Journal of Complex Networks (2019) **0000**, 1–21
doi: 10.1093/comnet/cnz018

Deep learning systems as complex networks

ALBERTO TESTOLIN
Department of General Psychology, University of Padova, Via Venezia 12, 35131 Padova, Italy
MICHELE PICCOLINI
Department of Physics and Astronomy, University of Padova, Via Marzolo 8, 35128 Padova, Italy
AND
SAMIR SUWEIS
Department of Physics and Astronomy, University of Padova, Via Marzolo 8, 35128 Padova, Italy and Padova Neuroscience Center, University of Padova, Via Giuseppe Orus 2b, Padua, Veneto, Italy
†Corresponding author. Email: samir.suweis@unipd.it

Edited by: Thilo Gross

[Received on 9 October 2018; editorial decision on 15 April 2019; accepted on 11 May 2019]

Thanks to the availability of large scale digital datasets and massive amounts of computational power, deep learning algorithms can learn representations of data by exploiting multiple levels of abstraction. These machine-learning methods have greatly improved the state-of-the-art in many challenging cognitive tasks, such as visual object recognition, speech processing, natural language understanding and automatic translation. In particular, one class of deep learning models, known as *deep belief networks* (DBNs), can discover intricate statistical structure in large datasets in a completely unsupervised fashion, by learning a generative model of the data using Hebbian-like learning mechanisms. Although these self-organizing systems can be conveniently formalized within the framework of statistical mechanics, their internal functioning remains opaque, because their emergent dynamics cannot be solved analytically. In this article, we propose to study DBNs using techniques commonly employed in the study of complex networks, in order to gain some insights into the structural and functional properties of the computational graph resulting from the learning process.

Keywords: networks theory; artificial neural networks; deep belief networks; hierarchical generative models; machine learning; graph analysis.

1. Introduction

Recent strides in artificial intelligence research have opened tremendous opportunities for technological development. In particular, the last decade has been marked by the so-called ‘deep learning revolution’, which is having strong impact both for scientific investigation and for engineering applications [1]. Deep learning allows building artificial neural networks composed of many processing layers, which can learn high-level representations of the data by exploiting multiple levels of abstraction [2]. To differ from conventional machine-learning techniques, this allows to automatically discover intricate statistical structure in large datasets without the need for domain expert knowledge: the relevant features needed to describe the data distribution are learned by the machine from the raw input (e.g. pixels values in a digital image). An intriguing aspect of deep learning systems is that they are inspired by neuronal networks in

biological brains: information processing occurs in a parallel and distributed fashion [3], thereby allowing cognitive abilities to emerge from the orchestrated operation of many simple, non-linear processing units [4, 5].

Deep learning has dramatically improved the state-of-the-art in challenging cognitive tasks, such as image classification [6, 7], speech recognition [8], natural language understanding [9] and even high-level reasoning [10, 11]. It is currently employed by all major IT companies (Google, Facebook, Microsoft, Apple, just to mention a few) to automatically extract knowledge from large digital datasets, and it is achieving impressive performance also in many other domains such as drug discovery [12], genomics [13], high-energy physics [14] and telecommunications [15, 16].

However, despite the continuous progress and the widespread deployment in real-world applications of deep learning, there is still a limited comprehension of its working principles [17]. How does these multilayer networks self-organize to solve a particular task? How is information represented in these systems? Is there a set of fundamental properties underlying the structure and dynamics of deep neural networks?

Some insights into these challenging questions have been gained by inspecting deep learning systems with methods borrowed from neuroscience. For example, response profiles of individual neurons in deep networks often exhibit an impressive match with neurophysiological data [18–21]. Similarly, at the neuronal population level it has been shown that the representational space developed by deep networks has a striking overlap with that observed in the inferior temporal cortex of the primate brain [22, 23]. However, these empirical analyses are somewhat limited in scope because they do not allow to systematically assess structural and functional properties of these complex systems.

We believe that a fresh perspective on these issues can be provided by studying deep learning using the analytical and numerical techniques developed by network science [24, 25], which have already provided very useful in neuroscience research [26–29]. Indeed, in deep learning even knowing perfectly how a single neuron (node) of the network works does not allow to understand how learning occurs, why these systems work so efficiently in many different tasks, and how they avoid getting trapped in configurations that deteriorate computational performance [17, 30, 31]. In these models, interactions play a crucial role during the learning process, therefore a step forward toward a more comprehensive understanding of deep learning systems is their study also in terms of their emerging topological properties [32]. For example, a first characterization of deep networks can be done through several statistical graph properties: connectance, degree distribution, strength distribution and other centrality measures that are now standard in network theory [25]. In particular, we expect that these properties encapsulate relevant information about the learning process of the system. Can we unveil some general relationship between the function (learning outcome) and the structure (topology) of the network? What does depend on the learning task, and what is instead independent of the specific distribution of the input data?

In summary, the aim of the present work is to show that a network science perspective on deep learning may enlighten some of these relationships. We also make available the source code of our software analyses in order to promote the use of the analytical methods we describe.[1]

2. Deep neural networks

Strictly speaking, the main goal of deep learning is to support the creation of 'intelligent' machines that can autonomously learn from experience. Indeed, according to the view promoted by neural network models,

[1] The complete source code can be found at `https://osf.io/sg7rn/`.

perceptual and cognitive phenomena can be conceived as the evolution over time of a complex system of interconnected units that self-organize according to physical principles [4]. Within this framework, the pattern seen in overt behaviour (macroscopic dynamics of the system) emerges from the coordination of subcognitive processes (microscopic dynamics of the system), such as the propagation of activation and inhibition among simple, but non-linear, processing units. The computational properties of neural networks have been investigated since the dawn of artificial intelligence research [33, 34], but only recently the computational power of these systems has been fully unleashed: the major achievement of deep learning algorithms has been to show that artificial neural networks can learn a hierarchy of increasingly more complex concepts, with each concept defined through its relation to simpler concepts (for an historical review, see [35]).

In the following, we will first briefly review the basic theory of learning in deep neural networks, and we will then describe the details of the deep learning model used in our study. For a recent and comprehensive survey about the topic, the reader could refer to [36].

2.1 *Theoretical background*

Artificial neural networks can be formally described using the theory of probabilistic graphical models [37, 38], which provides a general framework to model the stochastic behaviour of a large number of interacting variables. Learning in probabilistic graphical models can be framed within two different settings: in *discriminative* models, the goal is to model conditional distributions over a set of output (target) variables, whose values are specified by explicitly labelling each pattern given as input to the system. This approach is usually referred to as *supervised learning*, because the system is always guided by an external teacher who provides the correct labels. Classification, discrimination and regression problems can be easily framed within this scenario, and are usually solved by applying feed-forward, convolutional deep neural networks trained with error backpropagation (e.g. [7]). In *generative* models, instead, the aim is to capture the joint distribution of all the variables in the system, thereby including also the input variables. This learning modality is usually described as being *unsupervised*, because there are no correct labels that must be associated with each input pattern. The goal is rather to build an internal model of the environment, that is, to discover a set of *latent features* that compactly describe the statistical correlations observed between the variables at play. Clustering, density estimation and dimensionality reduction problems can be framed within this scenario.

In this article we will focus on the latter approach, because generative neural networks can be more naturally characterized using the physical formalism of statistical mechanics, as we will highlight in the following.

2.1.1 *Boltzmann machines*

Generative models can be implemented using different types of probabilistic graphical models. One of them is the Boltzmann machine [39], which is an undirected model (i.e. edges are symmetric, implying a bidirectional flow of information between the nodes) that has been traditionally defined using concepts borrowed from statistical physics. In particular, following the seminal model introduced by Hopfield [40], it can be shown that this type of fully connected, recurrent networks (see Fig. 1A) can develop a point-attractor dynamics, which can be analysed using techniques inspired by the study of pattern formation in physical systems composed by many interacting units. This allows to draw a useful analogy between physical systems with a metastable behaviour and information processing systems that implement content-addressable associative memories: each local energy minima in a metastable physical system can be interpreted as an embodiment of a 'prototype' in an associative memory, where the aim is to store as much information as possible in the form of static configurations of a set of variables.

