UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Modeling cognition with generative neural networks: The case of orthographic processing

University of Padova

Department of Developmental Psychology and Socialization

Ph. D. School in Psychological Sciences - XXVII Cycle

Cognitive Science Program

January 2015

Candidate:	Alberto Testolin
Supervisor:	Ch.mo Prof. Marco Zorzi
School Director:	Ch.ma Prof. Francesca Peressotti

Abstract

This thesis investigates the potential of generative neural networks to model cognitive processes. In contrast to many popular connectionist models, the computational framework adopted in this research work emphasizes the generative nature of cognition, suggesting that one of the primary goals of cognitive systems is to learn an internal model of the surrounding environment that can be used to infer causes and make predictions about the upcoming sensory information. In particular, we consider a powerful class of recurrent neural networks that learn probabilistic generative models from experience in a completely unsupervised way, by extracting high-order statistical structure from a set of observed variables. Notably, this type of networks can be conveniently formalized within the more general framework of probabilistic graphical models, which provides a unified language to describe both neural networks and structured Bayesian models. Moreover, recent advances allow to extend basic network architectures to build more powerful systems, which exploit multiple processing stages to perform learning and inference over hierarchical models, or which exploit delayed recurrent connections to process sequential information. We argue that these advanced network architectures constitute a promising alternative to the more traditional, feed-forward, supervised neural networks, because they more neatly capture the functional and structural organization of cortical circuits, providing a principled way to combine top-down, high-level contextual information with bottom-up, sensory evidence. We provide empirical support justifying the use of these models by studying how efficient implementations of hierarchical and temporal generative networks can extract information from large datasets containing thousands of patterns. In particular, we perform computational simulations of recognition of handwritten and printed characters belonging to different writing scripts, which are successively combined spatially or temporally in order to build more complex orthographic units such as those constituting English words.

Acknowledgements

When studying adaptive systems, one realizes that their astonishing complexity largely derives from their intimate coupling with the surrounding environment, which often makes it difficult to draw a clear distinction between the cognitive entity and its "external" world. My thoughts and ideas have been constantly shaped by so many external forces that I feel indebted to hundreds of people, who enriched my life with great insights and sources of inspiration. It would be impossible to list all of them here, but I feel very lucky that I've had such a stimulating, warm environment always feeding my mind. Here I will try to mention those who have been mostly responsible for the exciting path that allowed me to produce this modest research work. The first acknowledgement goes to my mother Luigina, my father Giuseppe and my sister Chiara, who always encouraged me to explore and critically analyze the world around me, and who always strongly supports all my disparate enterprises. Many thanks also go to the uncountable number of friends who accompany my life with happiness, and with whom I shared my greatest joys and my saddest moments. They are my driving force, and they constantly fill my brain with unforgettable memories. I would also like to express my gratitude to the many teachers and professors that contributed to all stages of my formal education. Special thanks go to my supervisor, Marco Zorzi, who always supported my research work with all the available resources, and who has been a great guide, helping me to navigate the turbulent ocean of cognitive neuroscience. I would also like to acknowledge my machine learning professor, Alessandro Sperduti, who introduced me to the fascinating world of artificial neural networks and self-organizing systems, which have been the starting point of my postgraduate studies. A special mention also goes to all my lab mates, who provided me the best environment to explore my ideas in a friendly atmosphere. In particular, the boundless enthusiasm and optimism of Ivo Stoianov has been a great source of inspiration, and the valuable advices of my colleagues and friends Simone Cutini

and Mario Bonato have been very useful to improve my understanding of the challenging world of scientific research. Warm thanks also go to Jay McClelland, who kindly welcomed me in his lab for a wonderful research period, always encouraging me to explore my own ideas and their implications. He is a great example of open mindedness and intellectual honesty, and our fruitful conversations contributed to further boost my interest in cognitive science and to consolidate my respect for academic institutions. A special mention also goes to all the PDP lab crew, in particular to my lighthouse Michael Henry Tessler, and to all the friends of the Stanford Italian Society, who made my stay in California such a memorable time.

Contents

1 Introduction

The computational approach is one of the cornerstones of modern science, and since its appearance its impact on the scientific method has been tremendous. The advent of digital computers created immense, new possibilities to explore and test scientific theories, even leading to novel epistemological foundations for the scientific enterprise (e.g., Winsberg, 2009). At present, computational modeling is a common practice not only in the more formal disciplines from which it derived (e.g., physics, chemistry, mechanics), but also in disciplines that were traditionally considered to lie far away from computational principles, such as psychology, sociology and linguistics. By framing a theory in computational terms, scientists are forced to adopt a precise, formal language, because all the details of the theory should be explicitly stated in order to simulate it on a computer. In turn, the use of mathematical principles to describe a theory facilitates communication between researchers: ideas are formulated using a common, abstract language that reduces ambiguities and misunderstandings, and that should eventually make it easier to quantitatively compare different hypotheses.

Nowadays, the need for computational theories in the "brain sciences" is particularly pressing (Abbott, 2008). Experimental research keeps producing a huge amount of data and facts, but we still lack a comprehensive theory that can make sense of it. In particular, methodological advances in experimental psychology and the modern techniques adopted in neuroscience research are providing an increasing amount of detailed information about brain and cognitive processes at many different levels of analysis, spanning from genetic, molecular, cellular and physiological processes to the more high-level phenomena investigated in psychophysics, neuroimaging and behavioral studies. However, from a theoretical standpoint we do not yet have a clear, comprehensive explanation about how psychological phenomena might emerge

from the underlying neural substrate (Bassett & Gazzaniga, 2011), and computational simulation is considered an irreplaceable tool to bridge these different levels of description (Churchland & Sejnowski, 1992; Gerstner, Sprekeler, & Deco, 2012). The relevance of computational modeling in this scenario is also highlighted by the consistent amount of resources that are being invested to boost computational research in cognitive neuroscience. Two examples of this trend are represented by the European Human Brain Project (Markram, 2012) and its U.S. counterpart, the BRAIN initiative (Insel, Landis, & Collins, 2013), which both have the ambitious goal of building a large-scale simulation of the human brain.

At the same time, and especially in the above-mentioned brain sciences, the notion of computational model is often opaque, and sometimes it is implicitly used with reference to particular, restricted meanings. Interestingly, the etymology of the term "computation" derives from the Latin *com-putare*, which literally means "considering together". The original meaning of the term is therefore extremely general, and should not be restricted to the most popular acceptations. This issue is particularly relevant in the field of cognitive science, where the notion of computational model is often interpreted according to the idea of symbolic processing proposed by cognitivism. In this thesis we will be concerned with a much broader meaning of computation, which finds its roots in the cybernetic movement and that later became popular with the appearance of the ***parallel distributed processing*** approach. In order to better grasp this distinction, it is useful to briefly review the main theoretical ideas that have been introduced since the first attempts to understand the brain from a computational perspective.

1.1 Historical overview

Interestingly, the first attempts to characterize cognitive systems using the language of computation came from scientists interested in artificial intelligence research. According to some historical reconstructions (Dupuy, 2009), the whole

cognitive science enterprise should be considered a consequence of the cybernetic movement, which was initiated right after the Second World War by some of the greatest minds of the twentieth century such as Warren McCulloch, John von Neumann, Walter Pitts, Claude Shannon, Alan Turing, and Norbert Wiener, just to name a few. The founders of cybernetics believed that nervous systems – and, more generally, biological systems – could be described as a very general type of machines, and that they could therefore be studied using the laws of physics and mechanics. The so-called "Macy Conferences", which took place from 1946 to 1953 in the U.S., have been the starting place where these scientists coming from many different disciplines, such as mathematics, logic, engineering, physics, physiology, biology, psychology, anthropology, and economics, meet together to build a general science of how the human mind works. One of the characterizing aspects of these meetings was their interdisciplinary nature, which highlighted the broad scope and the ambitious goal of the enterprise and the necessity to draw ideas from many different perspectives and traditions. At the same time, this heterogeneity also led to the creation of different visions among the cyberneticians, which resulted in the emergence of different research paradigms. In particular, two main positions appeared to impose themselves and divide the cybernetic community.

The first one, which lately became the prevailing paradigm in cognitive science, was also one of the major driving forces behind the following research that more explicitly focused on artificial intelligence. According to this paradigm, the mind can be characterized as a symbolic processor, which exploits structured data representations to manipulate incoming information (inputs) and produce proper responses (outputs). This type of computation is based on rule-based constructs, like those implemented in the syntax of the newly-invented programming languages of the era (McCarthy, 1959). Not surprisingly, among the defenders of this approach there were John von Neumann, one of the inventors of the modern digital computer, and Warren McCulloch, one of the pioneers proposing how logic-based behavior could be implemented in neural

circuits (McCulloch & Pitts, 1943). The symbolic paradigm provided the foundation for cognitivism (Chomsky, 1957), and soon received widespread support from what will be called the "strong artificial intelligence" community (Minsky, 1961; Newell & Simon, 1961). One of the distinguishing features of cognitivism is therefore the ***computer metaphor***, which claims that cognition is based on internal, mental representations analogous to data structures that are serially manipulated according to computational procedures analogous to algorithms. Indeed, within cognitivism the most common way to represent the flow of information processing in the brain is based on flow-charts, as is typically done for computer programs. Importantly, according to this view the mind can be abstracted from the underlying physical medium, just as the software level in digital computers is independent from the hardware architecture over which it is implemented. Cognitivism had a huge impact on the development of cognitive science, and it allowed to create important theories about the way we think about the mind (Anderson, 1983; Pinker, 1999). Some authors went even further and suggested to literally conceive the mind as a special type of computer that implements a powerful "language of thought" (Fodor, 1975; Pylyshyn, 1984).

The other major paradigm produced by the cybernetic movement proposed a substantially different interpretation of the mind in mechanistic terms. Its theoretical roots were not primarily grounded in logic and algorithmic theory, but instead derived from physics and the theory of dynamical systems. Among its stronger advocates there was Norbert Weiner, one of the founders of control theory and the theory of adaptive systems (Wiener, 1948). According to this view, the computational nature of nervous systems (and, more generally, of many types of complex systems) should not be characterized in terms of symbolic processing, but should instead be described as a form of dynamical evolution of a system over the space of its possible configurations. Information processing in a system thus corresponds to the dynamic change of its own internal organization according to its current internal state and, eventually, to perturbations coming from the external environment (principle of "self-

regulation"). Importantly, this definition of computation introduces the notion of *feedback* as a critical feature of cognitive systems, because their behavior is intrinsically coupled with the environment with which they interact through circular, causal relationships (Ashby, 1956; Maturana & Varela, 1980; Von Foerster, 1984; Wiener, 1948). According to this approach, cognitive processes cannot be considered separately from the physical medium that implements them, because they intrinsically emerge from it. This computational paradigm led to the introduction of the ***network metaphor***, which claims that cognition should be conceived as a collection of emergent dynamics rising in networks of interconnected units that collectively self-organize according to physical principles. Even if many authors studied the self-organizing behavior of artificial neural networks in the decades following the cybernetic movement (Grossberg, 1976; Hopfield, 1982; Kohonen, 1982), these ideas became popular only after the spread of connectionism and the Parallel Distributed Processing (PDP) approach (Rumelhart & McClelland, 1986b). Many different reasons have been proposed to explain the slower acceptance of the network metaphor by the scientific community compared to the computer metaphor (e.g., McClelland, 2009). However, it is interesting to note that scientists tend to formulate descriptions of the human brain using analogies from the most powerful technologies available at the time (Daugman, 2001); for example, brains were often compared to windmills or to clockwork mechanisms during the 18th century, but these metaphors were gradually replaced by the steam engine and hydraulic machines at the beginning of the 20th century. It is therefore not surprising that the computer metaphor imposed itself during the half of the 20th century, and that scientists are switching to a network conception only nowadays, in the Internet era (Graham & Rockmore, 2011).

The computational characterization of the brain provided by connectionist and dynamical systems models is significantly different from the more popular one based on the computer metaphor. In particular, a distinguishing concept considered by these models is that of *emergence* (McClelland, 2010). Emergent

properties can be defined as properties that are not found in any component of the system but are still features of the system as a whole. More precisely, emergence is often described as the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems (Goldstein, 1999). Examples of emergent phenomena abound in nature: in physics, transitions between solid, liquid and gaseous states are interpreted as emergent phenomena, as the formation of convection cells and whirlpools; chemical reactions can also be described as dynamics emerging from the interactions between molecules, and the same framework can be applied to describe more sophisticated biological structures like cell membranes and cell assemblies, up to very complex organizations such as ant colonies, swarms and economic markets (Morowitz, 2002). In cognitive science, emergentist approaches entail that the structure seen in overt behavior (macroscopic dynamics of the system) reflects the operation of subcognitive processes (microscopic dynamics of the system), such as the propagation of activation and inhibition among interconnected neurons and the adjustment of their connection strength (Beer, 2000; McClelland et al., 2010). Cognition can therefore be described as the evolution of a system composed by many interacting units that are connected together. Due to the presence of non-linear interactions, the global behavior of the system can result in complex dynamics, which are usually studied by simulating the evolution of the system starting from an initial configuration and moving through a sequence of states.

As we will discuss in Chapter 2, cognitive models based on artificial neural networks usually share the fundamental principles of the PDP framework, such as the idea that knowledge underlying cognitive activity is stored in the connections among neurons, and that learning processes adaptively change the strength of these connections in order to improve the global behavior of the system according to past experience. However, there are other distinguishing features of the PDP approach that are often ignored in neural network research and that could have a great impact on the resulting models. In the following

section, we will briefly discuss some critical principles that were initially emphasized within the PDP approach, but which have not been seriously taken into account in many subsequent models.

As a matter of terminology, it should be mentioned that most of the times PDP models are said to be "connectionist". Even if there is no explicit consensus about the use of these terms, we consider them as being comparable in this thesis due to the widespread use of the term connectionism among the cognitive science community.

1.2 Some neglected aspects of connectionism

As we discussed before, a fundamental principle included in the first cybernetic models was the notion of feedback. In the neural networks literature, the term feedback is sometimes associated with that of ***recurrence***, because they both indicate the presence of some form of circular causation inside the system. The relevance of feedback loops in self-organizing systems was one of the major concerns of cyberneticians, because according to their view the processing flow in nervous systems cannot be conceived as being unidirectional: causation seems to occur both upwards and downwards between multiple levels of the system, creating a complementary or mutually constraining environment between cognitive processes and their underlying neural dynamics. In other words, emergent phenomena can feedback to lower levels from which they generated, causing lower level changes through what is called "downward causation". Within this assumption, emergence of mental properties cannot be understood using fundamental reductionism (Bassett & Gazzaniga, 2011). Notably, this argument is also central in other modern disciplines, such as epigenetics and system biology, where the traditional, bottom-up notion of "genetic program" is being replaced by a more integrated view that takes into account the causal role of high-order structures in the organization of lower levels, leading to an intricate interplay between different levels of biological organization (Noble, 2006). Despite the existence of powerful recurrent neural

network models (Ackley, Hinton, & Sejnowski, 1985; Grossberg, 1976; Hopfield, 1982; Williams & Zipser, 1989), the most commonly adopted architectures by cognitive modelers are *feed-forward*; that is, processing is directed from the input layer to the output layer forming a bottom-up flow. Even if their mathematical treatment appears to be more complicated, here we argue that recurrent networks have a series of advantages compared to feed-forward networks and that they should be considered as a primary choice to model cognitive processes within the PDP framework.

Another important point is concerned with the nature of learning algorithms used to adapt the connection weights in neural network models. Indeed, the majority of connectionist models rely on a supervised training regimen, which is usually implemented using the popular backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986). Although backpropagation constitutes a powerful way to efficiently train feed-forward neural networks, supervised learning alone might be insufficient to fully address the type of learning mechanisms implemented in the brain. In particular, the assumptions that learning is largely discriminative (e.g., classification or function learning) and that an external teaching signal is always available at each learning event (that is, all training data is labeled) appear to be implausible from a cognitive perspective. Reinforcement learning (Sutton & Barto, 1998) is a valuable alternative, but there is a broad range of situations where learning seems to be fully unsupervised and its only objective is that of building rich internal representations of the environment (Hinton & Sejnowski, 1999), for example by learning a generative model that tries to explain the latent factors underlying sensory information. Here we therefore argue that ***generative learning*** constitute a promising alternative to feed-forward, discriminative learning, especially considering the fact that it can be efficiently implemented in some general classes of recurrent neural networks (Dayan, Hinton, Neal, & Zemel, 1995; Hinton & Ghahramani, 1997; Hinton, 2002).

A related aspect refers to another architectural principle commonly associated with feed-forward neural networks. Due to limitations of the backpropagation algorithm (Bengio, Simard, & Frasconi, 1994), these type of networks usually exploit only few processing layers. However, as also suggested by the hierarchical organization of many areas of the cerebral cortex (Felleman & Van Essen, 1991), ***hierarchical architectures*** composed by many processing layers are likely to improve the computational efficiency of nervous systems. Indeed, a recent breakthrough in neural computation research has been the introduction of powerful, hierarchical generative neural networks known as deep learning systems (Bengio, 2009; Hinton & Salakhutdinov, 2006). Here we argue that a promising avenue for cognitive modeling would therefore be to investigate artificial neural networks composed by many processing layers, like those implementing hierarchical generative models.

During the last decades, another important achievement in the formal definition of artificial neural networks has been their characterization within a ***probabilistic framework*** (Bishop, 1995; Jordan & Sejnowski, 2001; McClelland, 2013; Neal, 1992). At the same time, recent developments in the theory and practice of probabilistic graphical models (Koller & Friedman, 2009) paved the way to integrate powerful analytical techniques (such as those derived from Bayesian statistics) into the neural computing framework. We therefore also argue that an important step forward to improve current connectionist models would be to replace the commonly used deterministic models with stochastic neural networks (e.g., Ackley et al., 1985).

As a final point, we also believe that time has come to seriously consider using alternative types of digital computers when performing PDP simulations. While symbolic processing models are particularly suitable for being implemented and simulated using standard digital computers, neural network models exhibit an intrinsic parallelism that requires the use of ***parallel computing architectures*** to properly simulate them. In other words, the intimate coupling between structure and function in PDP models suggests that

we must resort to a radically different form of computing equipment in order to study them experimentally. In turn, though small-scale, "toy" models can provide important theoretical insights, the use of parallel computing architectures allows to significantly scale-up connectionist models, for example by simulating large-scale neural networks composed by thousands, or even millions, of neurons. Here we propose to exploit recent advances in parallel computing architectures introduced by the CUDA framework (Nickolls, Buck, Garland, & Skadron, 2008) to perform computational simulations using low-cost graphic processing units (GPUs). This promising hardware equipment allows to greatly reduce computational times required to train large-scale deep learning systems (Krizhevsky, Sutskever, & Hinton, 2012; Raina, Madhavan, & Ng, 2009), at the same time guaranteeing simplicity and usability of the source code (Testolin, Stoianov, De Filippo De Grazia, & Zorzi, 2013).

It is important to reiterate that all the above-mentioned principles do not constitute novel ideas *per se*, because they have been traditionally incorporated in comprehensive PDP theories of cognition (e.g., Churchland & Sejnowski, 1992; Rumelhart & McClelland, 1986). However, due to theoretical and/or practical issues they did not became standard practice in connectionist modeling, and they gradually became overshadowed by the use of simpler types of artificial neural networks. Recent advances in neural computation research now allows overcoming many of the original difficulties, but the highly technical knowledge required to master these improvements tends to keep them confined inside the machine learning community. We believe that these exciting ideas should be spread and adopted also by cognitive scientists, and hopefully one of the contributions of this thesis will be to make them accessible to a wider audience.

1.3 Why orthographic processing?

Since the first appearance of cognitive models inspired by the PDP framework, orthographic processing has been a very influential domain to investigate theoretical hypotheses by means of computational simulations

(McClelland & Rumelhart, 1981; Rumelhart, 1977). The appeal of orthographic processing as a modeling domain is due to the fact that it incorporates many sophisticated aspects of cognitive processes, at the same time providing a simplified, controlled setting where cognitive theories can be evaluated more accurately.

Orthographic processing requires an efficient, multilevel integration of fine-grained visual features, which must be structurally combined and interpreted according to detailed knowledge about letters, graphemes, morphemes and words. Furthermore, written symbols are often represented using a variety of different shapes and styles, thereby significantly increasing the complexity faced by pattern recognition mechanisms. Finally, the central role of contextual information in orthographic processing suggests that bottom-up information should heavily interact with high-level, top-down information in order to provide useful constraints that disambiguate sensory data. Despite the involvement of these sophisticated forms of processing, recognition of written symbols and words represents a much more circumscribed task compared to the general tasks faced by cognitive systems in real-life environments.

Furthermore, during the last decades cognitive psychologists investigated many aspects of orthographic processing using a wide variety of experimental techniques, which include detailed electrophysiological, neuroimaging and behavioral measures (Grainger, 2008). Orthographic processing therefore appears to be a valuable prototypical domain to investigate many fundamental perceptual and cognitive phenomena, and to validate the adequacy of computational models according to experimental evidence.

