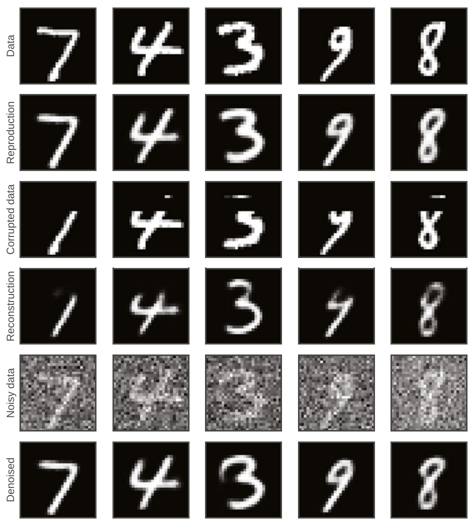


Fig. 8 Reconstruction error trend of the first hidden layer for the iterative (iDBN) vs. full-stack developmental schemes. We verified that convergence for the full-stack scheme did not improve even after prolonging learning for 100 epochs

Fig. 9 Examples from the image generation tasks. First row: original data. Second row: reproduction of the original data. Third row: Partially observed data. Fourth row: completion of partially observed data. Fifth row: noisy data. Sixth row: denoised data

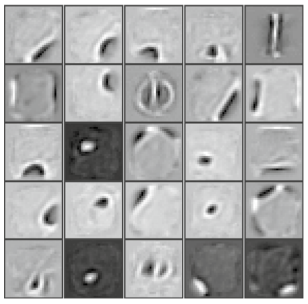


Greedy DBN.

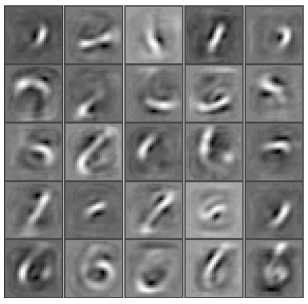


Iterative DBN.

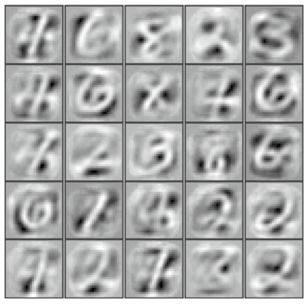
Fig. 10 Emerging receptive fields for the Normal initialization. Top row: Greedy algorithm. Bottom row: Iterative algorithm



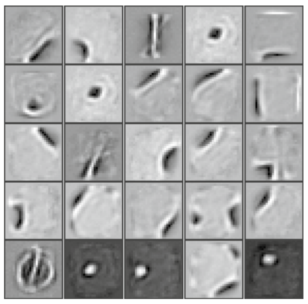
(a) Layer 1.



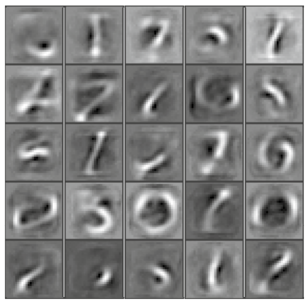
(b) Layer 2.



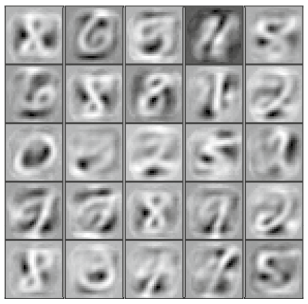
(c) Layer 3.



(d) Layer 1.



(e) Layer 2.



(f) Layer 3.

Appendix B: Robustness of the Iterative Learning Scheme Results

Results discussed in the main text about the equivalence in terms of performance between greedy and iterative training were focused on the Normal initialization configuration, with no dropout. To assess the robustness of our analyses, here we show that the same results are found when the weights are sampled Ã la Glorot and when dropout is added as a regularizer.

Glorot Initialization and Dropout

Figure 11 reports the readout and reconstruction profiles obtained with the Glorot initialization. The results show consistency with those presented in the main text. As mentioned before, the weight matrices are down-scaled by a factor 0.1 in such a way to bring these weights values in a range similar to the weights of Normal initialization.