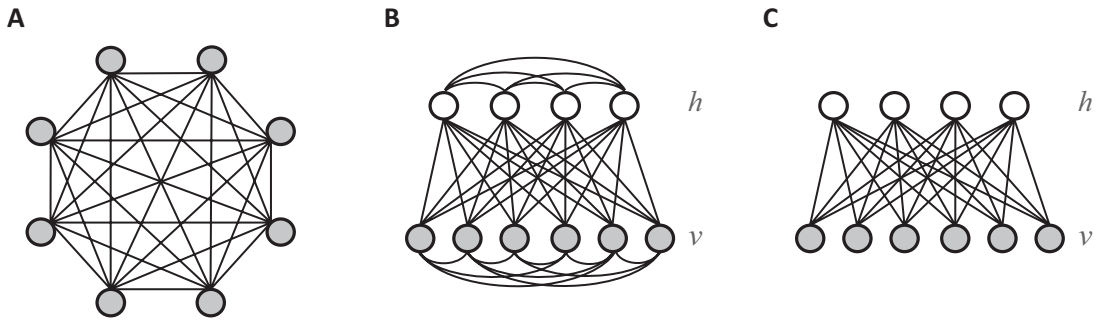

FIG. 1. Graphical representation of several recurrent neural network architectures. (A) Hopfield network, which is a fully connected graph. (B) Boltzmann machine, which is a fully connected graph with two separate sets of units: the visible neurons v are used to perceive input patterns, while the hidden neurons h capture the statistical structure observed in the data. (C) Restricted Boltzmann machine, where intra-layer connections are removed in order to form a bipartite graph.

If each configuration is defined by the actual state of the variables, it is possible to recall previously stored information by giving as input to the network a partially observed state (which would correspond to a specific initial condition acting as content-specific 'search key') and letting the system settle into a stable state, thereby completing the missing values of the remaining variables according to the closest prototype (i.e. by converging to the closest attractor).

Modelling phase transitions in such physical systems has been a longstanding issue in statistical mechanics, which has been explored especially in the context of Ising models. An Ising model is a collection of K binary variables ($\sigma_i = +1, -1$ representing the spins of the atoms in the up (1) or down (-1) state in a ferromagnetic material) arranged into a two-dimensional lattice, which are magnetically coupled to each other. This can be mathematically represented by assigning an *energy function* to the state of the whole lattice $\sigma = \sigma_1, ..., \sigma_K \in \mathbb{R}^K$, given the coupling $J_{mn} = J$ if m and n are neighbours and $J_{mn} = 0$ otherwise, and the external magnetic field H:

$$E(\sigma; J, H) = -\frac{1}{2}\sum_{m,n} J_{mn}\sigma_m\sigma_n - \sum_n H\sigma_n. \tag{1}$$

At equilibrium, spins try to align with the external field and to get parallel each other in order to fulfil the minimum energy principle, that is, minimize $E(J, H)$. The stationary state distribution for the spin system at temperature T is given by the usual Boltzmann distribution:

$$P(\sigma|\beta, J, H) = \frac{1}{Z(\beta, J, H)} e^{-\beta E(\sigma; J, H)}, \tag{2}$$

where $\beta = \frac{1}{k_B T}$ defines the inverse temperature of the system (k_B can be set to 1), and $Z(\beta, J, H) = \sum_\sigma e^{-\beta E(\sigma; J, H)}$ is the *partition function* that assures the normalization of the distribution.

A Boltzmann machine is a generalization of the Ising model, where all units are connected to each other by bidirectional links, that is, in this case the couplings are given by a fully connected matrix W. If we now call x the state of the machine, where each unit i can be off ($x_i = 0$) or on ($x_i = 1$), then we can write the energy gap of the jth unit, defined as the difference between the energy of the whole system

with the *j*th 'off' and its energy with *j*th 'on' by:

$$\Delta E_j = \sum_i W_{ij} x_i, \tag{3}$$

where W is the matrix of synaptic connection weights, which are symmetric (i.e. $w_{ji} = w_{ij}$) and which define the reciprocal interactions between all neurons in the network. If $\Delta E_j < 0$ then the switch off of the *j*th element decreases the total energy $E(x; W, H)$. Therefore, we can minimize the energy by evolving the system over time through stochastic dynamics, where each neuron j changes its local state regardless of its previous state to:

$$x_i = \begin{cases} 1 & \text{with probability } \frac{1}{1+e^{-\beta \Delta E_j}}, \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where the activation energy ΔE_j depends on the overall activation received by unit j from its neighbours. Iteratively updating the state of each unit using this rule, the global system configuration is driven towards thermal equilibrium, that is, towards a state where the energy is locally minimized, thereby following the Boltzmann distribution $P(x|W)$ with energy $E(x; W)$ (as in Eq. (2)). In order to avoid the system being trapped in local minima with relatively high-energy, the overall temperature of the system can be gradually decreased, thereby mimicking the annealing process in physical systems [41].

Perhaps the most interesting property of Boltzmann machines is their ability to 'learn' by modifying the connections strength between units in response to the statistical properties of an external signal that is provided as input to the system. To this aim, the whole network can be partitioned into two distinct functional subsets of units (see Fig. 1B): a set of *visible* neurons v, which are the interface between the network and the external environment and therefore receive the input pattern (the 'observed' data, in the language of graphical models), and a set of *hidden* neurons h, which constitute the internal state of the network and which are used to capture useful 'features' (i.e. statistical correlations) in the input variables.

Intuitively, the objective of learning is to discover a useful set of features, which would serve as latent variables that compactly encode the statistical structure contained in the input data. To this aim, during the learning phase all the network connections are initially set to small, random values and then are gradually adjusted as the network observes new input data (the 'training examples'). The network modifies the strengths of its connections so as to construct an internal generative model that produces examples with the same probability distribution as the examples it is shown. At each learning iteration, all the visible units are clamped into a specific state provided by one training vector, and the hidden units activates according to Eq. 4. This is called the *positive* (or 'wake') phase, because the system is driven by the input data [42]. The visible units are then unclamped, and the network is left free to generate its own visible states by starting from a random state in the hidden units activations. This is called the *negative* (or 'sleep') phase, because the system is driven by its own internal model and tries to generate plausible configurations in the visible layer. Following these two phases, the model parameters (i.e. the connection weights) are updated by maximizing the agreement between the empirical correlations among visible and hidden units resulting from the positive phase and those resulting from the negative phase [39].

Formally, given a set of training examples clamped to the visible neurons $\{v^{(n)}\}_1^N$ we would like to adjust the connection weights W such that the samples generated by the network are well matched by those provided in the training distribution. To this aim, we can define learning as a maximum likelihood problem, where a set of model parameters W has to be adjusted in order to maximize the likelihood of the sampled data. By performing gradient descent on the empirical negative log-likelihood of the training

data, we can analytically derive the equation for updating the connection weights (model parameters) at each learning iteration. Crucially, the derivative of the log-likelihood of a training example with respect to the weight w_{ij} turns out to be surprisingly simple:

$$\frac{\partial \log P(v,h;\beta,J,H)}{\partial w_{ij}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}, \tag{5}$$

where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows, that is, under the empirical data distribution (positive phase) and under the model distribution (negative phase). This leads to a very simple learning rule for updating each connection weight of the network:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}), \tag{6}$$

where ϵ is a small constant representing the learning rate.

2.1.2 *Restricted Boltzmann machines* Although interesting from a theoretical perspective, Boltzmann machines are seldom used in practice due to their extremely high computational complexity. Indeed, these models have an intractable partition function, which prevents the exact computation of the likelihood gradient. This issue can be mitigated using mean-field approximations [43], but computing the model's expectations $\langle v_i h_j \rangle_{\text{model}}$ still remains computationally demanding.

A more effective approach has been instead to constrain the connectivity of the network, moving from a fully connected topology to a bipartite graph (see Fig. 1C). By removing all the intra-layer connections we obtain a Restricted version of a Boltzmann Machine (RBM), where all the neurons in the same layer are conditionally independent given the state of neurons in the opposite layer. This allows to enormously speed-up learning, for example by exploiting efficient implementations of Monte Carlo methods based on parallel Gibbs sampling[44]. For example, when the neurons in one layer are clamped to a particular state (e.g. the visible neurons v are clamped to one training example), the activation probability of all the neurons in the other layer can be efficiently computed in one parallel step:

$$P(h|v) = \prod_i P(h_i|v), \tag{7}$$

where,

$$P(h_i = 1|v) = \frac{1}{1 + e^{-\sum_j w_{ji} v_j}} \tag{8}$$

and $P(v,h|1,J,H) = P(h|v)P(v)$, omitting for simplicity the dependence on the parameters.