1.4 Outline of the thesis

This research work is organized into four main parts. In Chapter 2 we review and discuss some fundamental ideas behind parallel distributed processing and their relationship with current theories of cortical computation. The aim is to provide a comprehensive view of the major points of interest, at

the same time highlighting open challenges and possible ways to address them. We then focus on the computational characterization of the ventral stream of the visual system, with particular attention to its role in orthographic processing. In Chapter 3 we formally present the framework of probabilistic graphical models, which provides a clean formalism to describe the models considered in this thesis. In particular, we focus on generative models and their implementation in stochastic, recurrent neural networks, by also discussing the most popular inference and learning algorithms. We then focus on recent advances in machine learning research, which allow to effectively train large-scale, hierarchical generative models in an unsupervised fashion, and we propose several techniques that can be used to model cognitive processed using this powerful class of models. We also discuss some possible extensions of generative neural networks to sequence learning, which allow including the temporal dimension in cognitive models. In Chapter 4 we present and discuss the experiments and the computational simulations performed to validate the proposed modeling framework. For each simulation, we provide details about the datasets used, the model formulation and the methodological procedure adopted to evaluate it. In the first set of simulations, we empirically assess the performance of an efficient implementation of deep belief networks based on graphic processors. In the following simulations, we exploit such implementation to investigate cognitive processes underlying the recognition of printed Latin letters. We then test the ability of the model to learn other types of scripts by studying the recognition of Farsi handwritten characters, and we also test its capability to transfer a set of learned visual features across different writing systems. Finally, we test the performance of generative neural networks when processing multiple letters, which can be arranged either in a spatial or in a temporal structure. Finally, in Chapter 5 we conclude the thesis by discussing the results. We also provide links to other important research areas, and propose critical future research directions that should be pursued to further test – and possibly improve – the computational framework described in this thesis.

2 Theoretical Background

2.1 Parallel Distributed Processing (PDP)

In this section we briefly review and discuss some critical features of the PDP framework and their relation to current theories of cortical processing. The interested reader could refer to several reference textbooks to have a more comprehensive treatment of the cognitive (Rumelhart & McClelland, 1986b), neurobiological (Churchland & Sejnowski, 1992) and mathematical (Bishop, 1995; Hertz, Krogh, & Palmer, 1991) aspects of the subject.

2.1.1 Artificial neurons

One of the central tenets in PDP models is to consider the neuron as the basic processing unit in nervous systems. This implies that PDP models are not too much concerned about complex dynamics emerging within single cells, but instead assume a simplified representation of neurons in order to focus the computational characterization at the "network level". The first information processing models of artificial neurons date back to the cybernetic age, when researchers already recognized the relevance of plasticity in learning processes to adaptively change the behavior of a system according to past experience (Hebb, 1949; McCulloch & Pitts, 1943; Rosenblatt, 1958; Widrow & Hoff, 1960). Artificial neurons (Fig. 1A) are characterized by a set of weighted input connections (representing synapses located along the dendrites of the neuron), an activity level (representing the state of polarization of the neuron) and an output value (representing the average firing rate of the neuron, that is, the mean electrical activity propagated along the axon in a fixed time interval). A single neuron can therefore be conceived as a simple information processing device, which collects inputs (encoded as activation values coming from other neurons) and computes an output value, which can in turn be transmitted to other neurons. Inputs can be represented as *n*-dimensional vectors, where each

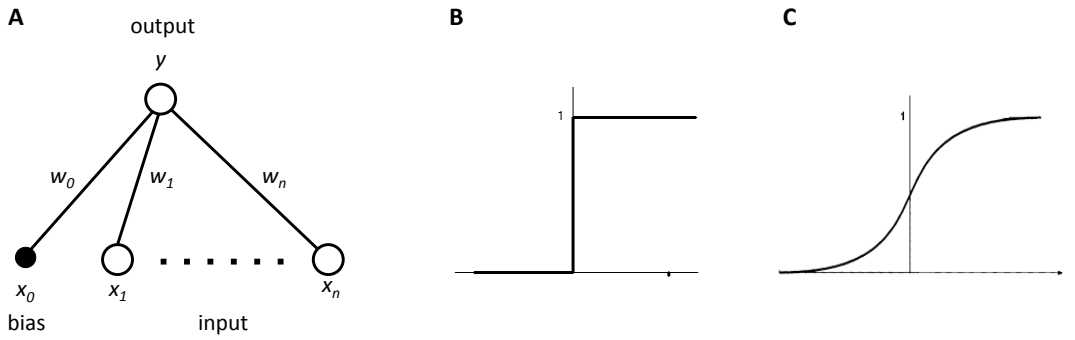

Figure 1. **(A)** Artificial neuron scheme. **(B)** Heaviside step function (hard threshold). **(C)** Logistic function (graded threshold).

element corresponds to an incoming activation value, thereby allowing to define the neuron as a multivariate mapping function that associates a certain scalar value to each input vector. The activity level of a neuron is usually computed as a weighted sum of its input signals, which can be compactly expressed using vector notation:

$$y(x) = \sum_i w_i x_i + w_0 = \boldsymbol{w}^T \boldsymbol{x} + w_0 \qquad (1)$$

where vectors are represented using bold letters and T denotes the transpose operator. The vector $\boldsymbol{x}$ represents the input pattern, the vector $\boldsymbol{w}$ represents the synaptic weights and the scalar w_0 represents the bias of the neuron, which can be thought as a default, internal resting level of the neuron. The output value transmitted through the axon is usually computed by applying some type of activation function σ to the activity level resulting from Eq. 1:

$$out(x) = \sigma(y(x)) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0) \qquad (2)$$

In the simplest case, σ is implemented as a linear function of its input. However, more powerful models can be obtained using non-linear, monotonic activation functions. Early influential models (McCulloch & Pitts, 1943; Rosenblatt, 1958)

implemented σ using a discrete threshold function, such as the Heaviside step function (Fig. 1B). Another possibility is to use a continuous, sigmoid activation function, such as the logistic function (Fig. 1C), which approximates a discrete step function as the slope parameter increases:

$$\sigma\big(y(x)\big) = \frac{1}{1+\mathrm{e}^{-y(x)}} \qquad (3)$$

Importantly, continuous activation functions such as that of Eq. 3 can be differentiated, which is a critical prerequisite for creating more complex network architectures and learning algorithms, as we will see below.

From a mathematical standpoint, this family of artificial neurons implements a simple form of linear discrimination, which assigns one of two possible classes to data points lying on an *n*-dimensional vector space, where *n* represents the dimensionality of the input pattern. Linear discriminant functions can be easily extended to multi-class problems by simply adding more neurons to the network: each output neuron corresponds to a different class, and the classification is performed by selecting the class corresponding to the neuron with the highest output value. Linear discriminant functions, also known as *linear classifiers*, have a straightforward interpretation in geometric terms. If we consider a two-dimensional space for simplicity, a linear classifier corresponds to a straight line that separates the data points into two distinct regions of the plan (as shown in Fig. 2). In the general case of *n* dimensions, the decision boundary becomes a hyperplane (i.e., it corresponds to a subspace of one dimension less than its ambient space). The weight vector $\boldsymbol{w}$ of the classifier defines the orientation of the hyperplane, while the bias w_0 defines the position of the hyperplane in terms of its perpendicular distance from the origin.

Interestingly, artificial neurons that use a logistic sigmoid activation function also have an interesting probabilistic interpretation. In particular, in this case the output values can be interpreted as posterior probabilities, and the

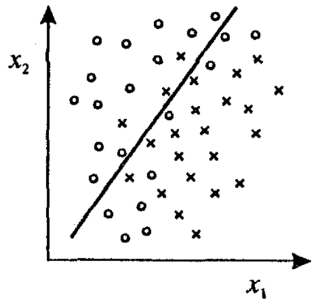

Figure 2. Geometrical interpretation of a linear discriminant function in a two-dimensional space, where data points belong to two classes identified by circles and crosses.

computation performed by the neuron corresponds to a form of logistic discrimination (Bishop, 1995).

2.1.2 From neurons to networks

Though appealing from an analytical perspective, linear classifiers have some important drawbacks that limit their use in complex scenarios. In particular, this type of models can perform accurate mappings only when data points are linearly separable (Minsky & Papert, 1969). As we will discuss in Chapter 3, this simplified behavior can often be useful to perform some interesting analyses on internal representations extracted by deep learning models. However, from a cognitive modeling perspective artificial networks composed by a single-layer of linear classifiers have limited applicability. More interesting models can be obtained by considering more powerful types of architectures, which perform successive transformations of the input signal and/or allow for bidirectional interactions among subsequent processing layers.

One of the most popular types of artificial neural networks is represented by the *multi-layer perceptron*, which extends the basic, single-layer network discussed above by adding several layers of adaptive weights (see Fig. 3C).

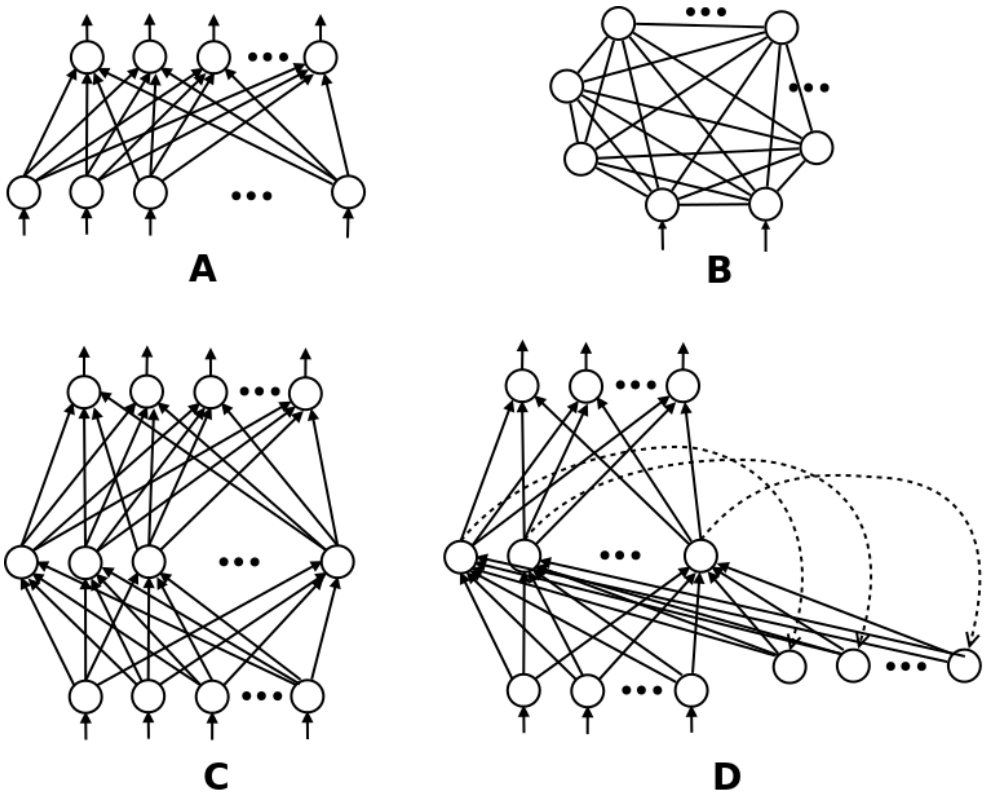

Figure 3. **(A)** Single-layer linear associator. **(B)** Fully recurrent, bi-directional network. **(C)** Multi-layer, feed-forward network. **(D)** Partially recurrent network with delayed connections.

The processing flow in multi-layer perceptrons is a generalization of that of single-layer networks: once the output units of the first layer have been computed using Eq. 2, their activation values are used as input for a subsequent network, until the top-layer activations are produced. The analytical function corresponding to a two-layer network (such as the one represented in Fig. 3C) can therefore be defined as:

$$y(x) = \tilde{\sigma}(\boldsymbol{w_2}^T \sigma(\boldsymbol{w_1}^T \boldsymbol{x})) \qquad (4)$$

The term included in the inner parentheses represents the sum of the input activations $\boldsymbol{x}$ weighted according to the first-layer connections $\boldsymbol{w_1}$, which is then transformed according to the activation function σ and taken as input by the

second layer in order to produce the final output according to the second-layer connections $\boldsymbol{w_2}$ (for notational simplicity, bias terms have been omitted). Intermediate units that are not treated as final outputs are called *hidden units*, and their activation function σ might not necessarily correspond to $\tilde{\sigma}$, which is that used for the output units. However, it is important to note that if the activation function of all hidden units is taken to be linear, then for any such network we can always find an equivalent network without hidden units. This follows from the fact that the composition of successive linear transformations is itself a linear transformation. More expressive mappings are obtained using hidden units with logistic activation, which allows to create internal, non-linear representations of the input vectors that can support more general transformations of the incoming signal.

Multi-layer perceptrons have very powerful representational capabilities: in fact, it has been demonstrated that networks with just two layers of weights are capable of approximating any continuous functional mapping (Hornik, Stinchcombe, & White, 1989). Moreover, for networks with differentiable activation functions there exists a computationally efficient method, called *error backpropagation*, for finding the derivatives of an error function with respect to the connection weights, which constitutes the basis for a variety of learning algorithms. However, the analytical properties of multi-layer perceptrons are derived under the fundamental assumption that the network diagram must be *feed-forward*, that is, it cannot contain feedback loops. This ensures that the output values can be calculated as explicit, deterministic functions of the inputs and the weights, and so the whole network represents a multivariate, non-linear functional mapping.

Other classes of artificial neural networks allow for the presence of feedback connections among different processing layers (see Fig. 3B and 3D). These architectures are usually called *recurrent neural networks* and, despite their more challenging analytical treatment, they also have some unique and appealing characteristics.

One of the milestones in the study of recurrent networks has been their characterization using the theory of statistical physics: in particular, it has been shown that recurrent networks with symmetric weights can develop a point-attractor dynamics (Hopfield, 1982), which can be modeled 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* behavior and information processing systems that implements content-addressable *associative memories*. The notion of metastability is used to characterize isolated systems that spend an extended time in a configuration other than the system's state of lowest energy (global minimum), that is, the energy configuration landscape is characterized by a substantial number of locally stable states (local minima), as shown in Fig. 4A. A prototypical example of metastable systems is constituted by spin glasses, where complex internal structures spontaneously arise through local interactions between the magnetic spins of the component atoms. This phenomenon is usually unwarranted, because it prevents the system to reach a uniform, ferromagnetic configuration where all the spins are aligned, causing instead the formation of frustrated interactions, which are geometrical distortions in the structure of atomic bonds. However, this richness of possible internal states has a very useful application to address the problem of representing and storing information in neural networks. Indeed, we can interpret each local energy minima 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. If each configuration is represented by a vector, then we can recall previously stored information by giving as input to the network a partial vector (which would correspond to a content-specific “search key”) and letting the system settle into a stable state, thereby completing the missing values in the vector according to the closest prototype.

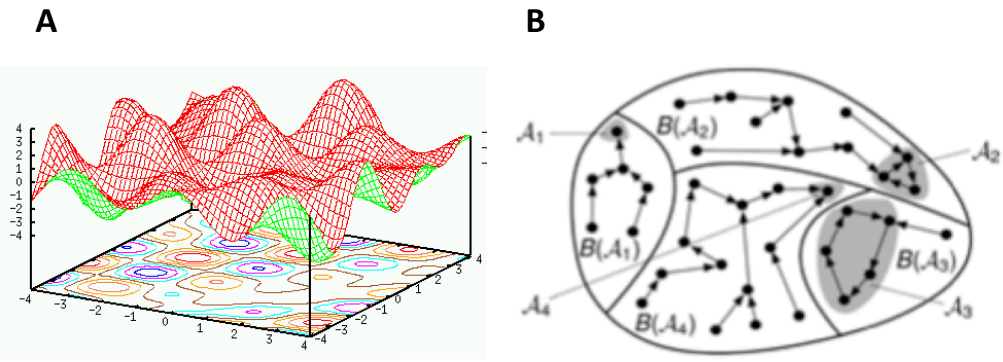

Figure 4. **(A)** A metastable energy landscape with many locally stable configurations. **(B)** 2D projection of the corresponding attractor-point dynamics, where each basin of attraction is bounded by a continuous line.

Interestingly, this process can also be described as a form of constraint optimization, where local interactions among neurons in the network specify a set of soft-constraints that must be satisfied to reach a stable, locally optimal configuration. The presence of too many local optima can be detrimental because it prevents the system to settle into robust, stable configurations, and a great deal of research work indeed focuses on implementing computationally efficient ways to reach better (if not global) optimal states. Also here, the analogy with physical systems allows to greatly improve the analytical treatment of the problem, for example by providing inspiration for powerful optimization techniques like simulated annealing (Geman & Geman, 1984; Kirkpatrick, Gelatt, & Vecchi, 1983). The interpretation of neural information processing in terms of state phase transitions through an attractor landscape has been successively refined and expanded by introducing more powerful, stochastic neural network models. In particular, as we discuss at length in Chapter 3, the Boltzmann machine (Ackley et al., 1985) represents a flexible and expressive recurrent architecture that allows building powerful learning systems such as those used in the present research work.

Another interesting class of recurrent neural networks includes systems where feedback interactions are not implemented using intrinsically

bidirectional connections, but are instead mediated by an additional set of variables, as shown in Fig. 3D. These networks have been shown particularly useful when modeling temporal phenomena, because they can exploit a set of temporally delayed recurrent connections to propagate information through time (Elman, 1990). In Chapter 3, we discuss a recently proposed model that exploits temporally delayed connections within a bidirectional network, thereby combining the strengths of both the recurrent approaches.

2.1.3 Learning algorithms

Up to this point, we introduced the main formalism behind popular types of artificial neural network models and we briefly discussed their representational capabilities. However, one of the most interesting feature of these systems is their ability to learn from experience, that is, to adapt their behavior according to a set of stimuli received from the environment. The literature on this topic is vast and multifaceted, and the discipline of machine learning is nowadays a well-established field within the artificial intelligence community. Here we only review and discuss some general common themes, which will be useful to better characterize the models used in our experimental simulations.

From a broad perspective, we can define a learning algorithm as a set of procedures that allows to improve the performance of a system based on experience. In general, performance is measured according to some type of error function, and experience is represented by a set of examples (training patterns) that are repeatedly shown to the learning system. During learning, the system is expected to extract and encode as much information as possible from the training patterns, with the aim of successively obtain high performance also on a novel set of patterns, which are usually contained in a separate test dataset. This implies that the problem of learning is not simply concerned with finding a good representation for a fixed set of patterns, but also to generalize the acquired knowledge to novel, previously unseen examples. In contrast with most of the

symbolic processing approaches, on which learning entails the extraction of a set of rule-based, propositional constructs, learning in artificial neural networks can be cast within the mathematical framework of *statistical learning* theory (Jain, Duin, & Mao, 2000), which deals with the problem of finding predictive functions based on available data. It is hard to provide a clean taxonomy of machine learning techniques, but a coarse-grained classification usually distinguishes between supervised, semi-supervised, unsupervised, and reinforcement learning settings.

In *supervised learning*, the aim of the system is to extract some form of functional mapping between a set of input patterns and a set of corresponding output patterns. Classification, discrimination and regression problems can be easily framed within this scenario, and feed-forward networks constitute the most studied architecture to learn supervised tasks. Common approaches to solve these problems consist in defining an appropriate, differentiable error function that measures the discrepancy between the desired, target output and the output predicted by the system, and then perform some form of gradient descent in order to find how each weight should be changed in order to obtain more accurate mappings. Thanks to its computational efficiency, the backpropagation algorithm (Rumelhart et al., 1986) is probably the most popular example of this approach. However, one of the major limits of supervised learning is that it always requires an external "teacher" to provide labeled information, that is, each input pattern must be associated with a corresponding output pattern. Apart for the need to manually label huge training datasets, this training regimen does not usually occur in real-life situations and therefore appears to be quite implausible from a cognitive modeling perspective. Even if asking to perform supervised tasks is a useful way to assess the knowledge level of a cognitive system, this approach must therefore be complemented with other learning schemes in order to better model the range of learning processes observed in biological organisms.

In *unsupervised learning*, the system is not concerned with learning an appropriate output response for a given input pattern. Instead, the main goal is to extract a useful set of features from the training data, which allow to accurately represent the input information and to support similarity judgments among different patterns. Clustering, density estimation, feature extraction and dimensionality reduction tasks can be framed within this scenario, and many different types of neural network architectures can be used to solve these problems. One successful approach exploited by a variety of neural networks consists in extracting statistical features from the input data by trying to build an *internal model* of the environment, that is, to encode the probability distribution observed in the training dataset using a set of internal, latent variables. In this setting, the objective of learning corresponds to find a set of parameters that allow to accurately reconstruct the input patterns, usually by means of maximum likelihood estimation methods. Unsupervised learning is therefore not concerned with finding mapping functions, but instead with discovering the hidden structure contained in the input signal. As we will discuss later, once the hidden causes that generated the input patterns have been made explicit, supervised mappings might be more easily established. This combination of unsupervised and supervised methods is also usually exploited in *semi-supervised learning* scenarios, where the goal is to find a mapping function but not all the training patterns have an associated label.