We also evaluated the effect of dropout on our training scheme. Dropout is an effective regularization method that can be applied to RBMs [48]. The idea is to randomly silence some neurons during training, in such a way to prevent connections to over-learn. This can yield improved generalization capability, though in some cases it might also hinder learning performance. Bottom panels in Fig. 11 shows that also in this setup the overall performance of the model is not affected.

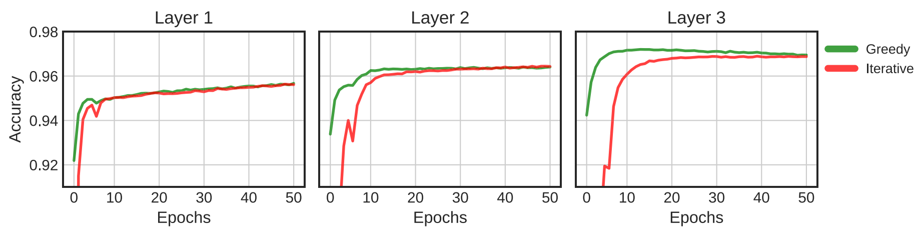
Appendix C: Graph Analysis in Depth

In “Graph analysis” we discussed the critical choice of the cut-off threshold and displayed aggregated data of the emerging global structural properties. Here the degrees distributions are discussed in finer-grained detail, since the graph structure extrapolated from a DBN poses some case-specific problems. Indeed, it is not straightforward to choose a model distribution to fit the emergent weights configuration, since during learning some connection strengths are significantly increased, leading to long-tail frequency distributions. Rather than searching or producing an ad-hoc probability distribution for the learned weights and then pruning the network according to some distribution-specific values (e.g. quantiles), an heuristic choice is rather to prune the network based on a user-defined cut-off threshold. In “Graph analysis”, a set of such meta-parameters is used jointly with the reference epochs to draw mean degrees, distances and connected components maps. Here the focus is on the degree distribution for a given cut-off threshold, which has different characteristics in random graphs and real networks [58].

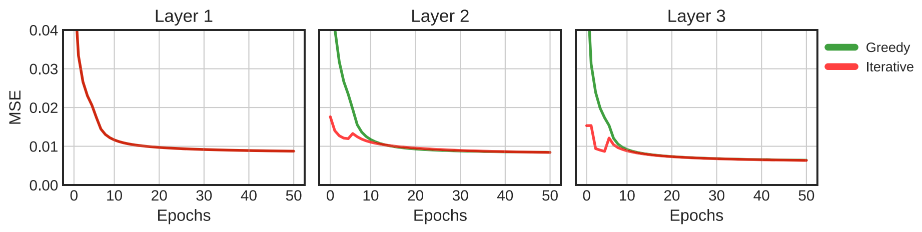
Degrees Distribution

Real networks typically exhibit a degree distribution than follows a power-law, expressed by:

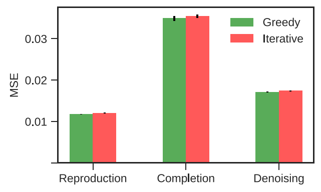
Fig. 11 **a**, **b** Performance of the greedy vs. iterative schemes during learning, for the Glorot weights initialization. **c–e** Generative accuracy of the greedy vs. iterative schemes at the end of learning, for combinations of Glorot initialization and dropout



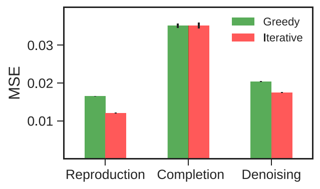
(a) Readout profiles at different layers of the DBN hierarchy during training.



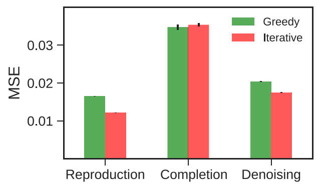
(b) Reconstruction errors at different layers of the DBN hierarchy during training.