2.1.3 *Deep belief networks* A groundbreaking discovery is that RBMs can be used as building blocks to build more complex neural network architectures, where the hidden variables of the generative model are organized into layers of a hierarchy (see Fig. 2). These models are usually referred to as deep belief networks (DBNs) [45, 46]. Such systems are built by stacking together multiple RBMs, which are learned

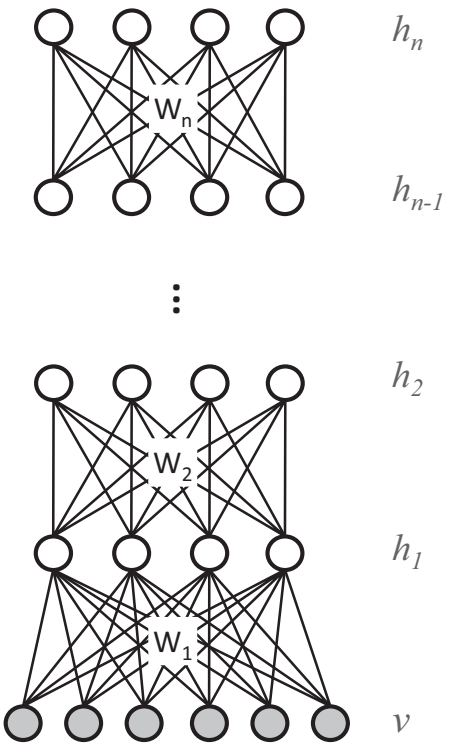

FIG. 2. Graphical representation of a deep belief network built as a stack of restricted Boltzmann machines.

in a layer-wise fashion, that is, the h_n layer is trained after training is completed for the h_{n-1} layer. In this way, the hierarchical generative model is built at separate stages, first starting with simpler features that are kept fixed in order to subsequently learn the more complex ones. After the first RBM has been learned, the activities of its hidden neurons are used as input for a second RBM, with the aim of extracting higher-order correlations from the original data. The main intuition behind these powerful architectures is that, by training a generative layer using as input the hidden causes discovered at the previous layer, the network will progressively build more structured and abstract representations of the input data. Importantly, architectures with multiple processing levels efficiently encode information by exploiting re-use of features among different layers: simple features extracted at lower levels can be successively combined to create more complex features, which will eventually unravel the main causal factors underlying the data distribution [2, 47]. Indeed, it has been shown that functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented by a depth $k-1$ architecture [36]. Moreover, adding a new layer to the architecture increases a lower bound on the log-likelihood of the generative model [45], thus improving the overall representational capacity of the network.

Thanks to its efficiency, the algorithm proposed by [45] solves the problem of learning in densely connected networks that have many hidden layers. Moreover, when implemented on multi-core hardware [e.g. graphical processing units (GPUs)] deep learning is practical even with billions of connections, thereby allowing the development of very large scale simulations [48].

2.2 *Network architecture and learning details*

In our analyses, we considered a popular deep network architecture that has been shown to achieve excellent performance on the MNIST benchmark task of handwritten digit recognition [45, 49]. To assess the robustness of our methods, we also analysed a single-layer RBM trained on a very different type of stimuli, that is, natural image patches [20].

2.2.1 *Training datasets* For the MNIST, learning was performed over a large dataset containing 60000 images of handwritten digits [50]. Each training example contains a digit between 0 and 9, represented as a grey-scale image of size 28×28 pixels. The input to the network was therefore a vectorized matrix of 784 real-valued elements in the range [0,1]. For the natural image patches, learning was performed over a large dataset containing 80000 images of natural scenes [51]. Each training example contains a small portion of a natural landscape, represented as a grey-scale image of size 40×40 pixels. The input to the network was therefore a vectorized matrix of 1600 real-valued elements in the range [0,1].

2.2.2 *Learning parameters* For the MNIST, the DBN was composed of a stack of three RBMs. The visible layer contained 784 neurons, each corresponding to one pixel in the input image. Hidden layers had, respectively, 500, 500 and 2000 neurons, for a total of about 1.6 million connections in the whole DBN. Learning was performed using one-step contrastive divergence, with a fixed learning rate $\epsilon = 0.1$, a momentum coefficient of 0.9 and a weight decay factor of 0.0004. The latter two hyperparameters serve as regularizers for the loss function in order to minimize the risk of overfitting the training data: for additional details, the reader is referred to [52]. For the natural images, the RBM had 1600 neurons in the visible layer and 1000 neurons in the hidden layer. Learning hyperparameters were the same adopted for the MNIST model.

Both models were implemented using MATLAB by exploiting an efficient code tailored for graphic processors [49].[2]

2.2.3 *Training time* Each layer of the DBN was trained for 50 epochs, where each epoch corresponds to a full sweep over all patterns in the training set. We verified learning convergence by monitoring the average reconstruction error after each epoch [52]. After 50 epochs, the reconstruction error did not significantly further improve and most of the neurons already developed structured receptive fields.

3. Network-based analyses

In deep learning systems, the initial processing architecture is fairly generic. For example, in our DBN with k-1 hidden layers, it corresponds to a fully connected (connectivity $C = 1$) k-partite [53] graph with random weights drawn from a Gaussian distribution of mean zero and standard deviation σ, that is, $w_{ij} \sim \mathcal{N}(0, \sigma)$. As a result of learning, complex structural patterns gradually emerge, that *a priori* depend also on the input given for the training algorithm that we have described in detail before.

In the case of input digit images, we are dealing with a multilayer neural network composed by three stacked RBMs. The analysis of the network properties can be performed both on the DBN seen as one aggregate graph, and on each of the three bipartite networks between pairs of layers. By denoting with

[2] The complete source code can be found at http://ccnl.psy.unipd.it/research/deeplearning.

W_a ($a = 1, 2, 3$) the weighted bipartite network between layers $v - h1, h1 - h2, h2 - h3$, respectively (see Fig. 2), we can write the adjacency matrix W of the whole DBN as:

$$W = \begin{pmatrix} \mathbb{0} & W_1 & \mathbb{0} & \mathbb{0} \\ W_1^{Tr} & \mathbb{0} & W_2 & \mathbb{0} \\ \mathbb{0} & W_2^{Tr} & \mathbb{0} & W_3 \\ \mathbb{0} & \mathbb{0} & W_3^{Tr} & \mathbb{0} \end{pmatrix}. \tag{9}$$

From W, we then calculate topological properties of both the intra-layers and the whole network. In particular, we are interested in studying the structure of subnetworks composed by groups of nodes that have characteristic functional properties (given by their receptive fields, see next section). In this way, we try to infer topological signatures of the functional roles of the nodes. We first calculated the distribution of the network degrees, strengths and weights emerging as a result of the learning process in each layer h_i, with $i = 1, 2, 3$. We also calculated the overlap [25] between each pair of layers, to see if group of nodes are less or not activated. We finally computed the average degree, strength, coefficient of variation and average nearest neighbours of the subnetworks formed by nodes with similar functional properties. In all these cases, we used a simplification threshold θ and set to zero all edge weights smaller than θ, in order to obtain a non-weighted graph to work on.

3.1 *Neuronal receptive fields*

It is useful also to introduce the notion of *receptive field* [47], which allows to have a straightforward visualization of how a given neuron 'sees' the input, that is, it incorporates the functional role of that neuron given the input. In other words, the receptive field of a neuron represents the type of visual feature that has been extracted during learning. Since neurons in the visual layer are directly clamped to the input pattern, we can define the concept of receptive field only for neurons that live in upper, hidden layers.