It should be noted that the distinction between supervised and unsupervised settings is not always sharp, and it might became blurred in certain situations. For example, supervised, feed-forward networks can be trained to perform a reconstruction of their own input, that is, the target output vector contains an exact copy of the input vector. This type of architecture is called autoencoder or autoassociator (Rumelhart et al., 1986), and it shares some interesting properties with unsupervised neural networks, for example the ability to learn compressed, distributed representations by performing dimensionality reduction on the input signal.

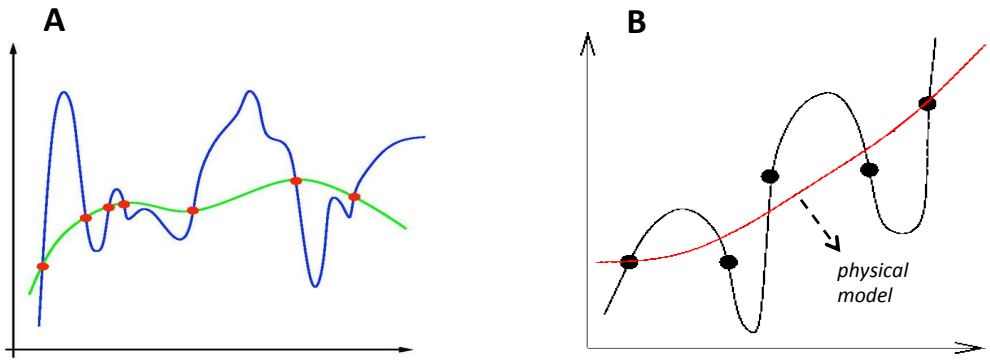

Figure 5. **(A)** Overfitting phenomenon and the bias-variance tradeoff. **(B)** Overfitting due to noise in the data points.

On the other hand, we can also attach label information to the input pattern, for example by simply concatenating the input and output vectors to form a single, larger vector. By training an unsupervised network to reconstruct this augmented input pattern, we are also creating a non-linear, stochastic mapping between the input and its corresponding label (Hinton, Osindero, & Teh, 2006).

Another important point to mention is related to the complexity of the learning model, which in the case of neural networks is related to the total number of connection weights (representing the fitting parameters). Due to the limited size of the training dataset, a network with many weights might extract too specific statistical information from the training patterns, thereby preventing the model to properly generalize to novel examples. This phenomenon is known as overfitting, and can be conveniently illustrated considering the problem of fitting a function of two variables according to some given, observed data points. As shown in Fig. 5A, there exist many (in fact, infinite) functions that exactly pass through the given data points. However, as the complexity of the function increases (dark line), its behavior becomes more instable, that is, the predictions for unseen points (either interpolation or extrapolation) can be significantly different from the real, underlying function from which the observed data points where collected. Simpler functions with less degrees of variability (light line) might learn a smoother mapping, which

results in a better fit for unseen data points. This phenomenon is particularly relevant in the presence of noise in the training data, because in this case more complex models are likely to fit also the noise and produce a bad generalization (Fig. 5B). A possible solution to control overfitting consists on restricting the hypothesis space of the model, thereby limiting model complexity using some form of regularization. A popular regularization technique for neural networks is known as *weight decay*, which can be implemented by appending an additional penalty term in the error function, which prevents the weights to grow indefinitely. Overfitting can be reduced also by significantly increasing the number of patterns contained in the training dataset, thereby allowing the network to extract more robust statistical information during learning. As we will discuss later, this approach has recently became particularly effective thanks to the advent of efficient parallel computing architectures and the availability of very large digital datasets.

The final major learning scheme is represented by *reinforcement learning*, where the system is not given an explicit teaching signal from an external supervisor, but only receives an implicit feedback from the environment through a reward function (Sutton & Barto, 1998). In this thesis we are not concerned with this learning regimen, though some interesting research work has recently suggested how hierarchical neural networks can be exploited within a reinforcement learning setting (Lange & Riedmiller, 2010; V. Mnih et al., 2013).

2.1.4 The rise of connectionist models

The popularity of PDP models significantly grew during the last decades. In this section we try to give a general (though necessarily incomplete) overview of the types of cognitive phenomena that have been addressed using connectionist models.

The description of mental states in terms of (possibly dynamical) patterns of activation arising in neural networks allowed to radically change the way we think about knowledge, and how it is acquired and represented in nervous

systems. In particular, a fundamental concept adopted by many connectionist models is that of *distributed representations*, which suppose that knowledge is encoded by patterns of activity distributed over many units, and each unit is involved in representing many different entities (Hinton, McClelland, & Rumelhart, 1986). This powerful representational scheme is extremely efficient, because it allows to encode an exponential number of patterns by means of compositionality (i.e., in the simplified case of n binary neurons we can encode up to 2^n patterns). Distributed representations entail many other important advantages over localized representations, for example by allowing the gradual emergence of new dynamics within the network and by improving resilience to noise (e.g., the corruption of few processing units does not compromise the global behavior of the system). The former feature has been shown very useful for studying cognitive development (Elman et al., 1996), because the learning trajectory underlying knowledge acquisition in artificial neural networks can be precisely measured and compared to available experimental data (e.g., Plunkett & Marchman, 1993). Similarly, the graceful degradation property has been useful to study acquired disorders in cognitive neuropsychology, where the performance of a system is impaired due to some damage (either by a slow, diffuse damage such as in dementia, or by some abrupt, localized damage such as in brain stroke). Neural network models allow to better understand which might be the underlying processes following brain damages, for example by simulating the change in behavior of the system after the application of virtual lesions (e.g., Hinton & Shallice, 1991; Joanisse & Seidenberg, 1999). Another important intuition stemming from these ideas concerns the nature of knowledge representation in nervous systems: within the PDP framework, knowledge does not need to be explicitly represented (as assumed by most of the symbolic approaches), because it is implicitly encoded in the connections weights. Indeed, there have been many examples of connectionist models exhibiting abstract, "rule-based" behaviors even without relying on a set of explicitly defined rules. In particular, these types of phenomena have been intensively studied in the

language domain, where neural networks have shown able to learn linguistic structures usually described by compositional rules at the morphological (e.g., Rumelhart & McClelland, 1986a), phonological (e.g., Burgess & Hitch, 1999; Nerbonne & Stoianov, 2004) and syntactic (e.g., Elman, 1990) levels. Related cognitive domains where connectionist models have been proven very successful include visual word recognition and reading aloud (e.g., McClelland & Rumelhart, 1981; Perry, Ziegler, & Zorzi, 2007; Plaut, McClelland, Seidenberg, & Patterson, 1996; Zorzi, Houghton, & Butterworth, 1998; Zorzi, 2010). At a more perceptual level, connectionist models have been applied to a variety of problems, including speech perception (e.g., McClelland & Elman, 1986), stereoscopic vision (e.g., Marr & Poggio, 1979), shape from shading (e.g., Lehky & Sejnowski, 1988) and spatial mapping (e.g., Zipser & Andersen, 1988), just to name a few. PDP models have been also important to investigate the specificity and generality of memory systems (McClelland, McNaughton, & O'Reilly, 1995), semantic cognition (McClelland & Rogers, 2003), executive control (e.g., Botvinick & Plaut, 2004), selective attention (e.g., Cohen, Dunbar, & McClelland, 1990) and mathematical cognition (e.g., Stoianov, Zorzi, Becker, & Umiltà, 2002; Stoianov & Zorzi, 2012; Zorzi, Stoianov, & Umiltà, 2005).

2.1.5 Looking ahead

Despite the broad range of domains that have been successfully addressed by PDP models, we believe that a deeper understanding of cognitive phenomena requires to more seriously consider some important principles that have often be neglected by connectionist modelers, as we already anticipated in Chapter 1.

In general, the intimate relation between structure and function in neural systems suggests that more informative models might be obtained by taking into account some established properties of cortical processing. In particular, the presence of massive feedback connections between cortical areas (Sillito, Cudeiro, & Jones, 2006) might be a critical architectural feature to improve neural processing by means of reciprocal interaction among different sources of

information. At the same time, many cortical areas exhibit a certain degree of hierarchical organization (Felleman & Van Essen, 1991), which might also constitute a critical architectural feature to efficiently processing information by exploiting multiple levels of representation. It therefore appears that cognition (including its intertwined perceptual and motor aspects) entails a delicate interplay between bottom-up, sensory information and top-down, contextual influences, which mutually constrain each other at many levels of organization (P. S. Churchland, Ramachandran, & Sejnowski, 1994; McClelland, Mirman, Bolger, & Khaitan, 2014).

The effects of contextual information in perception and cognition have been extensively studied by experimental psychologists, which accumulated a great deal of empirical evidence describing the role of top-down, "endogenous" processing in many domains. Well-known examples of these phenomena include the perception of ambiguous or noisy stimuli, which can be correctly interpreted only by using high-level, contextual information. Similarly, many of the classic effects described by Gestalt theory (e.g., Kanizsa, 1979) can be conveniently described as a form of top-down processing. Despite the most popular examples come from the visual domain, it should be noted that these phenomena are commonly found also in other sensory modalities. A striking example is provided by speech processing, where the sensory information appears incredibly poor and noisy given the complexity of the underlying linguistic constructs (e.g., Marslen-Wilson & Welsh, 1978). Another intensively studied domain is that of multi-sensory integration (e.g., McGurk & MacDonald, 1976), where feedback projections are thought to support cross-modal interactions among different cortical areas (Falchier & Clavagnier, 2002). Top-down processing has also central relevance in attentional mechanisms, allowing to modulate information processing according to expectations, goals, task instructions and other types of biases (Kastner & Ungerleider, 2000). Notably, these empirical effects find support also from physiological and functional neuroimaging studies. For example, it has been shown that the receptive fields of

neurons in primary sensory areas can be dynamically adjusted according to contextual information coming from higher processing areas: in a single cell recording study, neurons in the macaque V1 cortex have shown to respond to illusory contours of the Kanizsa figures, but only after the same type of response was first generated in area V2 (T. S. Lee & Nguyen, 2001). Similarly, it has been observed through fMRI measurements that perceptual expectation improves stimulus representation in human early visual cortex (Kok, Jehee, & de Lange, 2012).

From a computational perspective, the problem of finding the best possible interpretation of an ambiguous stimulus can be conveniently formalized in probabilistic terms as an unconscious, statistical inference process (von Helmholtz, 1925). This idea has recently regained popularity thanks to the spread of the "Bayesian brain" hypothesis (Knill & Pouget, 2004), which proposes to model cognitive processes using the powerful formalism of Bayesian statistics (e.g., Chikkerur, Serre, Tan, & Poggio, 2010; Griffiths, Kemp, & Tenenbaum, 2008; Norris, 2006; Tenenbaum, Kemp, Griffiths, & Goodman, 2011; Yuille & Kersten, 2006). According to the Bayesian view, the brain represents information probabilistically, by coding and computing with (possibly approximated) probability density functions. A possible role for recurrent feed-forward/feedback loops in the cortex might therefore be to integrate top-down, contextual priors and bottom-up, sensory observations, so as to implement concurrent probabilistic inference along the whole cortical hierarchy. Notably, it has also been suggested how Bayesian inference algorithms might be actually implemented in neural circuits (e.g., Lee & Mumford, 2003), with some relevant proposals providing a tight link between probabilistic inference and computation in interactive neural networks (Lochmann & Deneve, 2011; McClelland, 2013; Rao, 2004).

However, Bayesian models often rely on explicit, structured representations to perform inference tasks. A fundamental step forward to characterize neural computation within a probabilistic framework also requires

to specify how the brain might efficiently discover such statistical structure, that is, how learning processes might shape an internal, probabilistic model of the environment according to experience, and which is the role of processing constraints in efficient implementations of Bayesian reasoning systems (for a comprehensive discussion about the strenghts and the limits of current Bayesian models of cognition, see Jones & Love, 2011). We thus believe that a promising direction to complement the current Bayesian treatment of cognitive processes is to consider stochastic, recurrent neural networks that can learn probabilistic generative models from a set of training examples (e.g., Dayan et al., 1995; Hinton & Ghahramani, 1997). Moreover, in deep learning systems many simple networks are stacked together in order to learn more expressive, hierarchical generative models (Bengio, 2009; Hinton & Salakhutdinov, 2006). Deep architectures composed by many processing layers reach impressive performance on many difficult learning tasks (Krizhevsky et al., 2012; Mohamed, Dahl, & Hinton, 2012) and open the way for creating cognitive models that can infer causes and make predictions exploiting multiple levels of representation (Hinton, 2007, 2010b, 2013). Structure discovery in deep networks can be efficiently accomplished by means of unsupervised learning (Hinton et al., 2006), thereby showing how abstract representations could emerge by extracting useful statistics from the training data. As we will discuss in detail in Chapter 3, this family of neural networks can be conveniently described within the formalism of probabilistic graphical models (Jordan & Sejnowski, 2001; Koller & Friedman, 2009), which therefore allows to characterize Bayesian models and recurrent neural networks using a common language. Finally, the introduction of efficient learning algorithms (Bengio & Lamblin, 2007; Hinton et al., 2006) that can be easily implemented in parallel computing machines allows building large-scale deep learning systems (Krizhevsky et al., 2012; Raina et al., 2009), which rely on big digital datasets to extract more reliable statistical information from the training examples.

2.2 Orthographic processing

Even if literate adults appear to recognize written words without any effort, educators and developmental psychologists are well aware that learning to read is one of the hardest challenges faced by children in school (Dehaene, 2009). Despite its apparent simplicity, the recognition of written symbols and their combination into more complex orthographic patterns constitutes an extremely challenging task for the visual system. In this section, we will review some basic facts about the primate visual system, focusing on the processing pathway usually identified as the "ventral stream" (Goodale & Milner, 1992). We then give a computational characterization of visual object recognition, and we discuss its relation to the related problem of recognizing written patterns. We finally present some important characteristics of popular computational models proposed to study written character recognition and orthographic processing.

2.2.1 Structure and function of the ventral visual stream

The visual system has a central role in acquiring and exploiting knowledge about the environment, as exemplified by the fact that almost half of the non-human primate cerebral cortex is devoted to processing of visual information (Felleman & Van Essen, 1991). Visual processing entails an incredibly rich and multifaceted variety of tasks, and despite the great amount of research focusing on this domain we are still lacking a comprehensive picture describing the visual system from a broad perspective. However, a certain consensus is given to the hypothesis that visual object recognition is primarily supported by neural circuits located in the ventral visual steam (see Fig. 6), which enable accurate perceptual identification of objects in the visual scene by relying on a hierarchy of processing stages (DiCarlo, Zoccolan, & Rust, 2012; Kruger et al., 2013).

Neurons in the early levels of the hierarchy extract simple visual features over local regions of the visual field, while neurons in higher cortical areas respond to ever more complex features, with receptive fields covering larger and larger portions of the visual scene.

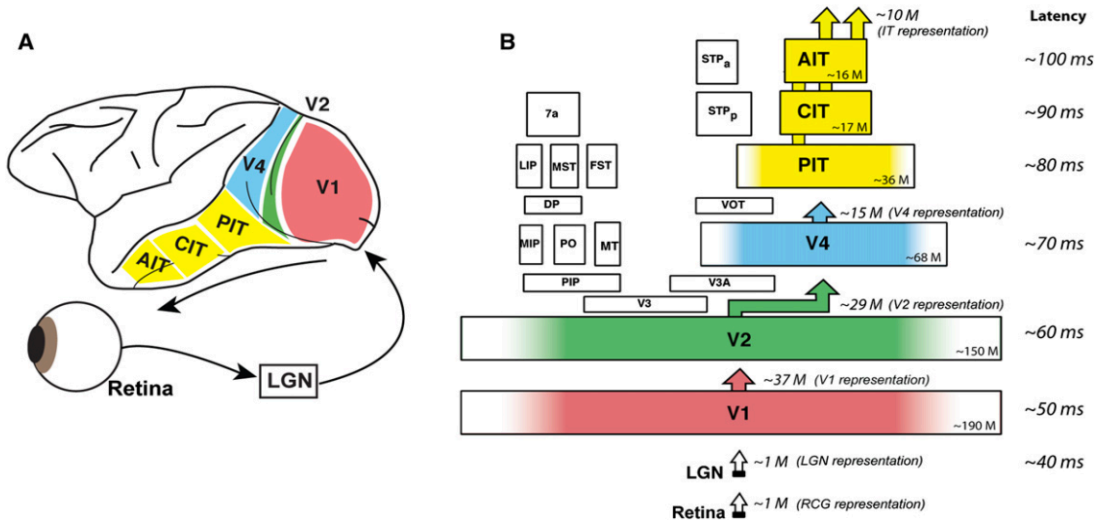

Figure 6. **(A)** Schematic representation of the ventral stream in a primate macaque brain, covering a set of cortical areas arranged along the occipital and temporal lobes. **(B)** Fine-grained parceling of the ventral stream, highlighting its hierarchical organization and the dimensionality of each processing stage. Adapted from (DiCarlo et al., 2012).

At the subcortical stages of processing, visual information flows from the photoreceptors of the retina through ganglion cells to reach the lateral geniculate nucleus of the thalamus. Though the functional role of these early areas is still partially unknown (Niell, 2013), they appear to perform some kind of signal filtering (Atick & Redlich, 1992), for example by computing spatial and temporal decorrelations of the input signal (Dong & Atick, 1995), which are usually simulated by applying whitening algorithms to input images (Simoncelli & Olshausen, 2001). The lateral geniculate nucleus is then massively connected to the primary visual cortex (V1), which still processes relatively simple visual features such as edges, gratings, oriented bars and line endings (Hubel & Wiesel, 1962). Notably, it has been shown that this type of features can be captured by trying to efficiently encode the statistical information contained in natural images, for example by building sparse codes (Olshausen & Field, 1996), by discovering independent components (Bell & Sejnowski, 1997; van Hateren & van der Schaaf, 1998), or by creating predictive codes (Rao & Ballard, 1999). As

we will show in Section 4.2, these features can also emerge in stochastic neural networks that learn a generative model of the input patterns. Moving up in the hierarchy, in area V2 neurons encode more sophisticated contour representations, including corners, junctions and illusory contours (Ffytche & Zeki, 1996; T. S. Lee & Nguyen, 2001). Deeper levels of the extrastriate and associative cortex eventually represent complex visual patterns such as object parts (Orban, 2008), with some neurons becoming shape-tuned (Tanaka, 1996) and sometimes reaching impressive levels of response specificity (Quian Quiroga, Reddy, Kreiman, Koch, & Fried, 2005).

From a computational perspective, the task of recognizing objects contained in a visual scene is extremely challenging, because the visual appearance of a certain object might change drastically depending on the circumstances. For example, changes in position, distance, illumination and perspective significantly alter the visual pattern generated by an object in the retina, and the recognition task is made even harder due to the presence of occlusions, noise and many forms of perceptual ambiguity. The object recognition pathway must therefore cope with the problem of finding useful invariances in the sensory signals, which allow to recognize specific objects even under a variety of identity-preserving visual transformations, and to generalize the acquired knowledge to similar objects that have never seen before. Importantly, the ability to exploit abstract object representations must be efficiently implemented in neural circuits, which need to operate in real-time in rapidly changing environments. A geometric interpretation of this computation has been discussed in Section 2.1.1, where pattern recognition was framed as the problem of finding a discriminant function that correctly separates a set of data points. This mapping is achieved by placing a decision boundary in the representational space of the data, which corresponds to a hyperplane in the case of linear classifiers such as those implemented in single-layer networks.

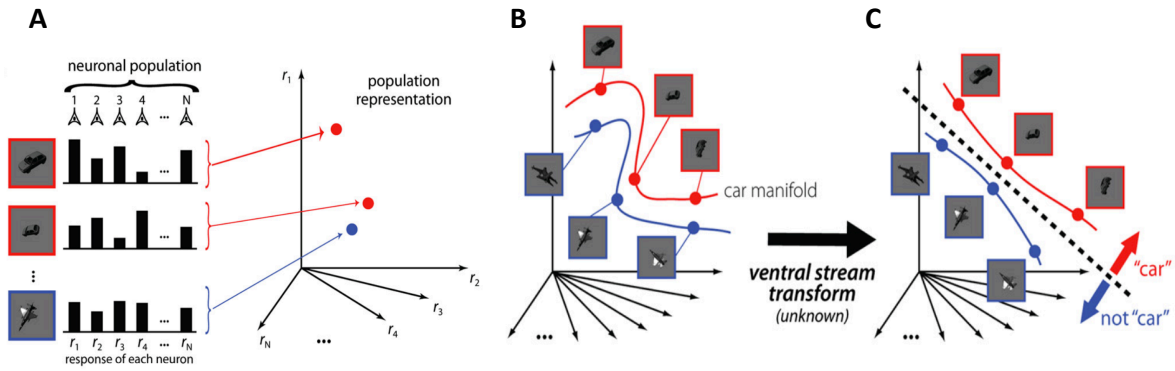

Figure 7. **(A)** Images representation as points in a high-dimensional vector space. **(B)** Retina representations correspond to highly tangled manifolds. **(C)** Disentangled manifold representation, which supports recognition through a linear decision boundary. Adapted from (DiCarlo et al., 2012).