(c) Generation tasks (Glorot, no dropout)



(d) Generation tasks (Normal, dropout)



(f) Generation tasks (Glorot, dropout)

$$p_k = a\,k^{-\gamma} \tag{3}$$

where k represents the degree, γ is the decaying exponent and a is a constant. Results in Fig. 5 were obtained by eliminating all the weights in the interval $[-0.4, 0.4]$, thus excluding the majority of connections. The power-law fit performs poorly for these DBN-like networks and the shape of the distribution deviates from the linear trend typical of scale-free networks, creating issues with the empirical way of computing the degrees distribution. Typically, one would count all the nodes having degree k, call this quantity N_k. The fraction of such nodes is:

$$p_k = \frac{N_k}{N} \tag{4}$$

where N is the number of nodes in the network and N_k is the fraction of nodes having degree k. This expression identifies a legitimate probability mass function, since $\sum_k p_k = 1$. The network architecture poses a subtle problem: while in real networks any node could be connected to virtually any other, here each node can be connected to all and solely those belonging to neighboring layers. Thus, each node should be associated with a potential maximum degree. This quantity is used to correct the degrees distribution, by penalizing a node depending on its maximum allowed connectivity, thus smoothing the resulting distribution. The nodes fraction is set to:

$$\tilde{p}_k = \frac{N_k}{N} \sum_{i=1}^{N} \mathbb{1}_{k_i = k} \frac{1}{\sum_{j=1}^{N} A_{ij}} \tag{5}$$

where $\mathbb{1}$ is the indicator function. This distribution is normalized computing the normalizing function numerically as $C = \sum_k \tilde{p}_k$. Assume that node i has degree k. Its relevance in the number of nodes N_k is weighted with its potential maximum degree, computed as the sum of the row i of the adjacency matrix A. The aim of this model is to normalize the nodes degrees according to the potential maximum degree of each node.

Binomial Replicas

To make the analysis more robust, we experiment a comparison with a synthetic null model having the exact same architecture as the DBN, created as a Binomial Random Graph (also known as Erdős-Renyj model, ER hereafter, [59]). The comparison between the degrees distributions of real network and binomial replicas is displayed in Fig. 5 in the main text. To generate the random network counterparts $G(N, p)$, the same architecture of the real network is used. The probability of edge existence p is then computed as the ratio between the number of the actual edges (once the network has been pruned) and the total number of connections (given by the architecture), $p = \frac{m}{M}$ [60]. The same probability is used to generate random links between all the four layers of nodes of the random network produced. The resulting replicas are random, but not strictly binomial: the degree distribution inevitably deviates from the Poissonian trend, and this is not (only) due to the fact that the number of nodes is relatively small (3784 nodes), but rather to the network bipartite structure.

Correction of Degrees Distribution

Due to architectural constraints, it is not possible to observe the idealized power-law and binomial degrees distributions for our networks. Binomial networks have a binomial degrees distribution and the probability mass function is expressed by:

$$P(k) = \binom{N-1}{k} p^k \,(1-p)^{N-1-k} \tag{6}$$

which means that a given node has k links with probability p, out of a total of $N-1$ links it could have if the network was fully connected, i.e. $p = 1$. The number of connections of a given node ranges between virtually zero and its potential maximum degree, which is the total number of neurons of the neighboring layers.