The receptive field for a neuron of the first hidden layer is just the visual representation of the weights of its links toward the neurons in the input layer below. To plot the receptive field of the jth neuron in the layer $h1$, we just need W_1, and extract the weights of the links that start from node j in $h1$ and arrive to all nodes in the layer below (v), that is, the vector $w^1_{i,j}$ for $i = 1, ..., n_1$, where n_1 is the number of nodes in the v-layer while j is fixed. We can then reshape this vector to the original input square matrix form (whose dimension is 28×28 pixels for the case of the MNIST handwritten digits). Each ith pixel value is represented in a grey-scale colour from the minimum (black) value to the maximum (white) value (receptive field samples are shown in Fig. 3). These receptive fields are informative about the neuron's function in the sense that they suggest which features of the input images mostly activate the neuron. For example, a neuron could be particularly sensitive to straight, vertical lines in the peripheral side of the image, while being completely indifferent to analogous straight lines drawn at the bottom of the image (e.g. the first of the four images in Fig. 3). Other neurons could be excited by circularish ring shapes, while being anti-correlated with the space inside this ellipse (see fourth receptive field). Some neurons do not encode localized features, as they assume a distribution of weights that makes the receptive field a fuzzy blurred blob (as in the cases of the second and third images in the figure).

For neurons that lie in the $h2$ and $h3$ layers, we define the receptive field in an indirect way, since there is no direct linkage between the neuron and the visible layer. For a hidden layer 2 neuron (say, l), we first make a weighted average of the receptive fields of the neurons in the hidden layer 1, using the weights $w^2_{j,l}$ leaving from the lth neuron and linking it with the j neurons below. Then, we plot the two-dimensional receptive field of the neuron as $\sum_j w^1_{i,j} w^2_{j,l}$ (l is fixed, while i varies over the nodes in the

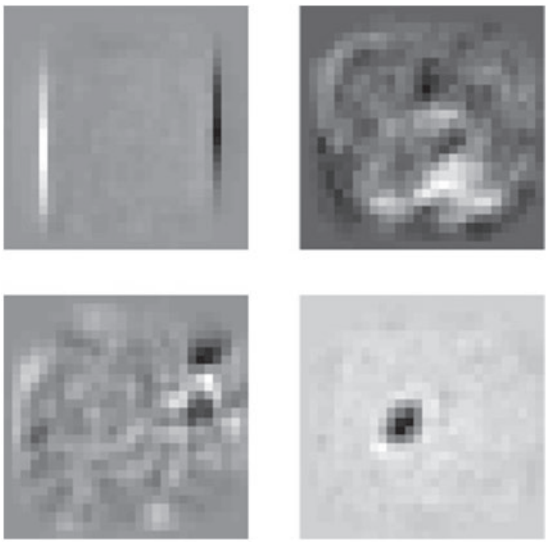

FIG. 3. Example of four different receptive fields emerging in the hidden neurons.

visible layer), obtaining an image of what pixels our lth neuron in the second hidden layer is responsible for. For a neuron m in the third layer we proceed in the same way, by plotting: $\sum_{j,l} w^1_{i,j} w^2_{j,l} w^3_{l,m}$. Results are somehow similar to the receptive fields observed in the first hidden layer, but we can see that at deepest layers the receptive fields become more 'structured' and complex.

3.2 *Clustering of receptive fields*

To infer a possible relation between the structure and the function of the trained deep networks, we need to establish whether neurons with similar 'function' developed similar topological properties. As explained above, we argue that receptive field images of each neuron could be used for such functional characterization.

In order to automatically group neurons with similar functional properties, we thus implemented a hierarchical clustering algorithm based on the earth mover's distance [54]. This distance metric defines the distance between two distributions as the minimal cost that must be paid to transform one distribution into the other, and it is widely used for content-based image retrieval. We adopted this metric because it is based on a solution to a transportation problem from linear optimization, for which efficient algorithms are available, and it guarantees a reasonable precision in measuring the visual similarity of two grey-scale images. Intuitively, the earth mover's distance is computed as follows: every pixel is represented by a certain number of 'pebbles', which is an integer number corresponding to the grey level of that pixel. After normalizing the two images to have the same number of 'pebbles', the distance between them is computed as the minimum cost of matching the pebbles between the two images, which can be formalized and solved as a transportation problem [54]. After computing the distance matrix between all receptive fields, hierarchical clustering was performed by creating a tree structure based on the Euclidean distances between all rows in the matrix. A dendrogram was finally produced by first computing the optimal ordering of the tree leaves using the `optimalleaforder` MATLAB function, and then calling the

`dendrogram` function by setting to 20 the maximum number of leaf nodes. This resulted in a manageable number of receptive field clusters, at the same time limiting the creation of singletons or clusters with only few elements.

4. Results

4.1 *Weights distribution*

By analysing the distribution of the edge weights W of the whole deep network before and after learning, we observed a clear increase of inhibitory (negative) interactions, highlighted by a shift of the weights mean toward negative values. Moreover, after learning the weights distribution is no longer Gaussian (compare panels A and B in Fig. 4), due to the increase of the skewness and kurtosis of the distribution. This effect is mainly due to the change of edge weights in the first $v - h1$ and third $h2 - h3$ bipartite networks (see panels D and F in Fig. 4). Interestingly, the distribution of the weights in the second layer $h1 - h2$ is still symmetric and quasi-normal, with an average close to zero (but still negative). In this case, the departure from a Gaussian distribution is mostly highlighted by an increase of kurtosis, which led to the increase of the distribution tails (see panel E in Fig. 4). A similar result holds for the network trained on natural images. In this case, however, we do not observe a clear shift of the weights distribution toward negative values (the average weight is around zero), but still the distribution becomes markedly skewed, with long tails (see panel C in Fig. 4). These findings show that the training increase the kurtosis of the DBN weights distribution, which in biological neural networks is very high. On the other hand, we found

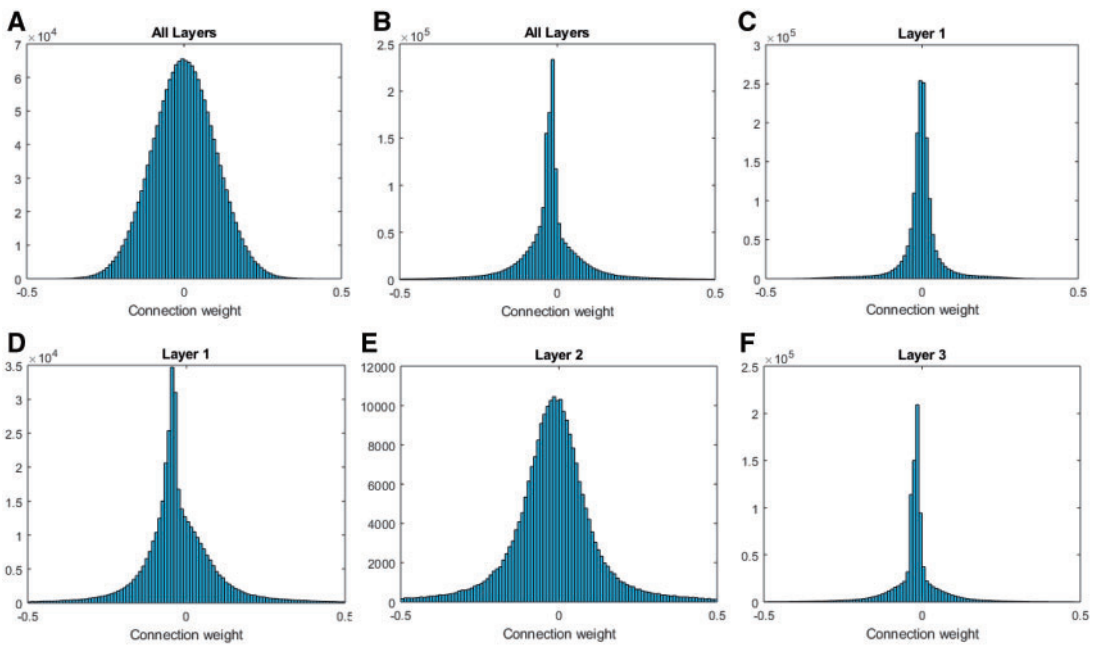

FIG. 4. Distribution of the edge's weights for: (A) the whole network before learning (initialization); (B) the whole network after learning on the MNIST dataset; (C) the network after learning on the natural images dataset; (D) the first layer $v - h1$, (E) second layer $h1 - h2$ and (F) third layer $h2 - h3$ bipartite networks after learning on the MNIST dataset.

that the tail of the trained weights distribution is nevertheless exponential and not log-normal or scale free, as typically observed in real biological networks [55, 56]. Interestingly, according to a recent proposal [57], biologically inspired kurtotic weight distributions could also be exploited to improve robustness of deep learning models to adversarial attacks.