However, vision operates in a high-dimensional space, which can be conceptualized as an extension of a simple three-dimensional Cartesian space in which each axis of the space is the response of one retinal ganglion cell (Fig. 7A). The response vectors representing all possible identity-preserving transformations of an object define a low-dimensional surface in this high-dimensional space, called the "object identity manifold" (DiCarlo & Cox, 2007). At early stages of visual processing, these manifolds are highly curved and tangled together, like pieces of paper crumpled into a ball (Fig. 7B). To correctly identify objects, retinal manifolds should be gradually transformed into a new form of representation, where they might be accurately separated even by a linear decision boundary (Fig. 7C). In order to mimic this type of transformation, popular computational models inspired by the visual cortex exploit a series of non-linear operations over a set of hand-coded features (Riesenhuber & Poggio, 1999; Serre, Wolf, Bileschi, Riesenhuber, & Poggio, 2007). However, as we will discuss later, the recent advent of deep learning models now allows to exploit a series of non-linear transformations operating over features extracted directly from the data, which allow to reach impressive performance on challenging visual object recognition benchmarks (Krizhevsky et al., 2012).

2.2.2 Recognizing written patterns

The task of recognizing written characters can be considered as a special case of object recognition, where objects are constituted by two-dimensional shapes drawn using a variety of writing styles. Despite the much-limited amount of possible object classes (e.g., 26 letters in the English alphabet), this visual recognition problem remains extremely challenging due to the large variability of character shapes, which is particularly evident in handwritten, cursive scripts. Moreover, the mapping between visual forms and letter identities is often quite arbitrary, forcing us to neglect significant differences (e.g., by identifying **a** and ***a*** within the same class) while attending fine-grained visual details (e.g., by identifying **i** and **j** as different classes). Finally, additional complexities come into play when considering combinations of several individual characters to form written words, which must be recognized despite transposition effects (such as in **dog** and **god**) or similarities caused by shared letters (such as in **confusion** and **contusion**).

The ability to recognize written patterns appears even more remarkable if we consider that reading is a recent cultural invention, which has been probably introduced less than 6000 years ago (Dehaene, 2009). This implies that evolutionary mechanisms did not have enough time to shape the human visual system in order to better support reading abilities, which must therefore be acquired through education. However, an intriguing hypothesis suggests that some cultural inventions (such as reading and arithmetic) are not learned completely from scratch, but they instead "invade" evolutionarily older brain circuits, thereby also inheriting many of their structural constraints. According to this *neuronal recycling* hypothesis (Dehaene & Cohen, 2007), writing systems do not rely on a set of randomly drawn strokes to encode words as two dimensional images, but they rather exploit visual patterns whose structure can be effectively processed by our primate visual system, and in particular by the object recognition pathway. Indeed, it appears that visual signs used in different scripts have been culturally selected to match the kinds of conglomeration of

contours found in natural scenes (Changizi, Zhang, & Ye, 2006). The recycling hypothesis is also supported by several neuroimaging studies that identified localized cortical circuits specifically involved in orthographic processing (Wandell, 2011). Not surprisingly, these circuits partially overlap with the object recognition pathway, and are usually referred to as the "visual word form area" (L. Cohen & Dehaene, 2004; L. Cohen et al., 2000; McCandliss, Cohen, & Dehaene, 2003).

The fact that orthographic processing and visual object recognition share many structural and functional properties suggests that also the neural code for written words might be efficiently organized into a hierarchy of increasingly complex representations (Dehaene, Cohen, Sigman, & Vinckier, 2005). Lower levels might encode basic visual features such as edges, curvatures and combinations of simple geometrical shapes that allow reliable letter identification (Fiset et al., 2008; Grainger, Rey, & Dufau, 2008). After single letters have been identified by their component features, their positional information can be taken into account in order to build more structured representations, which encode combinations of letters such as bigrams and open bigrams (Grainger, 2008), eventually allowing to represent morphemes and entire words. It should be stressed that this multilevel processing is supposed to be highly interactive, with higher-level information constantly shaping and being shaped by representations created at lower levels.

A central issue that is still intensively debated is to what extent linguistic information is necessary to recognize written words. Indeed, reading and writing systems have been specifically created to support and expand the capability of human communication, thus linguistic knowledge (e.g., phonological representations and syntactic constructs) might have an important role in orthographic processing. At the same time, the ability to recognize complex configurations of written patterns might be partially obtained even without relying on linguistic information. Some preliminary evidence supporting this view has been recently provided by a study on non-human primates, on

which baboons have been trained to accurately discriminate between existing English words and nonsense sequences of letters presented through a monitor (Grainger, Dufau, Montant, Ziegler, & Fagot, 2012). Though this finding deserves more careful investigation, it nevertheless suggests that written word recognition might be largely dependent on pure visual information, which in the primate brain is efficiently represented and processed by a hierarchy of neural circuits located in the ventral stream.

2.2.3 Computational models of orthographic processing

As we discussed in Section 1.3, recognition of printed patterns has often been considered as a prototypical problem for validating computational theories of cognition. In particular, one of the first attempts to apply automatic recognition systems to printed letters dates back to the Pandemonium model (Selfridge, 1959) whose hierarchical architecture has been a great source of inspiration for later models such as the Neocognitron (Fukushima, 1980, shown in Fig. 8A) and the seminal Interactive Activation Model (IAM; McClelland & Rumelhart, 1981, shown in Fig. 8B). The fundamental contribution of the IAM has been to explain how the context surrounding a certain letter might affect its correct identification by means of interactive activation and mutual inhibition among representational units organized into multiple layers. This critical property provided a computational explanation of the puzzling word superiority effect (Cattell, 1886; Reicher, 1969), which entails better recognition of letters presented within words as compared to isolated letters and to letters presented within nonwords. Even though one of the main goals of the IAM was to discuss the relevance of interactivity and hierarchical processing in perception, it therefore also provided a reference framework to model visual word recognition more in general. However, a major limitation of the IAM is the hand-coded nature of its representational features, which makes the model unable to learn or to adapt according to experience.

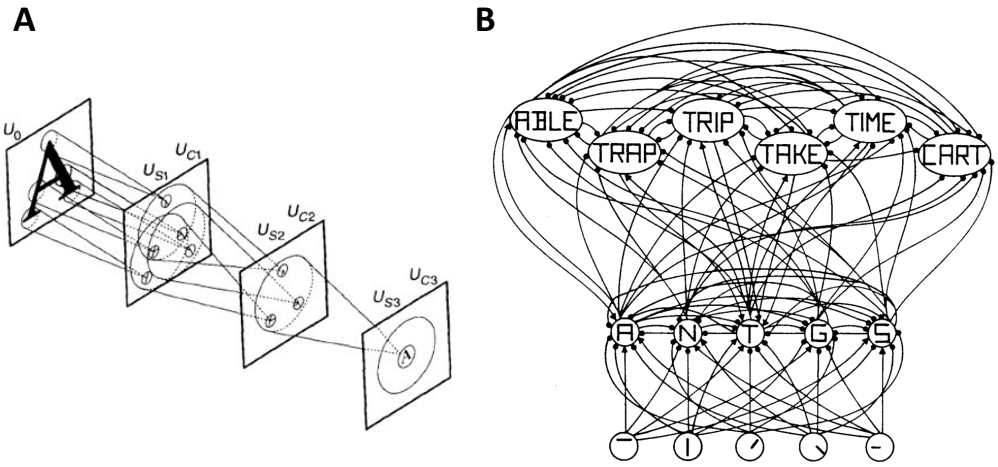

Figure 8. **(A)** The Neocognitron architecture for recognizing printed letters (Fukushima, 1980). **(B)** Schematic representation of the Interactive Activation Model for recognizing printed words (McClelland & Rumelhart, 1981).

Moreover, empirical research suggests that the letter and word representations entailed by the IAM are likely not to be the only functional orthographic units exploited in word recognition (Grainger, 2008). A crucial step forward to fully characterize the nature of orthographic processing thus requires to consider more flexible models, where learning mechanisms gradually shape the feature hierarchy according to experience. A successful, first attempt to implement such a system has been accomplished by convolutional neural networks, which obtain high performance on several challenging recognition tasks involving recognition of written text (LeCun, Bottou, Bengio, & Haffner, 1998). The subsequent advent of deep neural networks, which reach state-of-the-art accuracy on a benchmark task of handwritten digit recognition (Hinton & Salakhutdinov, 2006), suggests that a complete account of the processes involved in recognizing written characters and words might be within reach.

3 Generative Models and Deep Learning

Uncertainty is a ubiquitous characteristic of real-word problems, forcing us to consider not only certain but also *possible* events, and to quantify how *probable* they are, that is, which situations are more likely to be observed. In this chapter we review some useful concepts underlying the probabilistic modeling approach, which allow dealing with uncertainty in a principled and systematic way. Importantly, it turns out that some interesting recurrent neural network models can be described as specific instances of a broader class of undirected graphical models. We therefore discuss their representation, inference and learning capabilities within the probabilistic framework, and we explain how basic networks can be conveniently combined together in order to learn more powerful, hierarchical generative models.

3.1 Probabilistic graphical models

The framework of probabilistic graphical models (Koller & Friedman, 2009) provides a general approach to model arbitrarily complex statistical distributions, which can involve a large number of stochastic variables interacting together. Graphical models allow to describe complex relations between variables by exploiting the structure of their joint distribution, since in general interactions are not globally defined but instead involve only a limited subset of "neighbors". Probabilistic graphical models are usually defined by three main components: first, a formal structure that supports knowledge ***representation*** in terms of probability distributions; second, a series of ***inference*** procedures that allow to extract useful information from the knowledge structure; and, finally, a series of ***learning*** algorithms that can be

used to automatically build and refine the model structure by extracting knowledge from a given set of examples.

3.1.1 Graphical representation of probability distributions

In graphical models, probability distributions are represented using graph data structures, where a set of nodes (also called "vertices") represents random variables and a set of edges (also called "arcs" or "connections") represents their relations. The topology of the graph explicitly defines the scope of interaction of each variable, thereby highlighting the set of independencies that hold in the distribution. Graphical models can have ***directed*** connections between variables, such as in Bayesian networks (Fig. 9A), or ***undirected*** connections, such as in Markov random fields (Fig. 9B). Both types of connections might be present in the same graph, thereby forming a hybrid model. Though directed and undirected models share the underlying theoretical framework, they have rather different representational and computational capabilities. In what follows, our discussion will primarily focus on undirected models.

In Bayesian networks, the semantic of connections defines parent-of relationships between linked nodes. A conditional probability distribution (CPD) is associated with each node (note that for discrete distributions it can be easily encoded in a table).

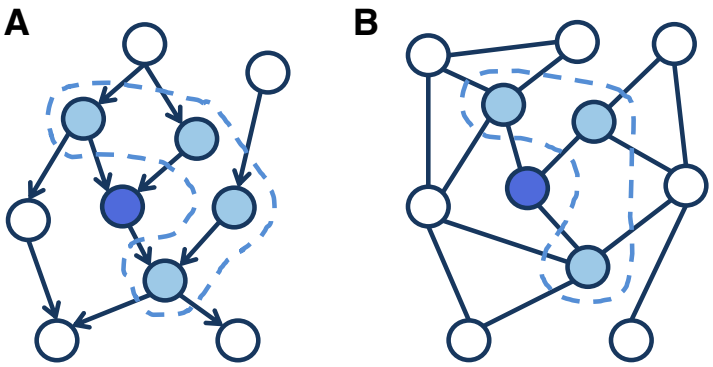

Figure 9. **(A)** Example of a directed model (Bayesian network). **(B)** Example of an undirected model (Markov random field). In both examples, a dashed line highlights the Markov blanket of the blue node.

The direction of the arrow attached to each edge indicates the presence of a direct dependence between variables: an arrow going from node A to node B entails that the variable A has a direct influence on the variable B. The variable B is referred to as a "child" of A, which in turn is referred to as a "parent" of B. Notably, this type of relationship allows to define a local semantic criterion for directed models: each node is conditionally independent from all the other nodes in the graph, given its parents, its children and the parents of its children. This set of nodes (highlighted in Fig. 9) is usually referred to as the "Markov blanket" of the variable (Pearl, 1988). Conditional independencies greatly reduce the number of parameters required to characterize a joint distribution, because they allow to compactly represent it through factorization:

$$P(x_1, x_2, \dots, x_n) = \prod_i^n P\left(x_i \middle| Parents(x_i)\right) \qquad (5)$$

As we will discuss later, factorization can be conveniently exploited to derive efficient inference and learning procedures even in the presence of a large number of variables, because only the Markov blanket of a certain node is required in order to sample from its conditional distribution.

In undirected models, the symmetric nature of the connections does not describe parent-of relationships, but instead encode a degree of correlation between the linked variables. This leads to a different local semantic criterion to represent independencies: each node is conditionally independent from all the others, given the nodes directly connected to it. Moreover, instead of relying on a separate CPD to represent the local probability distribution of each variable, in undirected models each edge is associated with a certain function, known as factor, which takes as input the values of the connected nodes and gives as output a non-negative scalar value representing their affinity. A factor with high value indicates that the two variables are strongly correlated, while a low value indicates a weak relation. This implies that also in undirected models the complete joint distribution can be efficiently defined as a product of local factors:

$$P(x_1, x_2, \dots, x_n) = \frac{1}{Z}\prod_i^n \emptyset_i(D_i) \quad (6)$$

where D_i represents the scope of each factor $\emptyset_i$ (i.e., which variables it involves) and Z is a global normalization constant called *partition function*, which ensures dealing with legal probabilities summing up to 1:

$$Z = \sum_D \prod_i^n \emptyset_i(D_i) \quad (7)$$

Importantly, the factorization specified in Eq. 6 could be defined using an equivalent notation based on a log-linear, feature-based parameterization, where features are obtained by applying a logarithmic function to each factor and can thus assume also negative values. In many cases, it might also be convenient to use binary features, which can be combined using different weights in order to specify the final degree of affinity:

$$P(x_1, x_2, \dots, x_n) = \frac{1}{Z}\mathrm{e}^{-\boldsymbol{wf}(D)} \quad (8)$$

where ***f(D)*** is the feature vector and ***w*** indicates the corresponding vector of weights. Log-linear models provide a much more compact representation for many distributions, especially in situations where variables have large domains. Moreover, as we will discuss in Section 3.2.1, this representation can be conveniently used to define energy-based models.

3.1.2 Inference and learning algorithms

The graphical representation of a probability distribution can be used to answer queries about the behavior of specific variables. Here we focus on the inference task arising when certain variables are observed and the goal is to infer the conditional probabilities of a separate set of target variables. In general, Bayes' theorem provides a formal rule for expressing how a subjective degree of belief (i.e., the probability associated with a certain variable) should be

rationally changed to account for observed evidence. In particular, the posterior probability of a variable *A* conditioned on the current evidence *B* can be computed as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (9)$$

Where *P(B|A)* represents the likelihood (i.e., the degree of belief in *B*, given that *A* is observed) and *P(A)* represents the prior (i.e., the initial degree of belief in *A*). The term *P(B)* accounts for the prior probability of observing the evidence *B*. This simple inference rule can be used in graphical models to compute conditional probabilities over many interacting variables, and several exact algorithms have been proposed to effectively implement it (e.g., variable elimination and message passing schemes). However, the computational complexity of these algorithms depends on the network structure, and it quickly grows as more complex topologies are considered. In particular, exact inference becomes intractable for networks with a large tree-width, where ***approximate inference*** techniques must be adopted.

A popular family of approximate inference algorithms is that of particle-based methods, which estimate the target conditional distribution by generating samples according to some criteria. In particular, a widely used sampling method is the ***Gibbs sampling*** algorithm, which can generates a sequence of samples that progressively approximate a specified multivariate probability distribution (Geman & Geman, 1984). Gibbs sampling belongs to the family of Markov Chain Monte Carlo (MCMC) methods, which draw samples from a probability distribution by constructing a Markov chain that has the desired distribution as its equilibrium distribution, and where each state of the chain represents a possible assignment to some variables of the network (Andrieu, De Freitas, Doucet, Jordan, & Freitas, 2003). Although the first samples may be inaccurate, under certain conditions the Markov chain will converge to the stationary distribution, thereby allowing to generate samples that provably get

closer to the desired target distribution. During Gibbs sampling, only one variable is sampled at each step, keeping fixed the current values of all the other variables. However, we can exploit the structure of the graph (i.e., the conditional independencies between the variables) to speed up the sampling process: since the value of each node is only influenced by its Markov blanket, if two variables are conditionally independent given the current evidence (i.e., their Markov blanket is observed) they can be sampled at the same time. This parallel variant is known as block Gibbs sampling, and it is heavily used in the recurrent neural network models that we will consider later.

Up to this point, we discussed representation and inference tasks assuming that the graph structure and the associated model parameters are given. However, a fundamental problem is to find effective ways to automatically learn them from available data. For simplicity, here we focus on the case where the graph structure is given and the goal is to estimate the model parameters associated with each edge. Within this setting, the goal of learning is to find a proper set of parameters such that the resulting model distribution captures the real distribution from which training data were sampled. Due to the limited amount of training examples, in practical settings the goal of learning is to find a model distribution that best approximates the real distribution.

Learning algorithms are usually derived from ***maximum likelihood*** methods, whose aim is to select the values of the model parameters that maximize the "agreement" of the selected model with the observed data. This can be achieved by applying gradient-based methods to maximize the likelihood function with respect to the model parameters. Interestingly, it has been shown (Jordan & Sejnowski, 2001) that maximizing the likelihood of the training data is equivalent to minimize another quantity, the Kullback-Leibler (KL) divergence, which measures the discrepancy between the model distribution and the empirical distribution observed in the data. Learning can thus be framed in terms of finding a model distribution Q that best approximates the real

distribution P, where the quality of this approximation is measured as the KL-divergence between P and Q:

$$D_{KL}(P||Q) = \sum_x P(x)\log\frac{P(x)}{Q(x)} \qquad (10)$$

Intuitively, the KL-divergence (also called relative entropy) measures the extent of the compression loss (in bits) of using Q rather than P, and it is zero when $P = Q$ and positive otherwise (Kullback & Leibler, 1951).

3.2 Generative models

Learning in probabilistic graphical models can be generally framed within two different settings. In discriminative learning, the goal is to model only conditional distributions over a set of target variables, whose values are specified in the training set by attaching an explicit value (i.e., a label) to each example. In generative learning, instead, the aim is to model the complete joint distribution of all the variables in the graph, thus including also the observed variables. Discriminative models are usually suited for classification and regression tasks, where we are only interested in the output response produced by the system. Generative models, instead, extract regularities from the data in an unsupervised way, in order to construct rich and expressive representations that can be successively exploited to perform other (possibly supervised) tasks. As an example, we can consider the challenging task of recognizing objects in a visual scene. The pixel values in realistic images are not randomly drawn, but are generated by some (possibly complex) physical processes, and the appropriate response to an image nearly always depends on the physical causes of the image rather than the pixel intensities. This suggests that unsupervised learning can be used to first solve the difficult problem of extracting the underlying causes from the visual scene, and decisions about responses can be left to a separate, supervised learning algorithm that takes the underlying causes rather than the raw sensory data as its inputs.

In this section we formally describe Boltzmann machines, which are stochastic recurrent neural networks that can learn probabilistic generative models from a set of training examples.

3.2.1 Restricted Boltzmann Machines

A Boltzmann machine (Ackley et al., 1985) is a particular type of undirected graphical model with a fully connected graph structure (a graphical representation is provided in Fig. 10A). The undirected nature of the edges implies that connections are symmetric, that is, the flow of information between the nodes (often referred to as "neurons" or "units") is bidirectional. Each connection is associated with a scalar value representing the synaptic strength between the units, which corresponds to a factor in the graphical model framework. Units can be conveniently grouped in two separate sets, usually referred to as "layers": a visible layer is used to provide an external input to the network, while a hidden layer is used to encode the latent causes of the input. Hidden variables greatly increase the expressive power of the network by allowing to model distributions over visible variables that cannot be directly captured by pairwise interactions. In particular, they allow to encode binary features representing higher-order statistics observed in the data. During learning, the strength of each connection weight is gradually adapted as to construct an internal generative model that produces examples with the same probability distribution as the examples contained in the training dataset. At the end of the learning process, when shown any particular example the network can "interpret" it by finding values of the variables in the internal model that would generate the example.