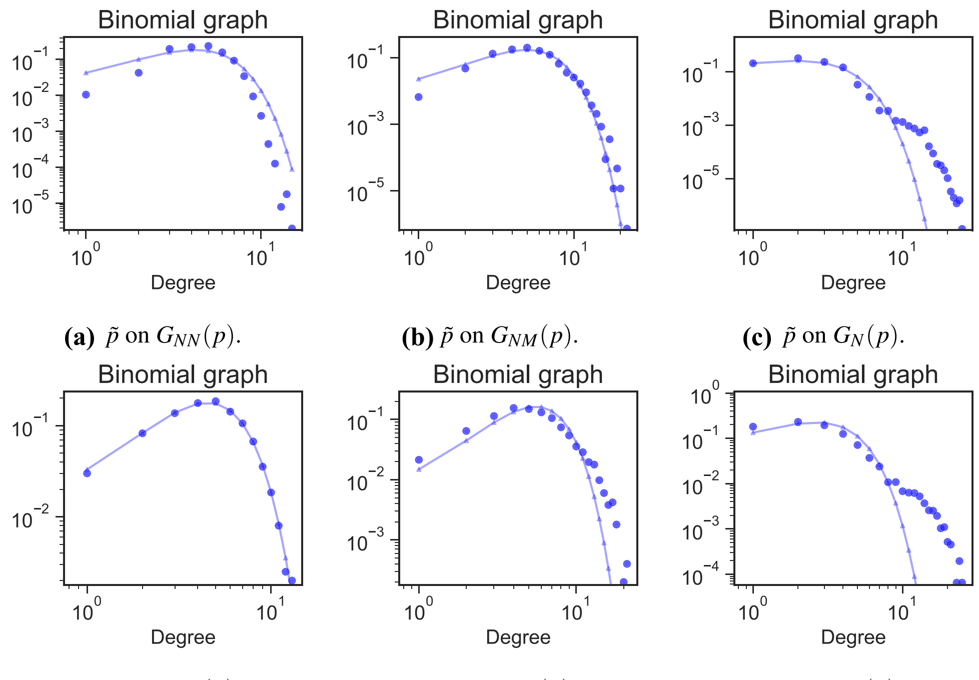
We propose a correction to normalize the node degrees in such a way to correct the architectural bias. The strategy chosen for generating the results in the main text is that of Eq. 5, however the choice of this model is not unique. For example, we could choose a model that sets the fraction of nodes having degree k to the corrective factor previously multiplied by N_k/N:

$$\tilde{q}_k = \sum_{i=1}^{N} \mathbb{1}_{k_i = k} w_i = \sum_{i=1}^{N} \mathbb{1}_{k_i = k} \frac{1}{\sum_{j=1}^{N} A_{ij}} \tag{7}$$

again being A the full-graph adjacency matrix. This fraction can be computed easily, then the actual distributions are computed normalizing the raw fractions $\tilde{p}$ and $\tilde{q}$ as follows:

$$P_k = \frac{\tilde{p}_k}{\sum_k \tilde{p}_k} \qquad Q_k = \frac{\tilde{q}_k}{\sum_k \tilde{q}_k} \tag{8}$$

The two model distributions have been chosen performing some observation on completely random graphs, completely independent of the probabilities of edge existence of the real network. The first synthetic graph $G_N(p)$ shares the same architecture of the DBN, with tunable probability of existence p, chosen small. A second test case involves a bipartite random graph with the same number of nodes N on both sides. The third test case involves a bipartite graph having N and M nodes on the two sides. The goal is to compare qualitatively both the distributions P and Q on these three test cases to check whether one or the other model distributions perform better. Figure 12 displays the results of these



(a) $\tilde{p}$ on $G_{NN}(p)$. **(b)** $\tilde{p}$ on $G_{NM}(p)$. **(c)** $\tilde{p}$ on $G_{N}(p)$.

(d) $\tilde{q}$ on $G_{NN}(p)$. **(e)** $\tilde{q}$ on $G_{NM}(p)$. **(f)** $\tilde{q}$ on $G_{N}(p)$.

Fig. 12 Experiments on the null models $G_{NN}(p)$, $G_{NM}(p)$ and $G_{N}(p)$. The value of the probability p is kept to 0.01, while $N = 1000$ for $G_{NN}(p)$, $M = 2000$ for $G_{NM}(p)$ and $G_{N}(p)$ retains the structure of the DBN

simulations. The experiments reveal that P adheres better to the Poissonian trend, thus motivating the choice of the model distribution P for the analyses presented in the main text.

Acknowledgements We are grateful to Michele De Filippo De Grazia for support with the simulations related to continual learning, and to Ivilin Stoianov for useful discussions about the iterative version of the developmental algorithm presented in this paper.

Author Contributions A.T. and M.Zo. contributed to the study conception and design. Computational simulations, data collection and analysis were performed by M.Za. The first draft of the manuscript was written by M.Za. and A.T. All authors commented on previous versions of the manuscript, and all authors read and approved the final manuscript.

Funding This work was supported by a Cariparo Excellence Grant 2017 (project "Numsense") to M.Zo.

Data Availability The complete source code of the iterative learning algorithm and additional data related to the analyses discussed in the current study are freely available through the following repository: https://github.com/CCNL-UniPD/iDBN.