4.2 *Strengths distribution*

We also analysed the strengths distribution of the network before and after learning. The strength for a given node i is given by $s_i = \sum_j W_{ij}$. The distribution of each layer at time $t = 0$ is still Gaussian as the strength is just the sum of the random weights of the links connected to that node, and these are Gaussian distributed at time $t = 0$. As expected from the results on the final weights, after learning the neurons displayed an overall negative strength. However, we also found elongated positive tails, especially in the layer $h1 - v$, $h1 - h2$ (considering links going from $h1$ to $h2$), $h2 - h1$ (links going from $h2$ to $h1$), reaching also very high values (up to $s_i \approx 30, 40$). The strengths of nodes in the $v - h1$ layer were all negative, indicating that nodes in the visible layers on average operated as inhibitors to nodes in the $h1$ layer. In the last hidden layer, the strength distribution displayed a high peak around $s_i \approx 10$. As we will explain later, this is a signature of the strong redundancy that has been found in the last layer's neurons: after learning, many units tend to fall into an almost identical set of features.

4.3 *Receptive fields*

At the end of the learning phase, hidden neurons in the network developed a variety of receptive fields that represent the set of visual features used to efficiently encode the statistical information contained in the training distribution. In particular, neurons in the first hidden layer developed receptive fields tuned to simple, localized spatial structures (such as blobs and small strokes), which were combined by neurons in the deepest layers in order to produce more complex visual features such as edge detectors and digit shapes. Some neurons, especially in the third hidden layer, learned features that were not location specific and covered the whole visual field.

After applying the clustering algorithm to the receptive fields of each hidden layer, we plotted a sample of receptive fields belonging to each cluster in order to verify that neurons encoding similar features were grouped together. As shown in Fig. 5, neurons with a similar functional role were indeed assigned to the same cluster (images in each column represent the receptive fields of the neurons belonging to the cluster identified by the numeric label). A similar result holds for the network trained on natural images (see Fig. 6).

4.4 *Relation between network structure and function*

In order to unveil a potential relation between node topological properties (network structure) and node receptive fields (network function), we first considered the subnetwork formed by nodes belonging to the same functional class $\mathcal{G}_i$, for $i = 1, ..., G$ (with $G = 19$ as described in Section 4.3). We then studied the following topological properties of the nodes for each sub-network:

- The average node degree $\langle k \rangle_j = \sum_{j \in \mathcal{G}_i} k_j / |G_i|$ for j=1..,G, where $|G_i|$ is the number of nodes in the subgraph G_i. We can also divide it into two subgroups: the average positive degree ($\langle k^+ \rangle = \sum_{j \in \mathcal{G}_i} k_j \Theta(k_j) / |G_i|$) and the averaged negative degree ($\langle k^- \rangle = \sum_{j \in \mathcal{G}_i} k_j \Theta(-k_j) / |G_i|$), where $\Theta(x)$ is the Heaviside theta function.

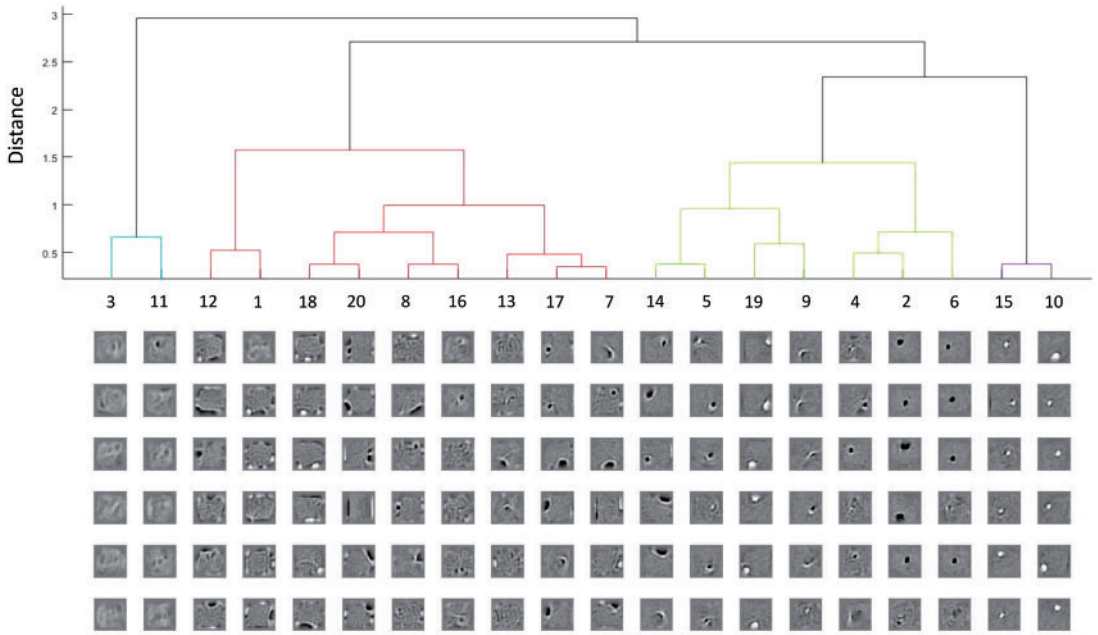

FIG. 5. Hierarchical clustering of the neuronal receptive fields of the first hidden layer emerged from the MNIST handwritten dataset. The tree structure represents the distances between each of the 20 clusters, with smaller values indicating more similar types of receptive fields.

- The average nearest neighbour degree $\langle k_{nn} \rangle = \sum_{i \in \mathcal{G}_i} \sum_{j \in nn(i)} k_j / G$, where $nn(i)$ denotes the nearest neighbours of the node i.
- The average node strength $\langle s \rangle = \sum_{i \in \mathcal{G}_i} s_i / G$.

We have also analysed other properties, such as the standard deviation of the above quantities, the related coefficient of variation, the eccentricity and other centralities measures, but they did not supply any additional relevant information to the overall picture. Figure 7 shows the results of the analysis. We note that we have removed from the analysis few subgraphs that were composed by very few nodes (less than 10), as for such clusters it was not possible to compute an average statistically meaningful behaviour.

Although a clear emergent pattern is missing, some common trends have been found. We can see that in all layers there is not a clear trend in the relation between the averaged degree and the nearest neighbour degree (leftmost column in Fig. 7). Therefore, it seems that the deep learning system does not develop any assortativity pattern following the training process. This holds also when the system is trained using natural images (see leftmost column in Fig. 8).

The relationship between the average node strengths and average node degrees is informative about the relationship between topology and networks weights. In fact, if there is no correlation between the degree of vertices and the weight of edges, then the weights w_{ij} are on average independent of node i and j, and in this case it can be shown [58] that the strength of a vertex is simply proportional to its degree, and thus node degree and strength provide the same information on the system. In our case, we found that the trained neural networks (both on the handwritten digits and the natural images) developed a non-trivial relationship between node degrees and strengths (central column in Figs 7 and 8). In particular, the first