Boltzmann machines were initially proposed as a stochastic variant of Hopfield networks, and their original formulation was indeed inspired by statistical mechanics: the behavior of the network is driven by an energy function E that describes which configurations of the units are more likely to occur by assigning them a certain probability value:

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h})} \qquad (11)$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ are, respectively, the vectors of visible and hidden units and Z is the partition function, which ensures that the values constitute a legal probability distribution (i.e., summing up to 1). The network state changes in a way that allows a gradual decrease of the associated energy, modulated by a "temperature" parameter so that at higher temperatures an occasional increase of energy is also permitted to avoid local minima.

Due to the full connectivity of the network, the system takes a long time to settle into thermal equilibrium. However, in Restricted Boltzmann Machines (RBMs; Smolensky, 1986) the connectivity of the network is constrained in order to form a bipartite graph (see Fig. 10B), which allows to greatly improve convergence on stable states. The energy function of an RBM is defined as:

$$E(v, h) = -\boldsymbol{a}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{v}^{\mathrm{T}}W\boldsymbol{h} \qquad (12)$$

where W represents the matrix of connection weights and $\boldsymbol{a}$ and $\boldsymbol{b}$ represent, respectively, the biases of the visible and hidden units.

From the perspective of probabilistic graphical models, this formulation of RBMs in terms of an energy function is equivalent to a factorization of the joint distribution induced by a log-linear parameterization based on binary indicator features (see Eq. 8). Within this interpretation, we can derive the activation probability of each hidden and visible unit in the network as a conditional probability, which specifies how we should update the value of the unit depending on the activation state of its neighbors:

$$P\left(h_j = 1 \middle| \boldsymbol{v}\right) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{v}+b_j}} \qquad (13)$$

$$P(v_i = 1|\boldsymbol{h}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{h}+a_i}} \qquad (14)$$

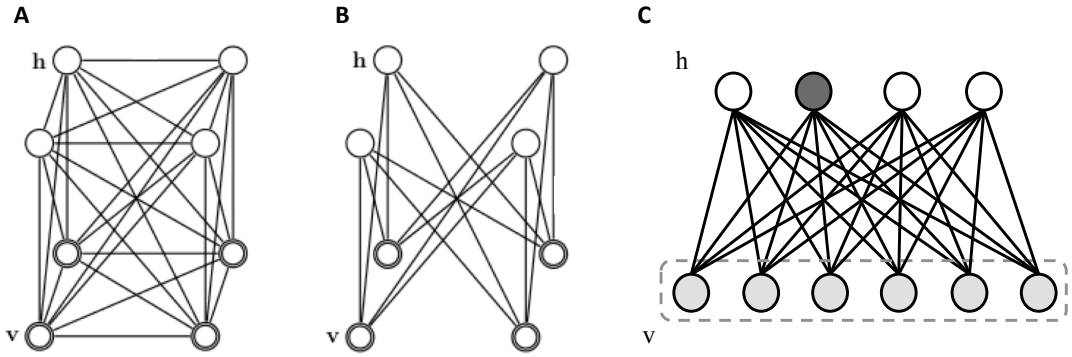

Figure 10. **(A)** A fully connected Boltzmann Machine. **(B)** The corresponding Restricted Boltzmann Machine (RBM). **(C)** An alternative representation of an RBM highlighting the Markov blanket of a hidden unit (dark gray), which coincides with the whole layer of visible units (and vice versa).

By iteratively sampling each unit's state according to this conditional probability, we are actually implementing Gibbs sampling for the Boltzmann distribution specified in Eq. 11. Interestingly, this sampling rule corresponds to the sigmoid activation function usually adopted in feed-forward neural networks (see Eq. 2 and Eq. 3).

Despite RBMs might not be able to represent efficiently some distributions that could be represented compactly with unrestricted Boltzmann machines, it has been demonstrated that an RBM can represent any discrete distribution if enough hidden units are used (i.e. RBMs are universal approximators of discrete distributions), and that adding hidden units strictly improves modeling power (Le Roux & Bengio, 2008).

3.2.2 Inference and learning in RBMs

If the visible units are externally forced or "clamped" into a particular state corresponding to a training pattern, the network will find the minimum energy configuration that is compatible with that input. The fundamental property of RBMs is that hidden units are independent of each other when conditioning on visible units (i.e. when visible units are clamped), and, vice versa, visible units are independent of each other when conditioning on hidden units. In other

words, the Markov blanket of each unit coincides with the variables of the opposite layer, as highlighted in Fig. 10C. This implies that we can infer the conditional probabilities of an entire layer of units using Eq. 13 and Eq. 14 in parallel, by performing block Gibbs sampling:

$$P(\boldsymbol{h}|\boldsymbol{v}) = \prod_i P(h_i|\boldsymbol{v}) \quad (15)$$

$$P(\boldsymbol{v}|\boldsymbol{h}) = \prod_i P(v_i|\boldsymbol{h}) \quad (16)$$

This efficient inference procedure can also be used to remove noise from corrupted patterns or to accurately reconstruct incomplete patterns, because the noisy or missing values in the visible layer can be iteratively resampled until they converge to a stable configuration. However, inference in RBMs becomes less efficient when the evidence is weak, that is, when only few visible units are observed or when the noise in the input pattern is high, because in these cases block Gibbs sampling might require many iterations to converge. In other words, in RBMs exact inference is fast when we condition the model on a given input pattern, but it becomes intractable when the model is unconstrained and we have to sample from its internal distribution. This limitation makes learning extremely challenging.

Learning in RBMs shapes the energy function in such a way that it will assign low values (i.e., high probability) to configurations of the variables that are more likely to occur, and high values (i.e., low probability) to undesired configurations. Since the form of the energy function is determined by the weight matrix (see Eq. 12), it can be shaped by changing the weights in such a way that the network will accurately reconstruct the observed training patterns. As we explained in Section 3.1.2, this can be accomplished by maximizing the log-likelihood function (i.e., the probability that the model assigns to each training pattern), which is equivalent to minimize the KL-divergence between the empirical data distribution and the model distribution (see Eq. 10). Because

the energy is a linear function of the weights, it turns out that the gradient of the log-likelihood with respect to a weight has a particularly simple form:

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \qquad (17)$$

where the two terms in angle brackets indicates, respectively, expectations under the data distribution and expectations under the unconstrained model distribution. The first term can be easily computed by measuring the correlations between visible and hidden activations when the variables in the visible layer are clamped to a training pattern and the variables in the hidden layer are inferred in a single step using Eq. 15. The second term, instead, measures the correlations when the network is not conditioned on a data vector, but is free to evolve according to its internal dynamics defined by the current model parameters. As we mentioned above, this requires sampling from the unconstrained model distribution, which might take a significant amount of time. Once these two quantities have been computed, each weight can be changed by using a simple update rule:

$$\Delta w_{ij} = \eta(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \qquad (18)$$

where η is a small learning rate. A surprising feature of this learning rule is that it uses only locally available information: the change in a connection weight depends only on the behavior of the two units it connects, even though the change optimizes a global measure, and the best value for each weight depends on the values of all the other weights.

The computational cost required to sample from the model distribution initially prevented to use RBMs in practical settings. However, it has lately been shown that this complexity can be avoided by adopting a slightly different learning procedure, called *contrastive divergence* (Hinton, 2002). The basic intuition behind contrastive divergence is that the network does not need to be

run until equilibrium in order to compute model's expectations: if sampling starts from the hidden unit activations computed in the data-driven phase, correlations computed after only a fixed number of steps in block Gibbs sampling are sufficient to drive the weights toward a state in which the input data will be accurately reconstructed. Hence, contrastive divergence approximates model's expectations by conditioning the initial activations of hidden units and performing only a fixed number of sampling iterations (see Fig. 11 for a graphical representation of this process). Surprisingly, in practice it has been shown that even a single iteration can produce good results. Intuitively, contrastive divergence works by increasing the likelihood of training data and decreasing the likelihood of "model confabulations", which are distortions of the training patterns that the model produces when it is left free to modify them according to its current parameters. This implies that contrastive divergence learning approximates the log-likelihood gradient locally around the training point from which the chain starts, decreasing the energy of that point and increasing the energy of its neighborhood (Bengio, 2009).

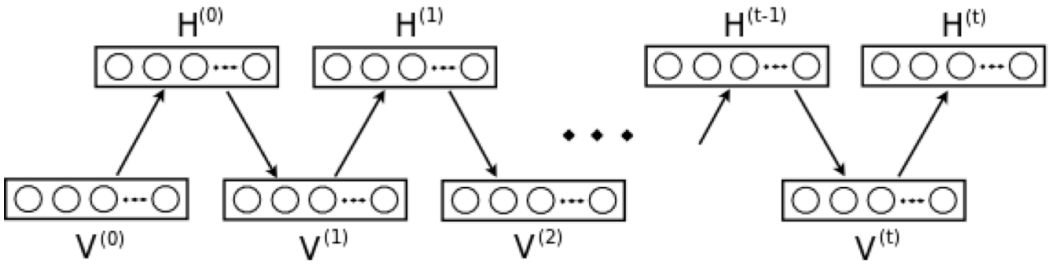

Figure 11. Block Gibbs sampling in an RBM. Each block in the scheme represents a layer of units, and directed arrows indicate how the sampling proceeds over time. As *t* goes to infinity, the values of visible and hidden units are guaranteed to accurately match samples from the joint distribution *P(v, h)*.

3.3 Hierarchical generative models

3.3.1 Deep neural networks

Restricted Boltzmann Machines can be used as basic blocks for building more complex network architectures, where the hidden variables of the generative model are hierarchically organized (Fig. 12B). The resulting architecture is referred to as a "deep network." The main intuition behind deep learning is that, by training a generative model at level l using as input the hidden causes discovered at level $l–1$, the network will progressively build more structured and abstract representations of the input data, which are invariant to most of the variations typically present in the training distribution, at the same time preserving as much as possible of the relevant information. Importantly, architectures with multiple processing levels efficiently structure the representation space by promoting features reuse: simple features extracted at lower levels can be successively combined to create more complex features, which will eventually unveil the main causal factors underlying the data distribution. 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 (Bengio, 2009). Moreover, adding a new layer to the architecture increases a lower bound on the log-likelihood of the generative model (Hinton et al., 2006), thus improving the overall representational capacity of the network. After learning of all layers, a deep network can be used as a generative model by reproducing the data when sampling from the model, that is by feeding the activations of the deepest layer all the way back to the input layer.

In this work we focus on deep architectures composed by a stack of RBMs, which are usually called *deep belief networks*. However, it should be noted that similar results can be obtained by stacking together single-layer autoencoders (Bengio & Lamblin, 2007). This approach has been successful in terms of machine learning benchmarks, but it is less appealing for cognitive modeling

purposes because learning is based on error backpropagation and it is not grounded in a sound probabilistic framework. Moreover, deep autoencoders are not used as generative models to produce predictions based on top–down signals.

3.3.2 Inference and learning in deep neural networks

Deep belief networks are built by stacking together several RBMs, which are learned in a greedy, layer-wise fashion as shown in Fig. 12. After the first RBM has been learned (lower part of Fig. 12A), the activities of its hidden units are used as input for a second RBM (higher part of Fig. 12A), with the aim of extracting higher-order correlations from the original data. The resulting composite generative model can be used to autonomously produce data patterns by first performing Gibbs sampling on the undirected, associative memory corresponding to the top two layers of the architecture and then exploiting the top-down, directed connections in a single backward pass (Fig. 12B). Conversely, when the units in the visible layer are clamped to a particular, observed data pattern we can sequentially infer the states of the two hidden layers by using the connections in a bottom-up fashion.

After a good generative model of the data has been learned, the whole network can be fine-tuned using a supervised criterion (Hinton et al., 2006). However, high-level features learned on a particular dataset might also be useful to represent information contained in examples that are not from the same distribution as the training distribution. In this case, instead of fine-tuning the whole system on a specific task, an arbitrary number of different classifiers can be trained directly on the top-level, abstract representations. This scenario can be described as a transfer learning problem (Pan & Yang, 2010), and it can be readily generalized to multi-task learning situations, where different labels on the same patterns are used for different tasks (Caruana, 1998). For example, high discriminative accuracy can be obtained even by a linear classifier applied

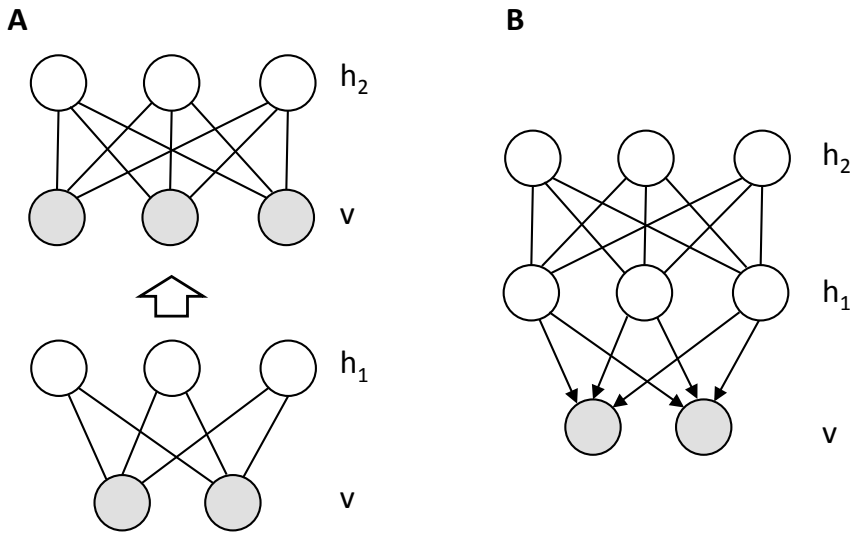

Figure 12. **(A)** A deep belief network with two hidden layers combines two RBMs that are learned in a greedy, layer-wise fashion, where the higher-level RBM is trained by using the hidden layer activities of the lower RBM as input data. **(B)** The resulting generative model is produced by stacking together the two RBMs. Adapted from Hinton (2007).

to the top-level internal representations of a deep network trained to only reconstruct the input patterns (Testolin et al., 2013).

3.3.3 Analyzing deep neural networks

In this section we briefly discuss some useful techniques that can be used to analyze deep neural networks, especially when they are used for modeling cognitive processes (Zorzi, Testolin, & Stoianov, 2013).

Supervised read-out

As we discussed above, an important feature of hierarchical generative models is that learning can be performed in a completely unsupervised fashion. However, combining generative learning with supervised tasks can be a useful way to better understand how information is extracted and encoded within the model hierarchy. Deep networks perform a non-linear projection of the feature vector at each hidden layer, gradually building increasingly more complex and

abstract representations of the data that eventually make explicit the latent causes of the sensory signal. This hierarchical organization suggests that a linear "read-out" of hidden unit representations should become increasingly more accurate as a function of layer depth. In this perspective, accuracy of linear read-out can be considered as a coarse measure of how well the relevant features are explicitly encoded at a given depth of the hierarchical generative model.

The linear read-out on internal representations can be easily implemented using another connectionist module, such as a linear perceptron, thereby preserving the biological plausibility of the model. The linear network can also be considered as a response module that supports a particular behavioral task, so that its responses can be assessed against the human data. For example, this approach has been adopted to simulate human behavior in a numerosity comparison task after training a deep network on thousands of images of sets of objects (Stoianov & Zorzi, 2012). The internal representations at the deepest layer provided the input to a linear network trained to decide whether the numerosity of the input image was larger or smaller than a reference number. Notably, the responses of this decision module were described by a psychometric function that was virtually identical to that of human adults, with the classic modulation by numerical ratio that is the signature of Weber's law for numbers.

Discovering learned representations

In the previous section we illustrated how it is possible to assess the quality of the internal representations learned at each layer of the hierarchy of a deep belief network by performing a discriminative task. However, this information is tied to a given classification task and is therefore limited in scope. Moreover, the supervised classifier operates on the pattern of activity over an entire hidden layer, that is a distributed representation encoding a variety of micro-features (Hinton et al., 1986) representing task-independent statistical regularities of the data. A very simple but informative approach to investigate the role of a particular unit in the network consists of visualizing its connection

weights using the original structure of the data (e.g., a bi-dimensional matrix if the network is learning a generative model of images). This is particularly intuitive for the first hidden layer, where the weight matrix defines how the visible units contribute to the activation of each hidden unit. We can therefore visualize the "receptive field" of each hidden unit by plotting the strength of its visible-to-hidden connections. The same principle can be applied to the deeper layers of the network, by combining their weight matrix with those of the lower layers. A straightforward way is to use a linear combination of the weight matrices, possibly imposing a threshold on the absolute values of the weights in order to select only strong connections. This allows to visualize the receptive field learned at a layer *k* as a weighted linear combination of the receptive fields learned at level *k-1* (H. Lee, Ekanadham, & Ng, 2008).

Sampling from the generative model

Up to this point, we only discussed methods that investigate bottom–up processing of sensory data. However, a deep belief network is a generative model, and it can be very useful to assess the top–down generation of sensory data, as well as the mixing of bottom–up and top–down signals during inference in ambiguous situations. In one scenario, we can provide to the model a noisy input pattern (e.g., randomly corrupted or partially occluded) and let the network find the most likely interpretation of the data under the generative model. In another scenario, we can study the generative capability of the network when the visible units are not clamped to an initial state, and the network is therefore let free to autonomously produce a sensory pattern through a completely top–down process. This generative process can be constrained to produce "class prototypes" by adding a multimodal RBM on the top of the network hierarchy (Hinton et al., 2006), which is jointly trained using two input sources, one containing the internal representation learned by the deep network and the other encoding the corresponding label. After learning, we can clamp the label units to a specific state and let the top RBM to settle to equilibrium, thereby recovering the internal representation of the given class.

The generative connections of the model can then be used to obtain a reconstruction of the visible layer, which can be thought of as the model's prototype for the corresponding abstract representation. An interesting, simpler variant of this top-down generation can be implemented by means of an inverse linear mapping (Zorzi et al., 2013). Instead of jointly training the top-level RBM using the internal representation of images and the corresponding class label, we can try to directly map the class label and the internal representation through a linear projection. This mapping is analogous to the read-out module discussed previously, but it works in the opposite direction.

Developmental trajectories in deep networks

As we explained in Section 3.3.2, the usual procedure to train deep neural networks involves an initial, greedy layer-wise pre-training, so that layer *n* is trained only after completing training of the *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. However, from a developmental perspective this training regimen is unrealistic, because cognitive functions are likely to be learned in an on-line setting, with weights updated after each example is observed. Moreover, training a hierarchical model in a greedy, layer-wise fashion does not allow to carefully investigate how more complex, abstract representations emerge throughout the developmental process. In order to analyze the developmental trajectories in deep network, we can exploit a progressive learning algorithm that allows joint training of all the layers in a deep network when only one input pattern is observed at each learning iteration (Zou, Testolin, & McClelland, *in preparation*). In the simulations presented in this work we did not investigate the development of cognitive abilities during learning, therefore we did not apply this progressive learning algorithm.

3.4 Temporal generative models

Up to this point, we have been concerned with learning generative models of a set of data patterns represented as high-dimensional vectors that are assumed to be independent from each other. However, in the case of temporal data input patterns appear in a precise, sequential order. A generative model should therefore consider not only the current observed evidence (i.e., the vector of visible units activations), but also the history provided by the previously presented items of the sequence. The aim is therefore to predict the probability distribution of an element of a sequence, possibly given other preceding elements as context. In the next section we briefly describe an interesting extension of restricted Boltzmann machines that can process temporal information, thereby allowing inference and learning over data vectors organized in coherent sequences.

3.4.1 The Recurrent temporal Restricted Boltzmann Machine

The Recurrent Temporal Restricted Boltzmann Machine (RTRBM; Sutskever, Hinton, & Taylor, 2008) extends the architecture of traditional RBMs by adding a set of delayed recurrent connections in the hidden layer, which are used to propagate information over time in order to keep track of past states of the system. This augmented network can be seen as a partially directed graphical model (see Fig. 13A), where some of the parameters are not free but are instead parameterized functions of conditioning random variables (i.e., the context). The internal representations created in the hidden layer can therefore implicitly encode distal temporal interactions, which can possibly span an arbitrary number of elements.