Declarations

Ethics Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. LeCun Y, Bengio Y, Hinton GE. Deep learning, Nature. 2015:521.
2. Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–54.
3. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–7.
4. Lee H, Grosse R, Ranganath R, Ng AY. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning; 2009. p. 609–616.
5. Mohamed AR, Dahl GE, Hinton G. Acoustic modeling using deep belief networks. IEEE Trans Audio Speech Lang Process. 2011;20(1):14–22.
6. Huang W, Song G, Hong H, Xie K. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. IEEE Trans Intell Transp Syst. 2014;15(5):2191–201.
7. Abdel-Zaher AM, Eldeib AM. Breast cancer classification using deep belief networks. Expert Syst Appl. 2016;46:139–44.
8. Du Y, Mordatch I. Implicit generation and generalization in energy-based models. 2019. arXiv preprint arXiv:1903.08689.
9. Tubiana J, Monasson R. Emergence of compositional representations in restricted boltzmann machines. Phys Rev Lett. 2017;118(13).
10. Melko RG, Carleo G, Carrasquilla J, Cirac JI. Restricted boltzmann machines in quantum physics. Nat Phys. 2019;15(9):887–92.

11. Zorzi M, Testolin A, Stoianov IP. Modeling language and cognition with deep unsupervised learning: a tutorial overview. Front Psychol. 2013;4:515.
12. Friston K. The free-energy principle: a unified brain theory? Nat Rev Neurosci. 2010;11:127–38.
13. Testolin A, Zorzi M. Probabilistic models and generative neural networks: Towards an unified framework for modeling normal and impaired neurocognitive functions. Front Comput Neurosci. 2016;10:73.
14. Stoianov I, Zorzi M. Emergence of a visual number sense in hierarchical generative models. Nat Neurosci. 2012;15:194–6.
15. Zorzi M, Testolin A. An emergentist perspective on the origin of number sense. Philosophical Transactions of the Royal Society B: Biological Sciences. 2018;373(1740):20170043.
16. Testolin A, Dolfi S, Rochus M, Zorzi M. Visual sense of number vs. sense of magnitude in humans and machines. Sci Rep. 2020;10(1):1–13.
17. Testolin A, Stoianov I, Zorzi M. Letter perception emerges from unsupervised deep learning and recycling of natural image features. Nat Hum Behav. 2017;1(9):657–64.
18. Sadeghi Z, Testolin A. Learning representation hierarchies by sharing visual features: a computational investigation of persian character recognition with unsupervised deep learning. Cogn Process. 2017;18(3):273–84.
19. Di Bono MG, Zorzi M. Deep generative learning of location-invariant visual word recognition. Front Psychol. 2013;4:635.
20. Grzyb BJ, Nagai Y, Asada M, Cattani A, Floccia C, Cangelosi A. Children's scale errors are a natural consequence of learning to associate objects with actions: A computational model. Dev Sci. 2019;22(4):e12777.
21. Reichert DP, Series P, Storkey AJ. Charles bonnet syndrome: evidence for a generative model in the cortex? PLoS Comput Biol. 2013;9(7).
22. Lee H, Ekanadham C, Ng AY. Sparse deep belief net model for visual area v2. In Adv Neural Inf Process Syst; 2008. p. 873–880.
23. Buesing L, Bill J, Nessler B, Maass W. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. PLoS Comput Biol. 2011;7(11).
24. O'Connor P, Neil D, Liu S-C, Delbruck T, Pfeiffer M. Real-time classification and sensor fusion with a spiking deep belief network. Front Neurosci. 2013;7:178.
25. Pezzulo G, Zorzi M, Corbetta M. The secret life of predictive brains: what's spontaneous activity for? Trends Cogn Sci. 2021;25:730–43.
26. Huttenlocher PR, Dabholkar AS. Regional differences in synaptogenesis in human cerebral cortex. J Comp Neurol. 1997;387:167–78.
27. Castaldi E, Lunghi C, Morrone MC. Neuroplasticity in adult human visual cortex. Neurosci Biobehav Rev. 2020;112:542–52.
28. Fransson P, Skiöld B, Horsch S, Nordell A, Blennow M, Lagercrantz H, øAden U. Resting-state networks in the infant brain. Proc Natl Acad Sci. 2007;104:15531–6.