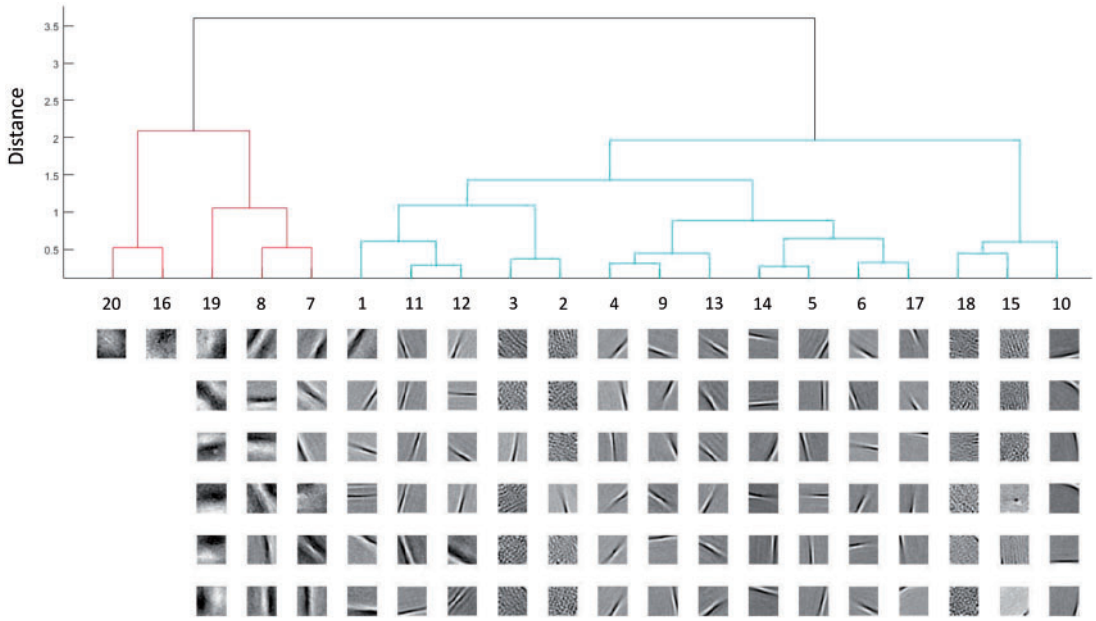

FIG. 6. Hierarchical clustering of the neuronal receptive fields of the first hidden layer emerged from the natural images dataset. The tree structure represents the distances between each of the 20 clusters, with smaller values indicating more similar types of receptive fields.

layer in the DBN displays a positive Pearson correlation ($\rho = 0.56$, p-value = 0.019), while in the second layer we find a negative correlation ($\rho = -0.71$, p-value = 0.001). The network trained on natural images displays a strong, positive correlation ($\rho = 0.87$, p-value <0.0001).

Finally, by analysing the relationship between the positive and negative averaged node degrees in the deep network we found that, depending on the layer, we have different results (see rightmost column in Figs 7 and 8). In the first layer, with the exception of the G_{10} cluster, we have a positive relationship between the two measures, quantified by a correlation of $\rho = 0.59$, p-value = 0.016. On the other hand, in the third layer the opposite is true, and we find a clear negative correlation ($\rho = -0.93$, p-value < 0.0001). In the second layer, no statistical correlation is detected, and there is not significant relation between the two types of degree. The relationship obtained using the neural network trained on natural images (see rightmost panel in Fig. 8) displays the same positive correlation found in the first layer of the deep neural network trained with the handwritten digits dataset ($\rho = 0.95$, p-value <0.0001).

5. Discussion and conclusions

In this research work, we analysed deep learning systems from a network science perspective, by investigating a variety of structural and functional properties of the emergent computational graph. Our analyses allowed to gain interesting insights about the internal functioning of these complex networks, suggesting that the proposed approach might be useful to better understand the principles governing these non-linear, self-organizing systems.

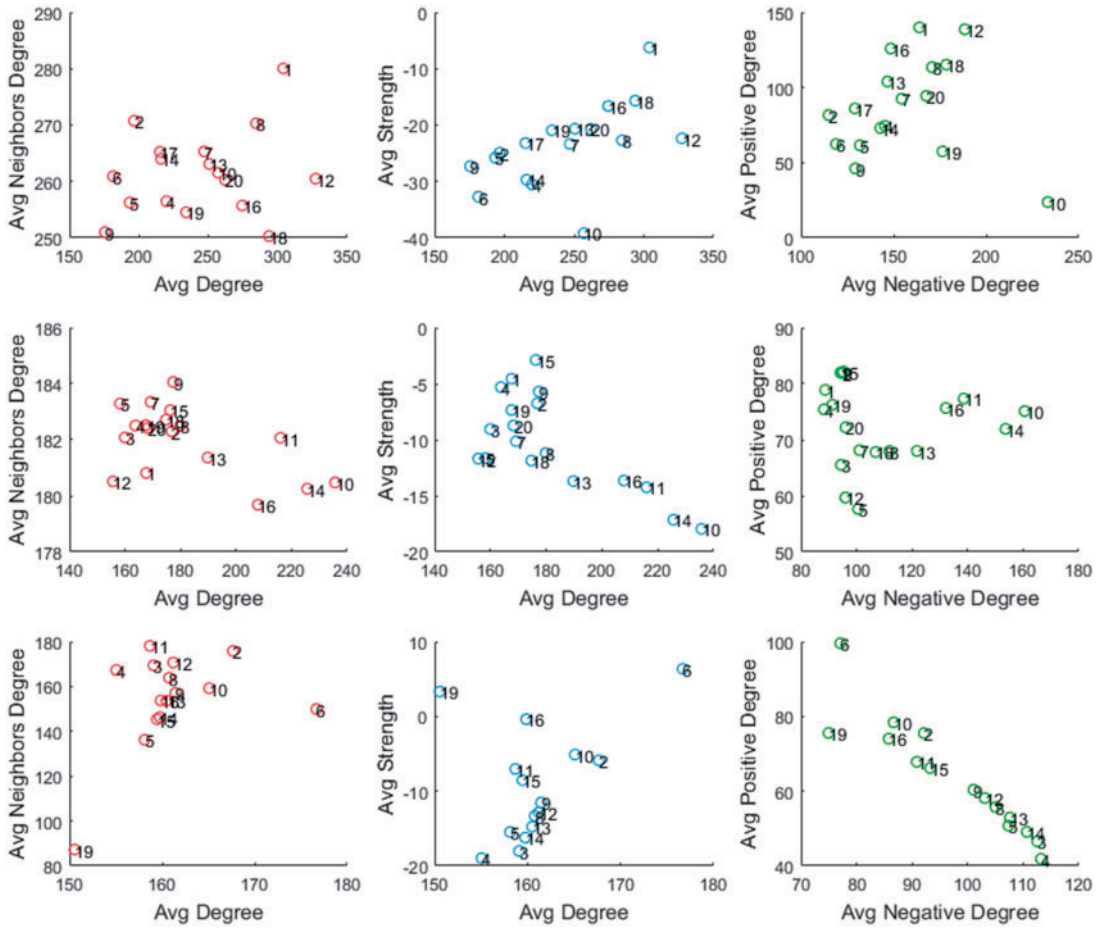

FIG. 7. Structural properties of the sub-graphs G_i of the deep learning system trained with the MNIST dataset, for a simplification threshold of $\theta = 0.1$, where each number denotes a 'functional' group of nodes defined for the similarity of the corresponding receptive fields, as described in the main text. In particular we show the relation between: (*first column*) averaged nearest neighbour degrees and node degree (also known as (diss)assortativity [25]), (*second column*) averaged node strength and degree and (*third column*) averaged positive and negative degrees. Each row corresponds to a different layer of the deep network, ordered from the first (top row) to the third (bottom row).

The ambitious goal of finding structural signatures of the functioning of deep learning systems turned out to be a challenging and delicate point of our analysis. First, it should be noted that in order to perform a non-trivial analysis of the topological properties of the trained networks we need to set a threshold on the connection weights. Otherwise, we would find a trivial 'all nodes connected to all nodes' structure. Of course, setting a threshold is always a delicate operation. Our rationale has been to set a threshold that enabled to remove all the weaker links, while preserving the most relevant ones. We have performed a sensibility analysis with other threshold values ($\theta = 0.01, 0.03, 0.05, 0.2$), and the main results presented here are robust with respect to the choice of different threshold values.