The joint distribution of a whole sequence of T pairs of visible and hidden variables induced by an RTRBM is defined as:

$$P(v_1^T, h_1^T) = P_0(v_1)P(h_1|v_1)\prod_{t=2}^{T} P(v_t|h_{t-1})P(h_t|v_t, h_{t-1}) \qquad (19)$$

where $P_0(v_1)P(h_1|v_1)$ specifies the distribution of the first pair of the sequence when no context is available. In this case, the probability distribution of visible units is not conditioned (there is no context), while the probability distribution of hidden units is conditioned to the state of visible units, which represents the current evidence. The conditional distributions for the successive elements $P(v_t|h_{t-1})P(h_t|v_t, h_{t-1})$ are computed step by step, conditioning the visible activations v_t on the previous hidden activations h_{t-1} (contextual information) and conditioning the hidden activations h_t on both the previous hidden activations h_{t-1} and current visible activations v_t. The joint distribution of visible and hidden variables for the whole sequence is given by the product of all these conditional distributions. During the processing of a sequence, in order to predict the successive visible-layer activations v_{t+1} we first infer the hidden state h_t given the current element of the sequence v_t and the previous hidden state h_{t-1} using a mean field approximation (Peterson & Anderson, 1987):

$$P(h_t|v_t, h_{t-1}) = \sigma(VH^T v_t + b_h + HHh_{t-1}) \qquad (20)$$

where VH is the matrix of visible-to-hidden connections, b_h is the static hidden unit bias, HH is the matrix of the additional hidden-to-hidden connections and σ is the usual logistic activation function. The term HHh_{t-1} represents the dynamic hidden bias, which is used to propagate contextual information over time. Once the conditional hidden activations h_t have been inferred, we can generate a prediction of v_{t+1} by starting from a random binary state of the pair (v_{t+1}, h'_{t+1}) and performing iterative block Gibbs sampling until convergence, in which the activation of the hidden units also accounts for the dynamic bias HHh_t (see Fig. 13B). If we do not condition the model on a given context, we can let the network generate a sequence by starting from an initial learned bias and sequentially generating visible and hidden states.

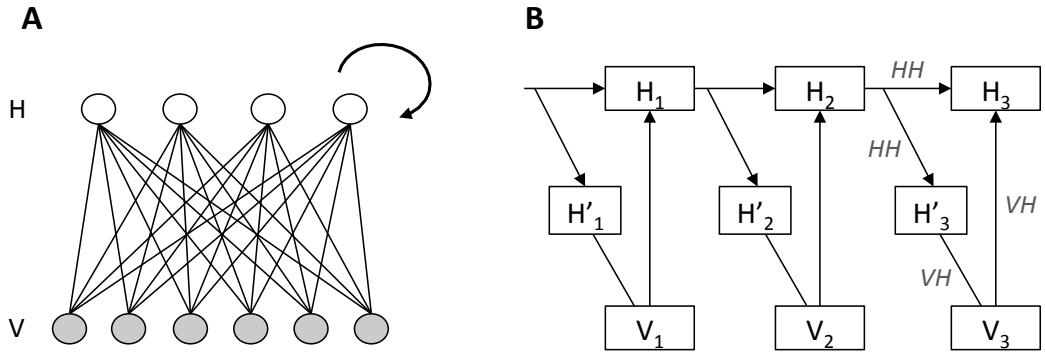

Figure 13. **(A)** Graphical representation of an RTRBM, which extends the basic RBM by using an additional set of hidden-to-hidden delayed connections (curved arrow) **(B)** Schematic diagram of the RTRBM processing a 3-elements sequence (from left to right). Note that there is one hidden layer with real-valued activations (H) that is used for inference and one with binary activations (H') that is used during the generative phase. The weights parameterization is reported for the last element, where visible-to-hidden connections are indicated with *VH* and hidden-to-hidden connections are indicated with *HH*. Adapted from Sutskever et al. (2008).

As for RBMs, RTRBMs can be efficiently trained in an unsupervised fashion by using contrastive divergence to compute the local gradient of the prediction error for each element of a sequence. The gradients are then propagated to previous time steps using backpropagation through time (Werbos, 1990).

It should be noted that the network processes temporal information in a strictly sequential way, one element at a time and only using the last hidden activations as context. Thus, in contrast to other probabilistic language models that introduce additional temporal connections between preceding elements and the hidden state (e.g., Mnih & Hinton, 2007), the RTRBM only exploits local temporal interactions, which can nonetheless allow to encode in the hidden layer an arbitrary number of preceding elements as context. In the original work (Sutskever et al., 2008), the network learned smooth dynamics that described the physical behavior of bouncing balls in a constrained space. In the simulations presented in Section 4.5 we address the intriguing question of whether the same network can also learn discrete dynamics, like the transition probabilities describing the orthographic structure of letter sequences.

4 Experiments

4.1 Parallel implementation on graphic processors

Artificial neural networks can be easily implemented on parallel computing machines thanks to their intrinsically parallel and distributed nature, which allows to significantly reduce training times even in the presence of large-scale models, where a great number of parameters (e.g., millions of connection weights) must be fit to the training data. Due to the high computational demand of deep learning algorithms, we therefore first created an efficient implementation of deep belief networks that can be executed on high-performance, parallel computing hardware.

The proposed implementation exploits graphic processing units (GPUs) to distribute the computation over many processors equipped with a shared, large-bandwidth memory. The enormous computing power of GPUs derives from their internal architecture, which exploits a great number of simple cores operating in parallel. This hardware design is well suited to efficiently process graphic information (e.g., real-time rendering of visual scenes by mapping textures and applying lightening to geometric shapes), which is usually represented using matrices of items that can be manipulated element-wise. The basic idea that allows to exploit GPUs for scientific computing is that here too we often have to apply simple functions to a large number of elements at the same time (Owens & Houston, 2008). Moreover, the use of GPUs for scientific computing greatly increased after the introduction of CUDA, which is an improved parallel computing platform created by NVIDIA that gives developers direct access to many powerful programming routines used to directly control the processing flow inside the graphic hardware (Nickolls et al., 2008).

Deep learning algorithms largely involve simple matrix manipulations and are therefore well suited to be implemented on GPUs. Machine learning researchers are already adopting this powerful technology for training deep

networks, obtaining impressive speed-ups that allow to scale-up the size of both networks and training databases (e.g., Ciresan, Meier, Gambardella, & Schmidhuber, 2010; Dean et al., 2012; Krizhevsky et al., 2012; Raina et al., 2009). At the same time, these advanced parallel implementations require to master a certain level of technical expertise, making them less accessible to cognitive modelers. Here we show how simulations of deep neural networks can be easily performed even on a desktop PC equipped with an entry-level GPU without any specific programing effort, thanks to the use of high-level programming routines available in MATLAB or Python (Testolin et al., 2013). Moreover, we show that our parallel implementation can even outperform a small high-performance computing cluster in terms of learning time and with no loss of learning quality. The complete source code for different software platforms is publicly available for download[1].

4.1.1 Implementation and hardware details

The recommended way to train RBMs is to split the entire dataset into smaller, non-overlapping subsets, called *mini-batches* (Hinton, 2010a). Instead of iteratively updating the network weights with the gradient computed on each training pattern ("on-line learning") or rather updating with the average gradient computed across the entire dataset ("off-line learning"), the gradient in mini-batch learning is averaged over the patterns of the mini-batch. This improves convergence and learning speed by both varying and smoothing the learning gradient (Wilson & Martinez, 2003). By adopting a mini-batch learning scheme the implementation on graphic processors is straightforward, because we can directly feed the GPU processors with the entire matrix containing the patterns belonging to each mini-batch. All we need to do is to load the training dataset into the GPU's memory and to adapt the source code to specify which operations have to be performed on the graphic card. We exploited high-level wrappers of CUDA available in MATLAB via the Parallel-Computing Toolbox

[1] http://ccnl.psy.unipd.it/research/deeplearning

(Sharma & Martin, 2008) and in Python via the Gnumpy module (Tieleman, 2010). The use of these high-level functions greatly simplified the parallelization, which only required the use of *gpu array* data types instead of standard arrays (i.e., the processing load was automatically distributed among the graphic processors, thereby making parallelization transparent).

Both the MATLAB and the Python parallel implementations were tested on two different graphic cards. One was an NVIDIA GeForce GTX 460 (Fermi architecture) equipped with 336 CUDA cores (1.35 GHz) and 1 GB of DDR5 dedicated memory. The other was an NVIDIA GeForce GTX 690 (Kepler architecture), for a total number of 1536 CUDA cores (1.41 GHz) and 2 GB of DDR5 memory. The cluster implementation used for comparison (De Filippo De Grazia, Stoianov, & Zorzi, 2012) was executed on a HP distributed computing cluster composed of seven nodes, each equipped with quad-core or hexa-core processors (2.27 GHz) and 32 GB of RAM. Overall, there were 60 cores, interconnected with an Infiniband technology network, and the parallelization was implemented using MPI synchronization routines. As a baseline, we also collected running times on a PC workstation equipped with an Intel Q6600 quad-core CPU (2.40 GHz).

4.1.2 Dataset and network architecture

As a benchmark for our parallel implementation we used a classic vision problem, which consists in recognizing handwritten digits contained in the popular MNIST dataset (LeCun & Cortes, 1998; some samples are reported in Fig. 14A). The MNIST dataset contains 60,000 training images and 10,000 test images representing handwritten digits encoded as 28x28 pixel gray-level images, size-normalized, mass-centered, and manually classified into ten digit classes (from 0 to 9). Notably, is has been shown that deep networks reach state-of-the-art accuracy on this challenging machine learning problem (Hinton & Salakhutdinov, 2006).

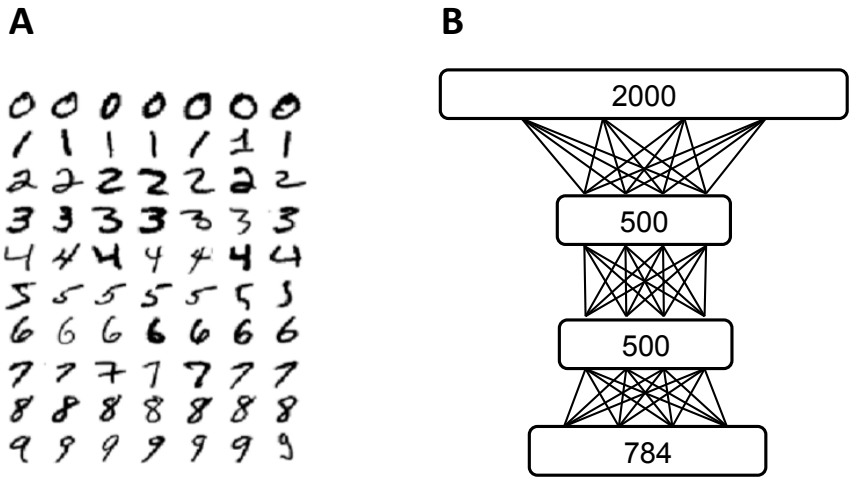

Figure 14. **(A)** Samples from the MNIST dataset. **(B)** Network architecture.

Learning parameters were selected following the original study of Hinton and Salakhutdinov (2006). In particular, we trained a deep network with three hidden layers composed by 500 (H1), 500 (H2) and 2000 (H3) units, for a total of about 1.6 million connections (see Fig. 14B for a graphical representation of the resulting architecture). Layer-wise unsupervised learning was performed with one step contrastive divergence for 50 epochs. The final quality of the model was assessed by training a linear classifier to recognize the test patterns using the top-level, hidden representation as input. As a further control, we also fine-tuned the whole network using backpropagation in order to compare the final quality of the model with previously published, state-of-the-art results.

The parallelization becomes more effective as the size of the mini-batches increases, because more processing resources can be allocated. However, with larger mini-batches learning is more prone to get stuck in local minima, thus leading to a deterioration of the final quality of the model. We investigated these effects by systematically varying the mini-batch size, which in different runs was set to 125, 250, 500, 1000, 2000, 5000, and 7500 patterns. Running times and classification errors were collected and averaged over 10 different networks for each type of parallel implementation.

4.1.3 Results and discussion

A comparison between learning times for the various implementations is shown in Fig. 15A. The most impressive result is the substantially lower times required by GPUs: on low and medium mini-batch sizes (e.g., 125, 250, 500, and 1000), even the cheap, entry-level GTX 460 outperformed the cluster by one order of magnitude. The high-performance GTX 690 further improved the result (as highlighted in Fig. 15B), requiring a learning time that was half of the cluster also on the largest mini-batch size. Notably, learning times on GPUs were between 11 and 45 times faster than on the quad-core workstation. The results show that the parallelization on the computer cluster is competitive only when using large mini-batches, that is when more cores are independently processing distinct subsets of training patterns. Indeed, on very large mini-batches, the cluster performance approaches that of GPUs. Unfortunately, this also causes a decrease of learning quality, which is reflected by the lower classification accuracy reached by models trained using large mini-batch sizes, as we will discuss in the following.

The learning quality was first evaluated by measuring the classification accuracy obtained by a linear classifier trained on the top-level (H3) hidden layer representations of the input data extracted after the unsupervised phase. As shown in Fig. 15C, the GPU and the cluster implementations obtained similar results in terms of model quality. More precisely, analysis of variance on the accuracy data (within-subject factor: mini-batch size, $n = 7$; between subject-factor: parallel implementation type, $n = 2$) revealed a significant effect of mini-batch size [$F(6,108) = 598.5$, $p < 0.001$] but no effect of implementation type [$F(1,18) = 1.3$] or their interaction [$F(6,108) = 2.0$]. Linear regression analysis with mini-batch size as a predictor (on a log scale) confirmed that the classification error increased along with mini-batch size (*slope* = 0.77, $r^2 = 0.99$), thus showing that a reduced mini-batch size is critical to learn a good model.

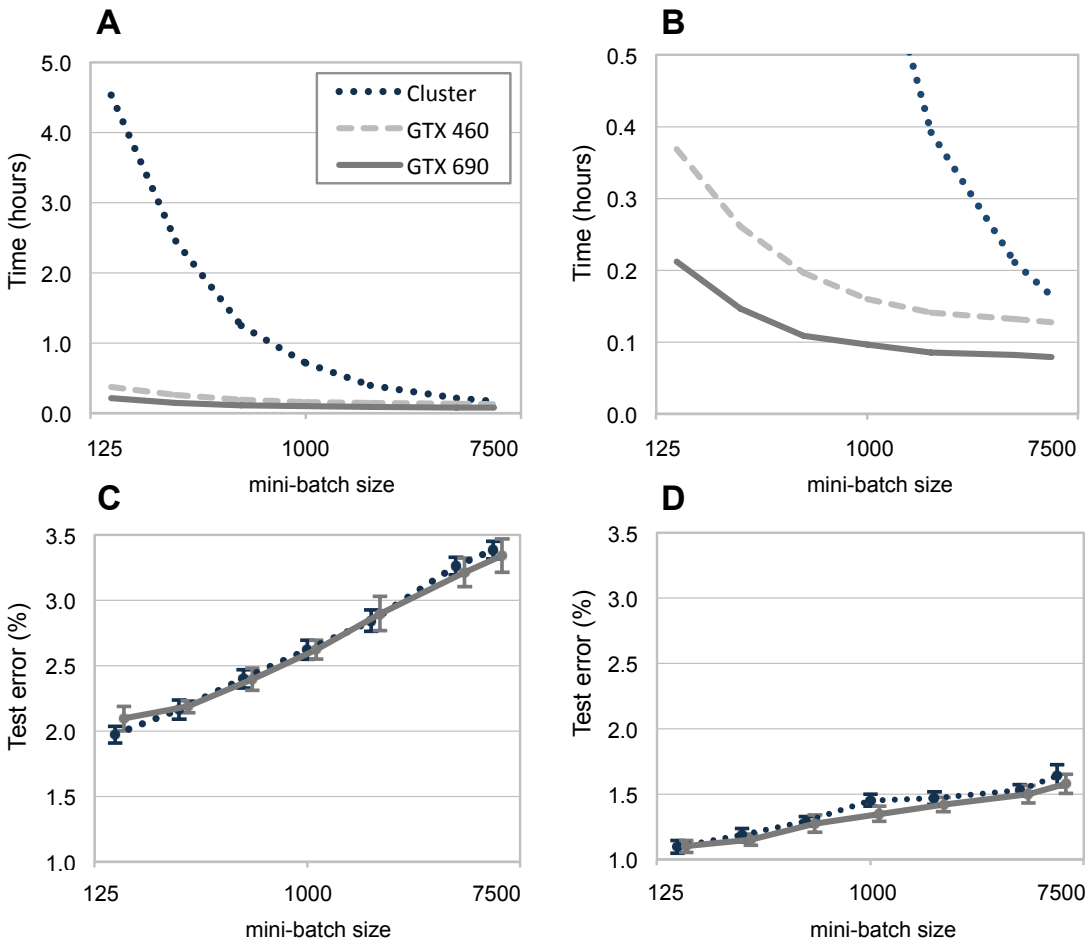

Figure 15. **(A)** Unsupervised learning times decreases with mini-batch size for all parallel implementations: the greater the number of patterns simultaneously processed, the more the computational resources involved. **(B)** Zoom-in of learning times highlighting the additional speed-up of the GTX 690 card. **(C)** Quality of learning on the cluster and on the GTX 690 implementations, measured as a linear read-out misclassification. **(D)** Quality of learning after fine-tuning the entire network with error backpropagation.

As shown in Fig. 15D, both implementations obtained comparable classification performance also after an additional fine-tuning phase of the whole network. In this case the overall performance slightly improved, leading to results comparable with the published state-of-the-art (Hinton & Salakhutdinov, 2006). Also here, a linear regression analysis demonstrated that the

classification error after fine-tuning increases with mini-batch size (*slope* = 0.28, r^2 = 0.98), though in this case the effect of mini-batch size is less pronounced.

In summary, both the cluster and the GPU implementations exhibited an exponential reduction of learning times when using mini-batches of increasing size. However, learning large mini-batches comes at the cost of a lower classification accuracy, which indexes a lower quality of the learned models. These results highlight a trade-off between learning times and learning quality and, accordingly, a clear advantage in using the parallel implementation based on GPUs. It should be mentioned that one bottleneck in GPU computing is the relatively slow transfer of data between the central memory (RAM) and the GPU. We optimized the algorithm by using single- instead of double-precision data types, which allowed to load the entire training dataset (200 MB) into the GPU and perform the entire computation with minimal RAM-GPU communication.

A final aspect that deserves attention regards the economic cost of the hardware required by the parallel implementations considered in our comparison. The price of an entry-level graphic card is about 100€, while the cost of a 60-nodes computer cluster exceeds 30,000€. Moreover, the Python solution provides a freeware implementation on graphic cards. In conclusion, parallelization of deep belief networks on GPUs using high-level languages can bring medium-scale simulations on a desktop computer at a very affordable cost and with no time investment on acquiring parallel programing skills.

4.2 Early vision: learning patches of natural images

The GPU implementation presented above allows to train restricted Boltzmann machines (and their hierarchical extensions) in a very efficient way, thereby opening the possibility to test these generative models on complex datasets composed by thousands of training patterns. Here we applied a single-layer RBM on a very large dataset containing patches of natural images, in order to study what type of features can be extracted by means of generative unsupervised learning.

During the last decades, the statistical structure of natural images has been modeled using a variety of probabilistic approaches (Hyvärinen, Hurri, & Hoyer, 2009), which have shown able to extract many types of visual features resembling those found in biological visual systems. Thanks to their flexibility, learning models thus constitute a powerful alternative to hand-coded models, which instead represent visual information using a set of carefully engineered features (e.g., Riesenhuber & Poggio, 1999). The basic idea behind these probabilistic approaches is that images projected on the retina are highly redundant, that is, the value of each single "pixel" is highly correlated to the value of many other neighboring pixels. This phenomenon is due to the ubiquitous presence of rigid spatial structures in natural scenes, which usually contain well-defined objects (e.g., trees, rocks, animals, etc.) displaced on well-shaped landscapes (e.g., mountains, forests, fields, etc.). Due to the limited representation and communication capacity of neural circuits, evolution has shaped nervous systems in order to efficiently code sensory signals by exploiting these redundancies (Barlow, 1961). For example, if a consistent part of the incoming visual signal is predictable (i.e., redundant), it can be efficiently encoded by using a set of maximally independent statistical components (ICA; Bell & Sejnowski, 1997), or by a set of components with a sparsity constraint (Olshausen & Field, 1996), leading to features having a remarkable affinity to the receptive fields of neurons in the primary visual cortex.

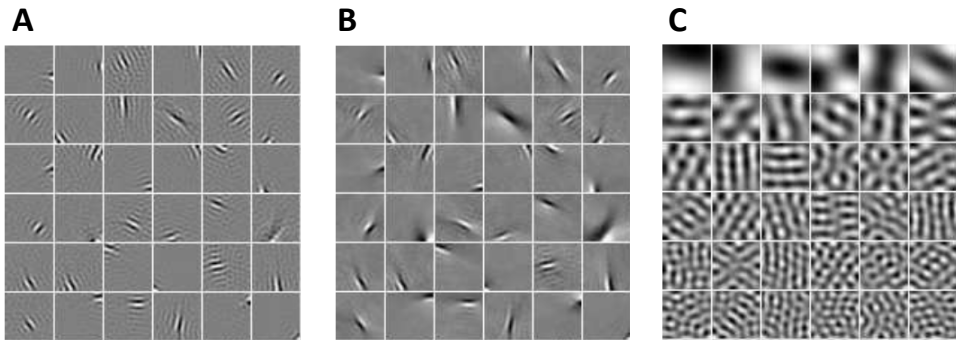

Figure 16. Visual features extracted from natural images using **(A)** Independent Component Analysis [ICA] **(B)** Sparse coding and **(C)** Principal Component Analysis [PCA].