29. Elman JL, Bates E, Johnson MH. Rethinking innateness: A connectionist perspective on development. MIT Press; 1996.
30. Zorzi M, Testolin A. Computational models of typical and atypical development of reading and numeracy, in The Cambridge Handbook of Dyslexia and Dyscalculia. Cambridge University Press; 2022.
31. Testolin A, Zou WY, McClelland JL. Numerosity discrimination in deep neural networks: Initial competence, developmental refinement and experience statistics. Dev Sci. 2020;e12940.
32. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323(6088):533–6.
33. Hinton G. Training products of experts by minimizing contrastive divergence. Neural Comput. 2002;14:1771–800.
34. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
35. Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: A review. Neural Netw. 2019;113:54–71.
36. McCloskey M, Cohen NJ. Catastrophic interference in connectionist networks: The sequential learning problem. Psychol Learn Motiv. 1989;24:109–65.
37. French RM. Catastrophic forgetting in connectionist networks. Trends Cogn Sci. 1999;3:128–35.
38. Ackley DH, Hinton GE, Sejnowski TJ. A learning algorithm for boltzmann machines. Cogn Sci. 1985;9(1):147–69.
39. Zhang N, Ding S, Zhang J, Xue Y. An overview on restricted boltzmann machines. Neurocomputing. 2018;275:1186–99.
40. Hinton G. Learning multiple layers of representation. Trends Cogn Sci. 2007;11:428–34.
41. Hinton GE. A Practical Guide to Training Restricted Boltzmann Machines. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 599–619.
42. Lamme VA, Roelfsema PR. The distinct modes of vision offered by feedforward and recurrent processing. Trends Neurosci. 2000;23(11):571–9.
43. VanRullen R. The power of the feed-forward sweep. Adv Cogn Psychol. 2007;3(1–2):167.
44. Kreiman G, Serre T. Beyond the feedforward sweep: feedback computations in the visual cortex. Ann N Y Acad Sci. 2020;1464(1):222.
45. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw. 1994;5(2):157–66.
46. Salakhutdinov R, Hinton G. Deep boltzmann machines, in Artificial intelligence and statistics. PMLR; 2009. p. 448–455.
47. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics; 2010. p. 249–256.
48. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15:1929–58.
49. Kumaran D, Hassabis D, McClelland JL. What learning systems do intelligent agents need? Complementary learning systems theory updated. Trends Cogn Sci. 2016;20:512–34.
50. Shin H, Lee J, Kim J, Kim J. Continual learning with deep generative replay. In Adv Neural Inf Process Syst. 2017;30.
51. Cohen G, Afshar S, Tapson J, Van Schaik A. EMNIST: Extending MNIST to handwritten letters. In: International Joint Conference on Neural Networks; 2017. p. 2921–2926.
52. Testolin A, Piccolini M, Suweis S. Deep learning systems as complex networks. J Complex Networks. 2019;8:06.
53. Zambra M, Maritan A, Testolin A. Emergence of network motifs is deep neural networks. Entropy. 2020;22.
54. Barabási AL, Albert R. Emergence of scaling in random networks. Science. 1999;286:509–12.
55. Halberda J, Feigenson L. Developmental change in the acuity of the number sense: The approximate number system in 3-, 4-, 5-, and 6-year-olds and adults. Dev Psychol. 2008;44(5):1457.
56. Piazza M, Facoetti A, Trussardi AN, Berteletti I, Conte S, Lucangeli D, Dehaene S, Zorzi M. Developmental trajectory of number acuity reveals a severe impairment in developmental dyscalculia. Cognition. 2010;116(1):33–41.
57. Calandra R, Raiko T, Deisenroth M, Pouzols FM. Learning deep belief networks from non-stationary streams. In International Conference on Artificial Neural Networks. 2012. p. 379–386.
58. Barabási AL, Pósfai M. Network Science. Cambridge University Press; 2016.
59. Newman ME. The structure and function of complex networks. SIAM Rev. 2003;45(2):167–256.

60. Latora V, Nicosia V, Russo G. Complex Networks: Principles, Methods and Applications. Complex Networks: Principles, Methods and Applications, Cambridge University Press; 2017.