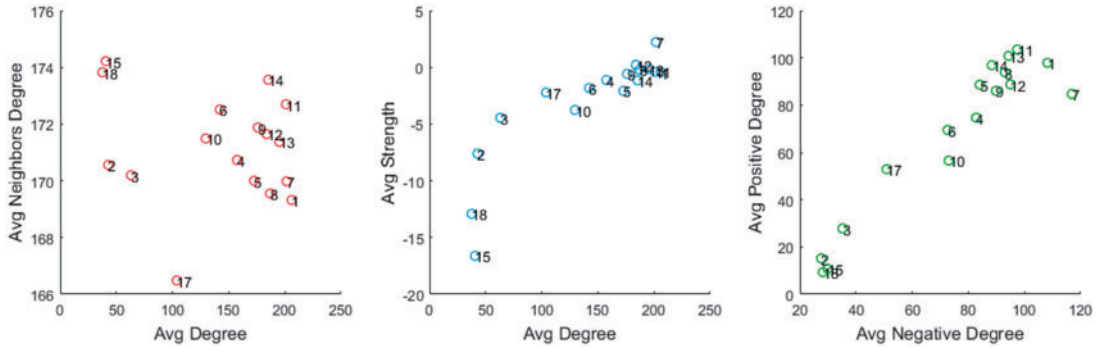

FIG. 8. Structural properties of the sub-graphs G_i of the deep learning system trained the natural images dataset, for a simplification threshold of $\theta = 0.1$, where each number denotes a functional group of nodes defined for the similarity of the corresponding receptive fields, as described in the main text. In particular, we show the relation between: ((*first column*) averaged nearest neighbour degrees and node degree, ((*second column*) averaged node strength and degree and (*third column*) averaged positive and negative degrees.

The emergence of inhibitory links in the network trained with the handwritten digits dataset, especially in the first and third layers, induces strong anti-correlations between neurons in the visible and top layer with those in the second one. This effect may be promoted by the type of input (i.e. the pixels distribution in handwritten digits), as this clear shift of the edge weights toward negative values disappears when the system is trained on natural images. For example, the MNIST dataset contains images of white digits written on a uniform, black background: the marked contrast between elements in the image may have induced strong anti-correlations among neurons. Indeed, we observe that many neurons settle their weights to negative values, thereby inhibiting the activity of several connected neurons. Looking at the receptive fields of these neurons, we observe that they tend to specialize in describing very localized features, thereby activating in response to very particular features in the stimulus (e.g. spots or straight edges), while they are anti-correlated with all the remaining pixels in the input image.

Another relevant information can be drawn from the strength distributions of the neurons in the third layer, where a high peak of neurons with negative strength (between $s_i \in [-8, -10]$) is observed. The corresponding receptive fields are shown in Fig. 9, which highlight a high level of redundancy in the neuron's function. In particular, all the nodes with a strength in the range of the peak have exactly the same receptive field, and are usually known as *dead units* [59]. This redundancy may be due to the use of too many neurons in the third hidden layer, which might not be all necessary to improve the representational capability of the network. This finding suggests that emerging topological properties might be used—at runtime, or once learning is completed—in order to adjust the model architecture to make it more compact and efficient (e.g. by removing dead units). Such method might be used in synergy with similar approaches that aim at reducing the number of connection weights in deep neural networks. For example, pruning algorithms have proved very effective for compressing large-scale models that need to be deployed on embedded systems with limited hardware resources [60], and sparse initialization schemes based on network-level properties can lead to a quadratic reduction in the number of connections with no decrease in accuracy [61].

A further non-trivial point is how to define the functional modules of a deep neural network. There is not an *a priori* definition about the function of the neurons in a deep learning system, as the overall activity of the network nodes is very complex and cannot be classified, for example, by a binary variable (e.g. inhibitory/excitatory neuron). Here, we have proposed to approximately characterize the neuron function

FIG. 9. Some receptive fields of the third hidden layer, highlighting the presence of many dead units.

by visualizing its receptive field. The receptive field is a complex, non-trivial emerging description of the overall activity of each neuron in a given layer. Therefore, it is a highly abstract representation of the neuron function and it might not always have a clear interpretation. However, we have proposed that functionally similar neurons can be detected by clustering the corresponding receptive fields, thereby allowing to define functional sub-networks. The analysis of the topological properties and weighted architectures of these subgraphs hardly gives clear explanations on how the neural network works.

In conclusion, in this work we have proposed a network science perspective to unveil topological and functional properties of deep learning systems. Although the relation between structure and function remains only outlined, it is a first step to go *beyond the black-box use* of such learning systems. In particular, our work highlights how some topological properties (i.e. emergence of inhibitory links) depend on the type of input signals, and thus might be initialized in a non-random way that is closer to the configuration observed after training. Other properties, such as system redundancy, do not depend on the input distribution, but might instead depend on the architecture of the system itself (i.e. number of neurons in each layer or the inclusion of sparsity constraints). An interesting future perspective will be to relate these results with the recently proposed hypothesis of criticality in deep learning [62]. A related issue would be to characterize the stability of learning in deep neural networks with respect to random attacks and link failure [25, 63, 64]: how many edges can we delete before the learning system will stop working?

We believe that our investigation represents a small step toward the challenging goal of developing analytical techniques for interpreting and understanding deep learning systems [65, 66]. Even though

deriving analytical descriptions of such complex, self-organizing systems might seem daunting, it has been recognized as one of the most fundamental issues to be solved in the near future [67]. Indeed, these powerful AI systems are already operating in our societies, and international regulatory agencies are pressing scientists and engineers to ensure that AI systems will produce human-understandable explanations of their automated decisions [68].

Acknowledgements

A.T. and S.S. gratefully acknowledges the support of NVIDIA Corporation with the donation of a Titan Xp GPU used for this research.

Funding

The STARS grant (DEEPMATH and ReACT, respectively) from the University of Padova to A.T. and S.S.

References

1. LeCun, Y., Bengio, Y. & Hinton, G. E. (2015) Deep learning. *Nature*, **521**, 436–444.
2. Hinton, G. E. (2007) Learning multiple layers of representation. *Trends Cogn. Sci.*, **11**, 428–434.
3. Rumelhart, D. E. & McClelland, J. L. (eds) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, vol. 1. Cambridge, MA: MIT Press.
4. McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S. & Smith, L. B. (2010) Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends Cogn. Sci.*, **14**, 348–356.
5. Testolin, A. & Zorzi, M. (2016) Probabilistic models and generative neural networks: towards an unified framework for modeling normal and impaired neurocognitive functions. *Front. Comput. Neurosci.*, **10**, doi:10.3389/fncom.2016.00073.
6. He, K., Zhang, X., Ren, S. & Sun, J. (2016) Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ, USA: IEEE, pp. 770–778.
7. Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012) ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.*, **24**, 609–616.
8. Mohamed, A., Dahl, G. E. & Hinton, G. E. (2012) Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.*, **20**, 14–22.
9. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011) Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, **12**, 2493–2537.
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. (2015) Human-level control through deep reinforcement learning. *Nature*, **518**, 529–533.
11. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**, 484–489.
12. Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E. & Svetnik, V. (2015) Deep neural nets as a method for quantitative structure-activity relationships. *J. Chem. Inf. Model.* , **55**, 263–274.
13. Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K. C., Hua, Y., Gueroussov, S., Najafabadi, H. S., Hughes, T. R., Morris, Q., Barash, Y., Krainer, A. R., Jojic, N., Scherer, S. W., Blencowe, B. J. & Frey, B. J. (2015) The human splicing code reveals new insights into the genetic determinants of disease. *Science*, **347**, pp. 144–154.

14. BALDI, P., SADOWSKI, P. & WHITESON, D. (2014) Searching for exotic particles in high-energy physics with deep learning. *Nat. Commun.*, **5**, doi:10.1038/ncomms5308.

15. TESTOLIN, A., ZANFORLIN, M., DE GRAZIA, M. D. F., MUNARETTO, D., ZANELLA, A., ZORZI, M. & ZORZI, M. (2014) A machine learning approach to QoE-based video admission control and resource allocation in wireless systems. *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*. Piscataway, NJ, USA: IEEE, pp. 31–38.

16. ZORZI, M., ZANELLA, A., TESTOLIN, A., DE FILIPPO DE GRAZIA, M. & ZORZI, M. (2015) Cognition-based networks: a new perspective on network optimization using learning and distributed intelligence. *IEEE Access*, **3**, 1512 – 1530.

17. BALDASSI, C., BORGS, C., CHAYES, J. T., INGROSSO, A., LUCIBELLO, C., SAGLIETTI, L. & ZECCHINA, R. (2016) Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proc. Natl. Acad. Sci. USA*, **113**, E7655–E7662.

18. LEE, H., EKANADHAM, C. & NG, A. Y. (2008) Sparse deep belief net models for visual area V2. *Adv. Neural Inf. Process. Syst.*, **20**, 873–880.