In particular, a set of features extracted by ICA and sparse coding are shown, respectively, in panel A and B of Fig. 16. On the contrary, other simpler types of statistics do not lead to this type of features, as shown by the non-local filters corresponding to the principal components of a natural scenes dataset (Fig. 16C). In this section we explore the type of features that can be learned by a generative model implemented in a restricted Boltzmann machine. Interestingly, it will turn out that the statistical features extracted by an RBM can also be very useful not only to represent natural scenes, but also to efficiently encode other types of visual structure, as we will show in Section 4.3.4.

4.2.1 Dataset and network architecture

The RBM was trained to accurately reconstruct a set of patches extracted from a natural image dataset (M. Brown, Hua, & Winder, 2011) that contained gray-scale pictures representing the Yosemite park, the Liberty state and the Notre Dame cathedral. The photographs were taken from several different viewpoints (see Fig. 17A for some examples). Though it might seem counterintuitive to consider human-made artifacts (such as monuments and buildings) to be part of natural scenes, it has been observed that all these types of spatial structures usually give rise to similar statistical features (Hyvärinen et al., 2009).

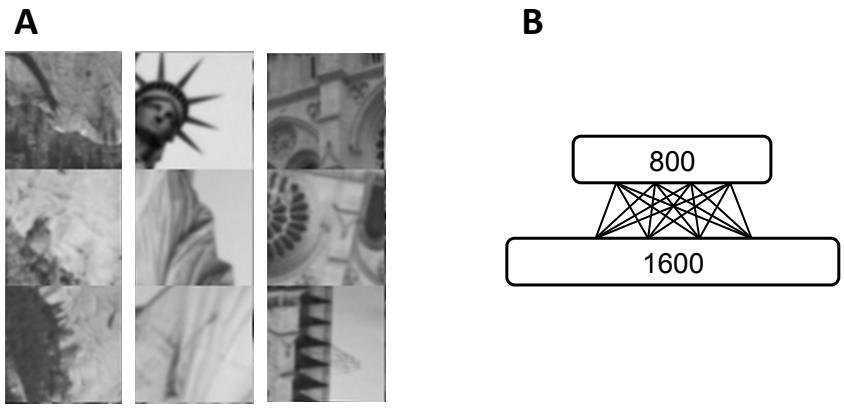

Figure 17. **(A)** Samples from the natural images dataset containing three subjects: the Yosemite national park (first column) the Liberty state (second column) and the Notre Dame cathedral (third column). **(B)** Network architecture.

Each patch was created by randomly placing a 40x40 pixel window on the original image and by successively rescaling the pixel values to the interval [0, 1]. In order to increase variability, each patch was also randomly rotated of 90°. The final patches dataset contained a total of 153,600 patterns.

The visible layer of the RBM had a total of 1600 units (40x40), while the size of the hidden layer (H1) was varied between 500 and 1500 units (the network architecture is represented in Fig. 17B). It appears that the size of the hidden layer does not qualitatively change the type of features extracted, and in the following we will refer to an RBM having 800 hidden units. Learning was performed for 200 epochs with a learning rate of 0.03, a momentum coefficient of 0.8 and a decay factor of 0.0001. Patterns were processed according to a mini-batch scheme, with size 100.

Pre-processing stage: image whitening

In order to simulate the type of processing operated by the retina and the geniculate nucleus of the thalamus, we first applied a *1/f* whitening algorithm to the patches dataset (Simoncelli & Olshausen, 2001). Basically, we designed a filter in the frequency domain that would flatten the spectrum of a natural image (on average). Since the power spectrum of natural images tends to fall as $1/f^2$, then the amplitude spectrum will fall as $1/f$.

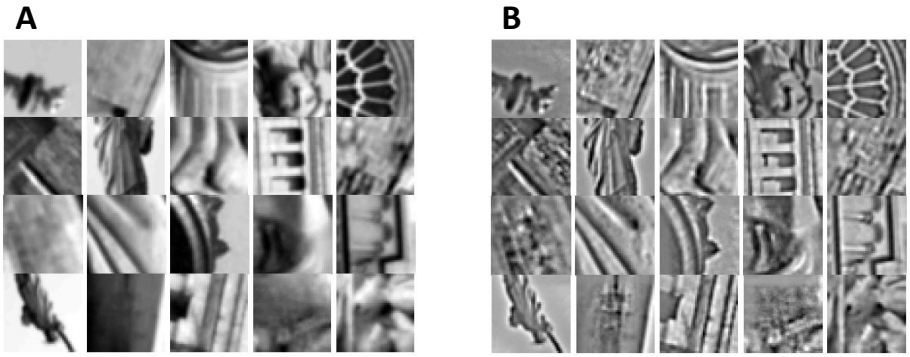

Figure 18. **(A)** Samples of natural image patches and **(B)** their corresponding whitened version.

We designed the amplitude spectrum of the filter so that it rises linearly with frequency, to compensate for the *1/f* amplitude spectrum of natural images. We then multiplied the image matrix by a circular, Gaussian filter in order to obtain a center-surround filter. Finally, local contrast normalization was obtained by dividing the value of each pixel by the standard deviation of the total activity of its neighborhood, using a Gaussian neighborhood with a diameter of 20 pixels. A set of patches with the corresponding whitened versions is reported in Fig. 18.

4.2.2 Results and discussion

After learning, we inspected the nature of the features extracted by the network by plotting the receptive fields of the hidden units using the procedure described in Section 3.3.3. As shown in Fig. 19, the RBM extracted a set of features similar to those obtained by other type of probabilistic models, such as ICA or sparse coding. In particular, some hidden units encoded some forms of Gabor filters with different spatial orientation and frequency, which can be effectively used to detect edges and borders in the visual scene. However, it seems that the network also extracted other types of features, which are not usually found when applying other types of probabilistic models to natural images. In particular, as highlighted in Fig. 19B, many hidden units encoded some form of sharp, elongated Gabor filters that possibly span the whole receptive field.

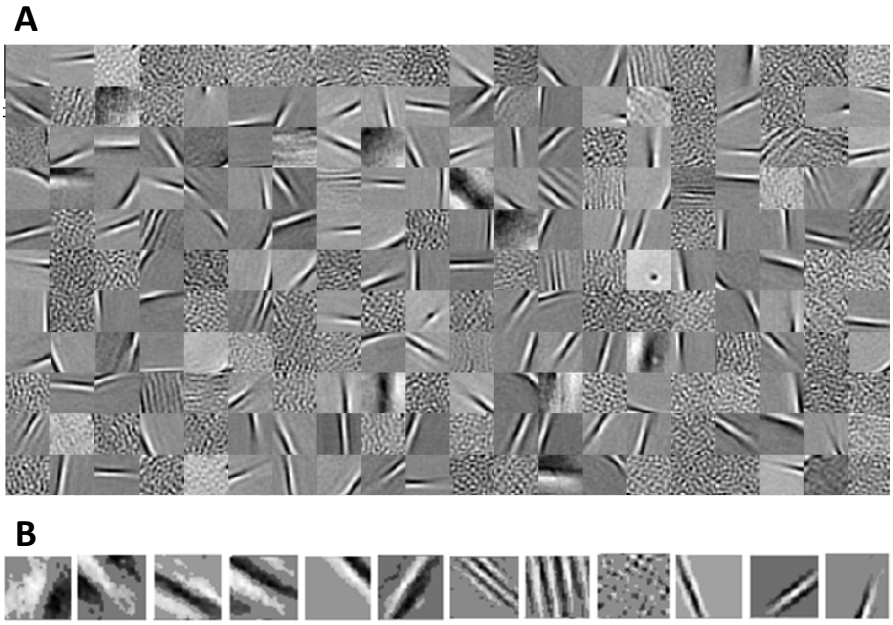

Figure 19. **(A)** A subset of receptive fields developed by the hidden units of a single-layer RBM trained to generate patches of natural images. **(B)** Some selected features of the model, some of which resemble ridgelets and gratings.

These features can be better described as "ridgelets", which differ from wavelets (e.g., Gabor filters) because they are constant along a hyperplane, which corresponds to a straight line in our two dimensional case. It has been shown that ridgelets have some interesting properties that make them particularly suited to compactly represent geometric structure and therefore process visual information (Candès & Donoho, 1999). Moreover, other hidden units learned to encode more complex visual structure, such as gratings with different spatial orientation and frequency. This result is some way unexpected and certainly requires a more careful investigation. However, it is interesting to note that these unexpected types of features (e.g., ridgelets and gratings) also emerge by applying a recent variant of sparse coding algorithms, where the representational space is made highly overcomplete by using a very large number of components (Olshausen, 2013).

In conclusion, our results show that learning a generative model of natural patches with a single-layer RBM can produce the same type of low-level visual features observed in the primary visual cortex and that are usually

extracted by applying other types of probabilistic models to natural images. Moreover, the network also learned some other types of features, such as ridgelets and gratings, which are also found when applying highly overcomplete sparse codes to natural images.

These types of spatial structures are useful to efficiently encode the statistical information contained in natural scenes, and they might constitute the basis for encoding more complex spatial structures in the successive levels of a processing hierarchy. In particular, these types of features could be extremely useful to also encode other type of visual patterns, for example those found in written symbols, which have likely be culturally selected to resemble the type of structures commonly found in natural environments (Changizi et al., 2006). We will further explore this hypothesis in Section 4.3.4.

4.3 Printed Latin letter recognition

As a starting point to model orthographic processing, we first studied recognition of isolated characters belonging to the Latin alphabet. To this aim, we created a synthetic dataset composed by a variety of printed letters, which were presented to a deep network in a completely unsupervised way.

Printed letters have a series of advantages compared to the handwritten patterns used in Section 4.1 to test our GPU implementation. Indeed, printed patterns have a more defined shape, with a better resolution and without the presence of noise or other confounds caused by different handwriting styles. This ensures a better control of the experimental conditions, because stimuli can be accurately grouped according to their visual properties. However, as we will discuss in the following, this more controlled setting also reduces the variability of the training dataset, which might cause problems when learning a hierarchy of visual features in a deep neural network.

4.3.1 Dataset and network architecture

In order to create a large-scale dataset of visual images containing printed characters, we implemented an automatic MATLAB script that exploited the "getframe" function to create gray-scale bitmaps of the 26 Latin letters. Pattern variability was obtained by combining different factors contributing to the visual appearance of each letter. In particular, the script generated letters according to 14 different fonts, half of which belonged to the Serif typeface (left column of Fig. 20) and half of which belonged to the Sans Serif typeface (right column of Fig. 20). All letters were printed uppercase with 5 different sizes: 22, 24, 26, 28 and 30 pixels. Each pattern was also encoded using 2 different weights (**bold** and not bold) and 2 different styles (*italic* and not italic).

Figure 20. Font types used to populate the printed Latin letters dataset.

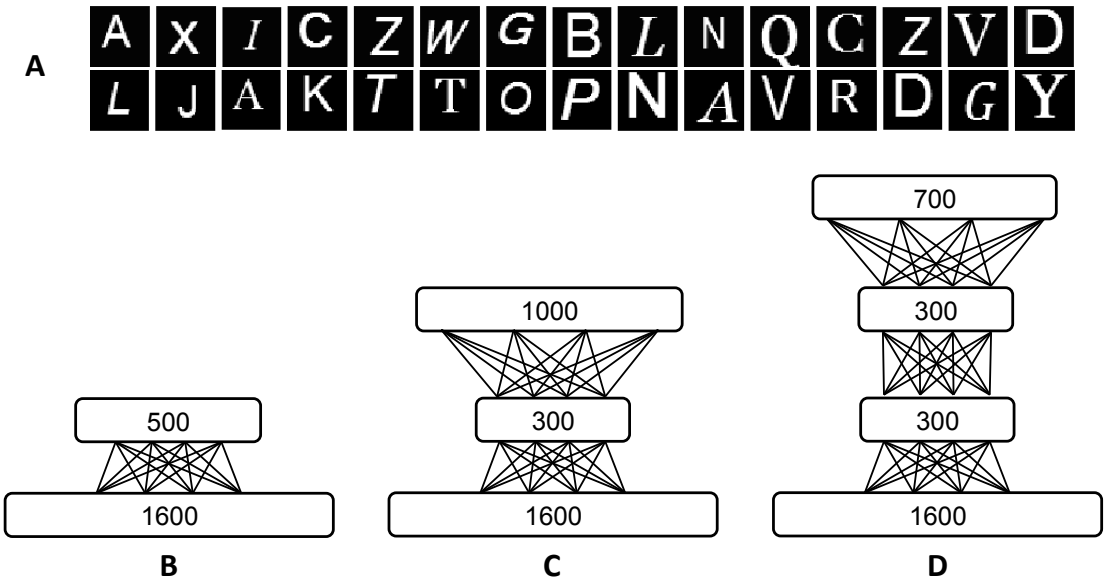

Figure 21. **(A)** Samples from the printed Latin letters dataset. **(B,C,D)** Considered network architectures.

Each pattern was printed in white on a 40x40 pixels black square, and its position was varied across the visual field according to a combination of both an x-axis offset (5 possible shifts) and a y-axis offset (5 possible shifts). Each shift was performed only up to the point where the complete pattern was still present in the visual field (i.e., the script was prevented to generate out-of-border offsets). By combining all the factors of variation listed above, in total there were 14 x 5 x 2 x 2 x 5 x 5 = 7000 possible versions of the same letter, which produced a dataset containing 182,000 training patterns (some samples are shown in Fig. 21A).

The visible layer of each tested model was composed of 1600 units (40x40). We tested different types of architectures: a single-layer RBM with 500 hidden units (Fig. 21B), a two-layers deep network with 300 (H1) and 1000 (H2) hidden units (Fig. 21C), and a three-layers deep network with 300 (H1), 300 (H2) and 700 hidden units (H3), represented in Fig. 21D. The rationale behind the choice of the sizes of the hidden layers was to approximately keep constant the total number of connections in all the networks, in order to match the different architectures according to model complexity. In all cases, learning was

performed for 60 epochs with a learning rate of 0.1, a momentum coefficient of 0.8 and a decay factor of 0.0004. Patterns were processed according to a mini-batch scheme, with size 100.

4.3.2 Simulation 1: learning letters from scratch

In the first series of simulations, the synthetic dataset was randomly split into a training set containing 75% of the patterns (i.e., 136,500 examples) and a test set containing the remaining 25% of the patterns (i.e., 45,500 examples). After a first phase of unsupervised learning, each network was tested on a classification task using the read-out method described in Section 3.3.3. It should be noted that the recognition results presented below seem to confirm that the training and test sets had a similar distribution (i.e., after learning each model reached a performance on a discrimination task over the test set that approximately matched the performance on the training set). The quality of the three-layer network was also assessed by inspecting the type of features extracted at different levels of the hierarchy and by letting the model to autonomously produce class prototypes through its top-down generative connections, using the methods described in Section 3.3.3.

4.3.3 Results and discussion

As anticipated above, all the models exhibited a good generalization when a read-out module was trained in their top-level, internal representations. Classification accuracy for the different network architectures is reported in Fig. 22A. As a baseline, we also computed the accuracy of the read-out module trained directly on the raw patterns.

Results show that a single-layer RBM already obtains fairly high recognition accuracy, approaching 95% on the test set. However, better results are obtained by adding more hidden layers to the hierarchy, with both the two- and three-layers networks reaching 98% on their deepest layer.

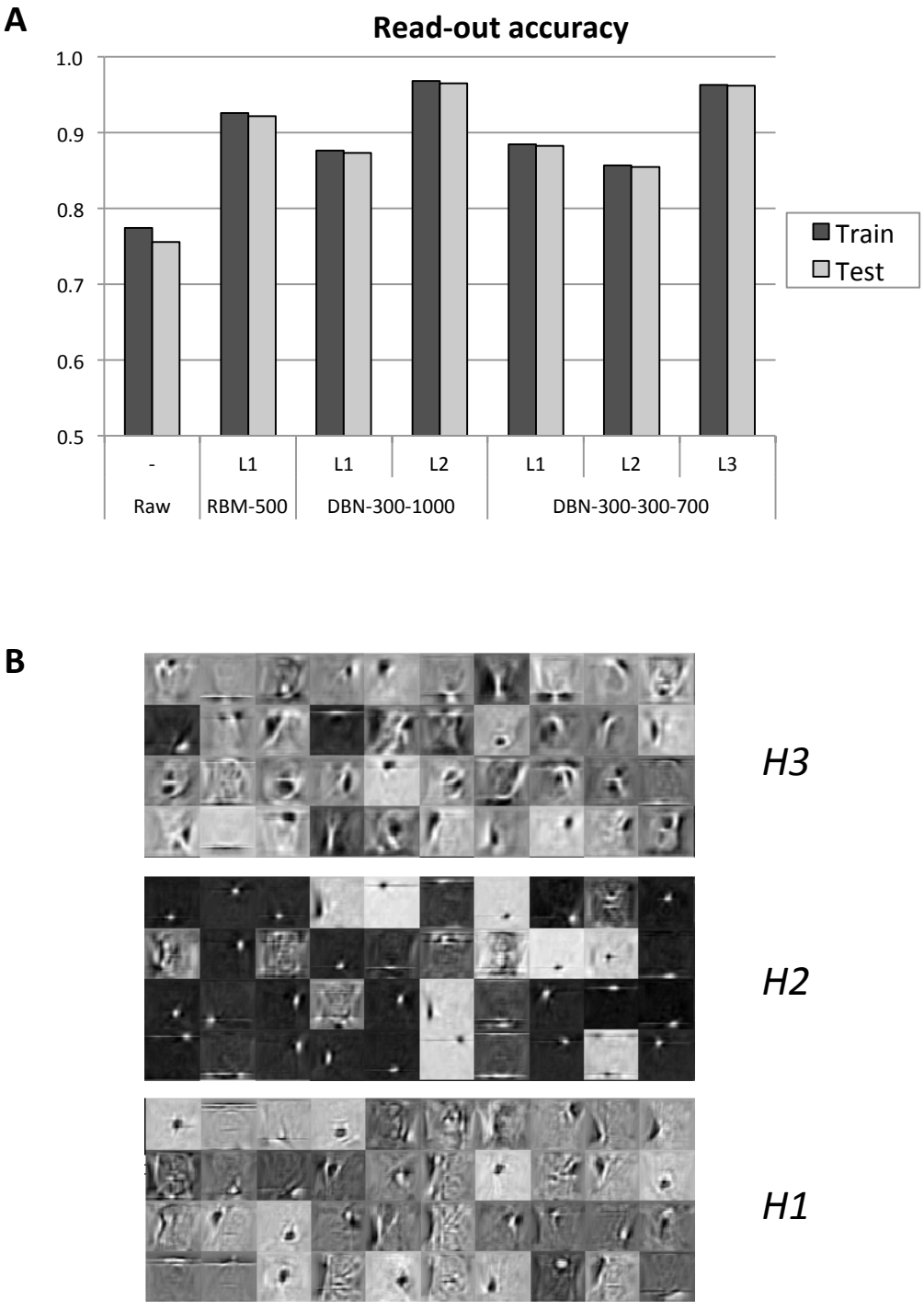

Figure 22. **(A)** Read-out accuracy on different levels of representations. **(B)** Receptive fields corresponding to units in different levels of representations learned by the three-layers deep network.

This trend suggests that a hierarchical architecture is particularly suited to represent the spatial structure of Latin printed letters. However, the analysis on the features learned at different levels of the hierarchy suggests that the three-layers network did not learn an optimal hierarchical generative model of the patterns, as highlighted by the receptive fields reported in Fig. 22B. In particular, low-level features encoded in the first layer do not always exhibit spatial selectivity, and even if it seems that features get more complex as we move up in the hierarchy, they do not closely resemble those usually extracted by deep networks trained on datasets containing visual structure.

Despite the sub-optimal feature hierarchy found by the receptive fields analysis, prototypes generation with top-down connections produced very precise results. In particular, as shown in Fig. 23, the network was able to generate high-quality visual patterns, which can even be produced according to specific, high-level biases. For example, when the inverse linear mapping is trained using all the available data patterns, prototypes have a fairly general visual appearance, and critical features corresponding to line endings and other salient letter parts are often omitted (see Fig. 23A). However, if we only select a specific subset of patterns to train the inverse mapping, we can bias the generation toward more specific prototypes. For example, Fig. 23B shows prototypes obtained using only small, bold letters encoded with the Arial font type. Fig. 23C instead reports prototypes obtained using only large, italic letters encoded with the Times New Roman font type. It can be observed that the network is able to generate fine-grained visual features, for example those corresponding to the “serifs” in the Times New Roman font.