19. TESTOLIN, A., DE FILIPPO DE GRAZIA, M. & ZORZI, M. (2017) The role of architectural and learning constraints in neural network models: a case study on visual space coding. *Front. Comput. Neurosci.*, **11**, 1–17.

20. TESTOLIN, A., STOIANOV, I. & ZORZI, M. (2017) Letter perception emerges from unsupervised deep learning and recycling of natural image features. *Nat. Hum. Behav.*, **1**, 657–664.

21. ZORZI, M. & TESTOLIN, A. (2018) An emergentist perspective on the origin of number sense. *Philos. Trans. R. Soc. B Biol. Sci*., 373:20170043.

22. GÜÇLÜ, U. & VAN GERVEN, M. A. J. (2015) Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *J. Neurosci.*, **35**, 10005–10014.

23. KRIEGESKORTE, N. (2015) Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annu. Rev. Vis. Sci.*, **1**, 417–446.

24. ALBERT, R. & BARABÁSI, A.-L. (2002) Statistical mechanics of complex networks. *Rev. Mod. Phys.*, **74**, 47–97.

25. NEWMAN, M. (2010) *Networks: An Introduction*. Oxford, UK: Oxford University Press.

26. BRESSLER, S. L. & MENON, V. (2010) Large-scale brain networks in cognition: emerging methods and principles. *Trends Cogn. Sci.*, **14**, 277–290.

27. BULLMORE, E. & SPORNS, O. (2009) Complex brain networks: graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.*, **10**, 186–198.

28. MEDAGLIA, J. D., LYNALL, M.-E. & BASSETT, D. S. (2015) Cognitive network neuroscience. *J. Cogn. Neurosci.*, **27**, 1471–1491.

29. PARK, H.-J. & FRISTON, K. J. (2013) Structural and functional brain networks: from connections to cognition. *Science*, **342**, 1238411–1238411.

30. LIN, H. W., TEGMARK, M. & ROLNICK, D. (2017) Why does deep and cheap learning work so well? *J. Stat. Phys.*, **168**, 1223–1247.

31. MHASKAR, H., LIAO, Q. & POGGIO, T. A. (2017) When and why are deep networks better than shallow ones? *31st AAAI Conference on Artificial Intelligence*. Menlo Park, CA, USA: AAAI Press, pp. 2343–2349.

32. AGLIARI, E., BARRA, A., GALLUZZI, A., GUERRA, F., TANTARI, D. & TAVANI, F. (2015) Topological properties of hierarchical networks. *Phys. Rev. E*, **91**, 062807.

33. MCCULLOCH, W. S. & PITTS, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, **5**, 115–133.

34. ROSENBLATT, F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, **65**, 386.

35. SCHMIDHUBER, J. (2015) Deep learning in neural networks: an overview. *Neural Netw.*, **61**, 85–117.

36. GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. (2016) *Deep Learning*. Cambridge, MA, USA: MIT Press. http://www.deeplearningbook.org.

37. JORDAN, M. I. & SEJNOWSKI, T. J. (eds) (2001) *Graphical Models: Foundations of Neural Computation*, vol. 5. Cambridge, MA: MIT Press.

38. Koller, D. & Friedman, N. (2009) *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: The MIT Press.

39. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985) A learning algorithm for Boltzmann machines. *Cogn. Sci.*, **9**, 147–169.

40. Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.*, **79**, 2554–2558.

41. Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, **220**, 671–680.

42. Hinton, G. E., Dayan, P., Frey, B. J. & Neal, R. M. (1995) The wake-sleep algorithm for unsupervised neural networks. *Science*, **268**, 1158.

43. Welling, M. & Hinton, G. E. (2002) A new learning algorithm for mean field Boltzmann machines. *International Conference on Artificial Neural Networks* (J. R. Dorronsoro ed.). London: Springer, pp. 351–357.

44. Hinton, G. E. (2002) Training products of experts by minimizing contrastive divergence. *Neural Comput.*, **14**, pp. 1771–1800.

45. Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006) A fast learning algorithm for deep belief nets. *Neural Comput.*, **18**, 1527–1554.

46. Salakhutdinov, R. (2015) Learning deep generative models. *Annu. Rev. Stat. Appl.*, **2**, 361–385.

47. Zorzi, M., Testolin, A. & Stoianov, I. P. (2013) Modeling language and cognition with deep unsupervised learning: a tutorial overview. *Front. Psychol.*, **4**, doi:10.3389/fpsyg.2013.00515.

48. Raina, R., Madhavan, A. & Ng, A. Y. (2009) Large-scale deep unsupervised learning using graphics processors. *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, pp. 873–880.

49. Testolin, A., Stoianov, I., De Grazia, M. D. F. & Zorzi, M. (2013) Deep unsupervised learning on a desktop PC: a primer for cognitive scientists. *Front. Psychol.*, **4**, doi:10.3389/fpsyg.2013.00251.

50. LeCun, Y. (1998) The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/.

51. Snavely, N., Seitz, S. M. & Szeliski, R. (2006) Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics (TOG)*, **3**, 835–846.

52. Hinton, G. E. (2012) A practical guide to training restricted Boltzmann machines. *Neural Networks: Tricks of the Trade* (G. Montavon, G. B. Orr & K. Müller eds). Berlin, Heidelberg: Springer, pp. 599–619.

53. Chartrand, G. & Zhang, P. (2008) *Chromatic Graph Theory*. Boca Raton, FL, USA: CRC Press.

54. Rubner, Y., Tomasi, C. & Guibas, L. J. (2000) The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.*, **40**, 99–121.

55. Buzsáki, G. & Mizuseki, K. (2014) The log-dynamic brain: how skewed distributions affect network operations. *Nat. Rev. Neurosci.*, **15**, 264.

56. Yasumatsu, N., Matsuzaki, M., Miyazaki, T., Noguchi, J. & Kasai, H. (2008) Principles of long-term dynamics of dendritic spines. *J. Neurosci.*, **28**, 13592–13608.

57. Nayebi, A. & Ganguli, S. (2017) Biologically inspired protection of deep networks from adversarial attacks. https://arxiv.org/abs/1703.09202.

58. Barrat, A., Barthelemy, M. & Vespignani, A. (2007) The architecture of complex weighted networks: measurements and models. *Large Scale Structure And Dynamics Of Complex Networks: From Information Technology to Finance and Natural Science* (G. Caldarelli & A. Vespignani eds). River Edge, NJ, USA: World Scientific, pp. 67–92.

59. Fritzke, B. (1995) A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, pp. 625–632.

60. Han, S., Mao, H. & Dally, W. J. (2015) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. https://arxiv.org/abs/1510.00149.

61. Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M. & Liotta, A. (2018) Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nat. Commun.*, **9**, 2383.

62. Song, J., Marsili, M. & Jo, J. (2017) Emergence and relevance of criticality in deep learning. *Resolution and Relevance Trade-offs in Deep Learning*, https://arxiv.org/abs/1710.11324.
63. Cohen, R. & Havlin, S. (2010) *Complex Networks: Structure, Robustness and Function*. Cambridge, UK: Cambridge University Press.
64. Kurant, M., Thiran, P. & Hagmann, P. (2007) Error and attack tolerance of layered complex networks. *Phys. Rev. E*, **76**, 026103.
65. Lipton, Z. C. (2016) The mythos of model interpretability. *ICML Workshop on Human Interpretability in Machine Learning*. (B. Kim, D. M. Malioutov & K. R. Varshney eds). ArXiv: https://arxiv.org/abs/1607.02531.
66. Samek, W., Wiegand, T. & Müller, K.-R. (2017) Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. https://arxiv.org/abs/1708.08296.
67. Shneiderman, B. (2016) Opinion: the dangers of faulty, biased, or malicious algorithms requires independent oversight. *Proc. Natl. Acad. Sci.*, **113**, 13538–13540.
68. Goodman, B. & Flaxman, S. (2016) European Union regulations on algorithmic decision-making and a right to explanation. *ICML Workshop on Human Interpretability in Machine Learning*. (B. Kim, D. M. Malioutov & K. R. Varshney eds). ArXiv: https://arxiv.org/abs/1607.02531.