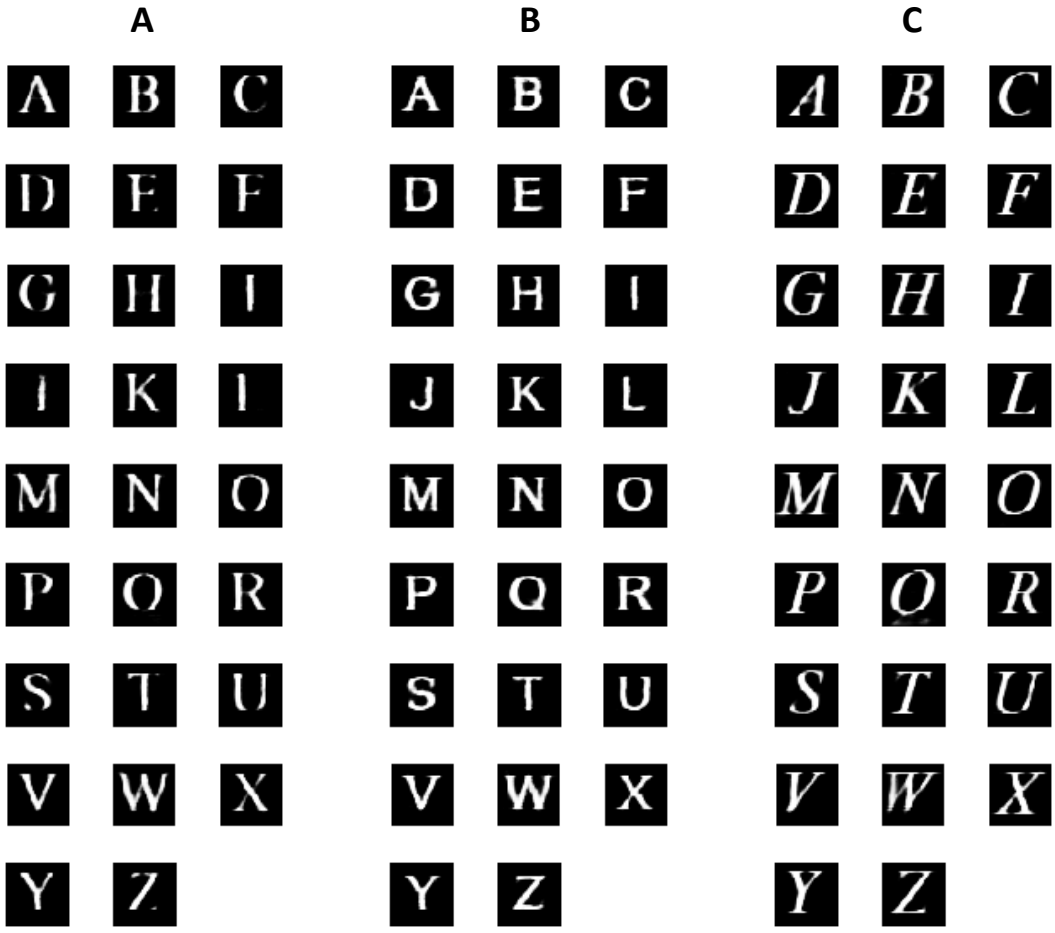

Figure 23. Prototypes generation obtained by training the inverse linear mapping with **(A)** all training patterns; **(B)** training patterns corresponding to small, bold letters printed in Arial font; **(C)** training patterns corresponding to large, italic letters printed in Times New Roman font.

4.3.4 Simulation 2: exploiting natural statistics

In a second series of simulations, we tested the hypothesis that natural scenes and written patterns share a common statistical structure, which can be effectively captured by a generative model implemented in a RBM. To this aim, we used the network trained on the patches of natural images described in Section 4.2 to create a more expressive representation of the raw letter patterns, which might support a better read-out performance. More precisely, we first applied the same whitening algorithm described in Section 4.2.1 to the whole letter dataset (some samples of whitened letters are reported in Fig. 24A). We then used the weight matrix of the network trained on natural images to compute an internal representation of the input letters, which was successively read-out by the linear classifier with the aim of recognizing the original pattern (see Fig. 24B for a schematic representation of this processing architecture). In this way, the read-out module was not trained on internal representations obtained by unsupervised learning of written patterns (as done in the previous experiments), but instead the read-out was performed directly on a more general representation derived from the statistics of natural images.

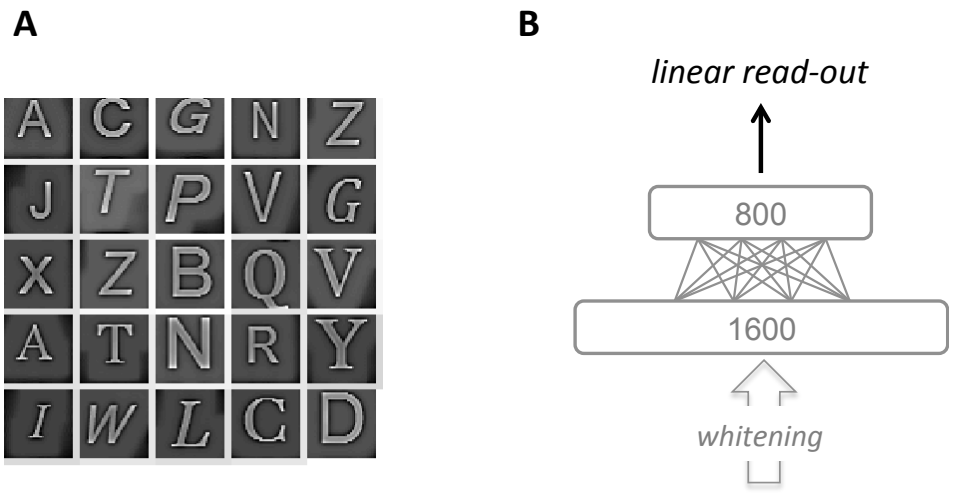

Figure 24. **(A)** Samples of whitened printed Latin letters. **(B)** Processing architecture for the task of reading-out letters from natural images representations. Note that the connection weights of the RBM have not been trained on printed letters, but only on patches of natural images.

4.3.5 Results and discussion

Quite surprisingly, the letter read-out accuracy on the representations created by unsupervised learning of natural images was extremely high, approaching 99%. This result might be partially due to the fact that the letter patterns have been pre-processed using a whitening algorithm, which might have improved the perceivability of the patterns. Indeed, the read-out accuracy on the raw, whitened patterns is slightly higher than that obtained on the raw, unwhitened patterns (see Fig. 25). However, the extremely high recognition accuracy obtained using the features learned from natural images suggests that the rich statistical information extracted from natural scenes can be readily applied to represent also to other types of spatial structures, for example those defining written patterns. Indeed, such expressive representation was not extracted by learning a generative model of written patterns, but anyway supported a very accurate letter classification. Notably, it should be mentioned that certain statistical features extracted from natural images have shown useful also in other tasks (e.g., Cottrell, 2008; Kanan & Cottrell, 2010), therefore it would be interesting to test the representational capability of the features extracted by the RBM also on other challenging recognition tasks.

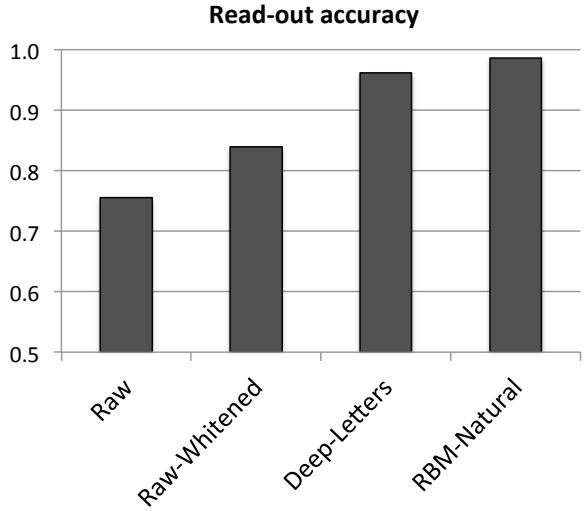

Figure 25. Classification accuracy obtained by a read-out module applied on different processing stages and representations.

4.4 Handwritten Farsi character recognition[2]

Besides studying the recognition of Latin digits and letters, the generality of this modeling approach should be evaluated by also applying it to other types of scripts. Indeed, even if different writing systems appear extremely heterogeneous at a first glance, they might share some basic underlying structures that can be unveiled by deep learning. At the same time, testing deep neural networks on more challenging scripts is a useful way to better assess their computational capabilities and their flexibility. In this section we report and discuss a set of simulations concerned with learning a hierarchical generative model of Farsi handwritten characters (Sadeghi & Testolin, *under review*). Despite the Farsi alphabet shares many features with other alphabets, the high similarity among particular characters and their high variation in appearance due to different writing styles make recognition particularly challenging.

The Farsi alphabet is based on Arabic scripts and consists of 32 letters. The only difference between Farsi and Arabic scripts is in the character sets, with Farsi having four more characters than the Arabic alphabet. The complexity of the Farsi script arises from the fact that distinguishability among many letters lies in the placement of dots or zigzag bars. Many Farsi letters have one, two or three dots located above or below the main pattern. While the dots appear isolated in printed documents, two or three dots are most of the time grouped together in handwritten letters and are shaped as a caret, dash, or tilt based on handwriting styles. Moreover, bars are joined to the main stroke in printed alphabet, but they are most of the time separated in handwritten letters. Farsi text is written horizontally, from right to left. Finally, besides the complexity introduced by handwritten (instead of printed) patterns, an additional complication derives from the fact that letters in a word are shaped differently based on the surrounding context, i.e., the writing pattern changes depending on the adjacent letters that will be linked to it. In particular, the first letter of a word

[2] I would like to thank Zahra Sadeghi for her great contribution to these simulations.

is joined from left, the middle letters are joined from both sides, and the last letter is joined from right.

4.4.1 Dataset and network architecture

In our simulations we used a freely available Farsi dataset called HODA (Khosravi & Kabir, 2007), containing both handwritten digits and letters collected from students registered in the Iran nationwide university entrance exam. The digit dataset includes 60000 training and 20000 test images, while the letter dataset includes 70645 training are 17706 test images. Each pattern is encoded in gray-scale using a 32x32 pixel matrix, thereby requiring 1024 visible units as input to the network. Two sample sets of handwritten digits (with 10 examples for each digit) and handwritten letters (with 3 examples for each letter) are provided in Fig. 26.

We used the same deep architecture adopted for the MNIST simulations presented in Section 4.1, composed by 500 (H1), 500 (H2) and 2000 (H3) hidden units (see Fig. 14B). The unsupervised learning algorithm for the RBMs used a learning rate of 0.0001 and a momentum coefficient of 0.9, which was initialized to 0.5 for the first few epochs. Learning was performed using a mini-batch scheme, with a mini-batch size of 100 patterns for the digit dataset and 300 patterns for the letter dataset. Learning continued until the reconstruction error converged to a small, asymptotic value.

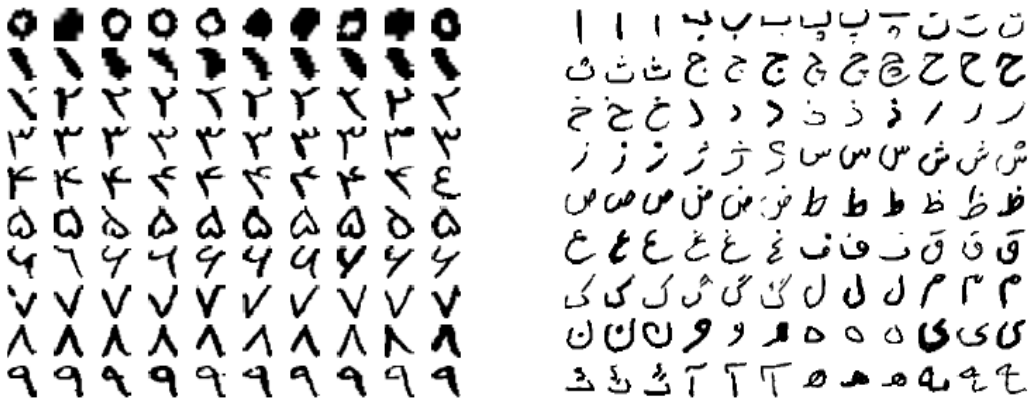

Figure 26. Samples of digits (left) and letters (right) from the HODA Farsi dataset.

4.4.2 Simulations

In our first set of experiments, we used two separate deep networks for learning digit and letter shapes. After learning a hierarchical generative model in a completely unsupervised way, we performed a simple linear read-out on the top-level internal representations extracted by the networks in order to classify the test patterns. As a baseline, we also applied the read-out module directly on the raw input patterns and on the top-level representations derived from a hierarchical architecture using random connections. Indeed, it has been shown that neural networks with random connection weights can nevertheless reach good recognition accuracy in several recognition tasks (e.g., Widrow, Greenblatt, Kim, & Park, 2013), and this comparison is also useful to check if the higher dimensionality of the internal representations could be largely responsible to the performance gain.

Following the transfer learning framework discussed in Section 3.3.2, in a second set of simulations we tested the hypothesis that handwritten digits and letters share many common features, and that a generative model that captures the statistical structure of the letter distribution might also be used to support the recognition of digits. We therefore used the deep network trained as a generative model on the letter dataset to create a high-level representation of the digit patterns, and then we trained the linear read-out module to classify such representations with the correct digit label. Moreover, we also tested the transfer ability over a different type of script by using the deep network trained on Farsi letters to create an high-level representation of the Latin digits contained in the MNIST dataset used in Section 4.1. This high-level representation was then used to support classification operated by a linear read-out module. Also in this case, baseline results were obtained by applying the read-out directly on the raw input patterns and on the top-level representations derived from a randomly connected deep network.

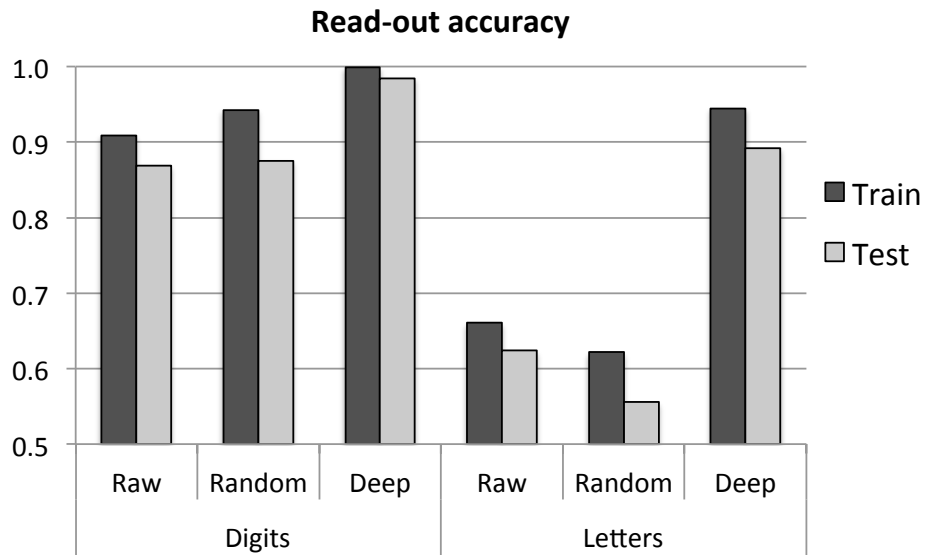

Figure 27. Classification accuracy for Farsi digits and letters obtained by a linear read-out module applied on different types of representation.

4.4.3 Results and discussion

In line with previous results on Latin characters, both deep networks reached very high classification accuracy also with the challenging Farsi script, thereby supporting the idea that a hierarchical generative model is a natural choice to model handwritten characters. Classification accuracy for both datasets is reported in Fig. 27. A qualitative analysis of the receptive fields extracted by the deep networks shows that the basic features encoded at the lower levels of the hierarchy resemble those found in other learning problems involving written patterns (see Fig. 28A). The features of the deepest layer, instead, appear to represent more complex structures, sometimes becoming tuned to specific classes.

In order to better investigate the type of errors made by the network, we analyzed the more common mistakes revealed by the confusion matrix. It turned out that some specific classes of Farsi letters are extremely difficult to discriminate in the presence of noise or inaccurate writing due to their fine-grained structure.

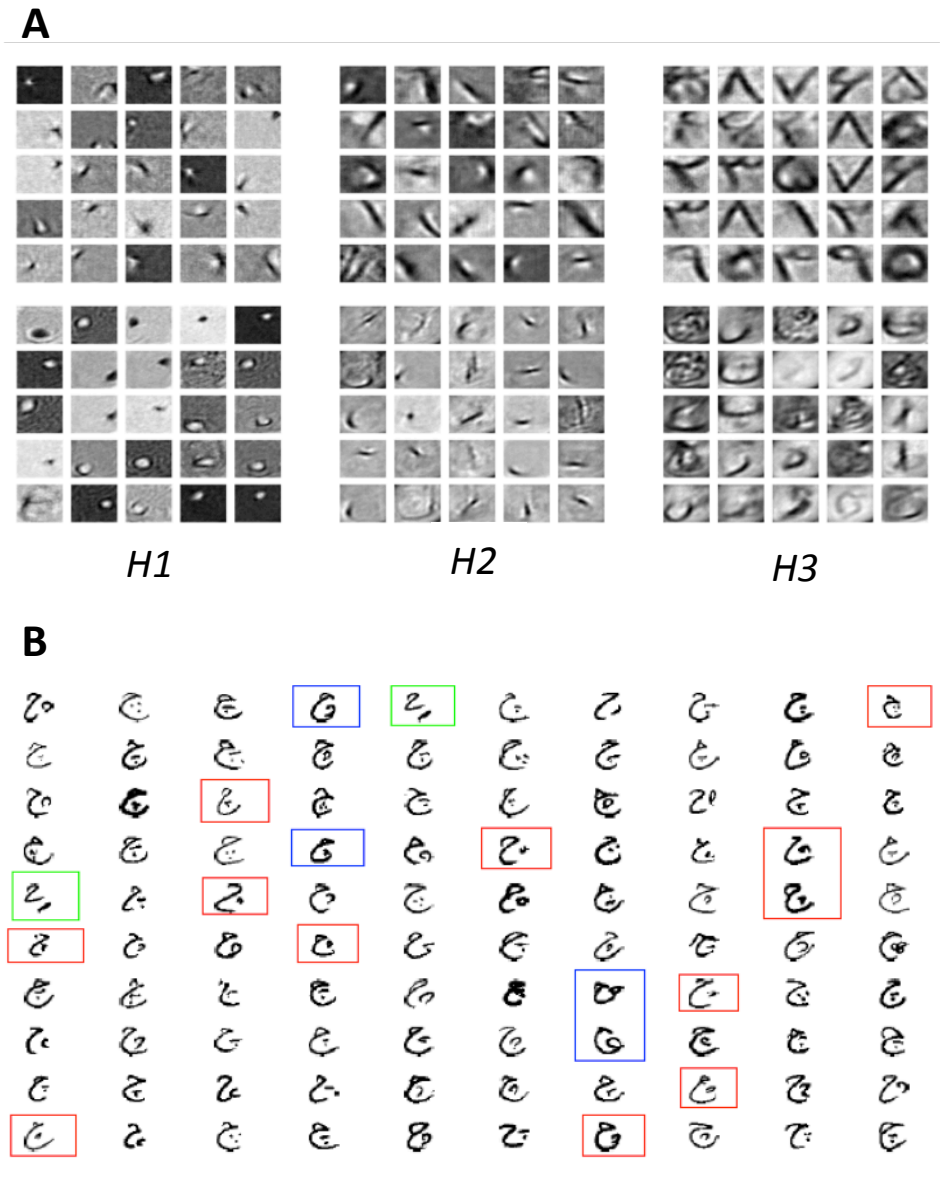

Figure 28. **(A)** Some representative receptive fields of the units belonging to the different hidden layers of the deep network trained on the digits (top row) and letters (bottom row). **(B)** Examples of challenging samples of the "Jim" Farsi letter. Three situations of poorly recognizable patterns are distinguished by red (dots compressed into a semi-circle), green (dots located below the main body), and blue (dots attached to the main body) rectangles.

In particular, in Fig. 28B we report some extremely challenging recognition situations, where the letter "Jim" has been written in a variety of different styles that greatly reduce pattern discriminability. Despite these challenging situations, the high read-out accuracy obtained by the linear classifier on the

deepest layer of the networks suggests that both the digit and letter generative models discovered useful abstract structure from the data distribution, which can be readily exploited by a simple classifier to discriminate between the underlying classes. We therefore investigated whether the same type of structure learned on handwritten letters can be used to also describe the structure of handwritten digits. As expected, the read-out accuracy remains very high, reaching 98.1% for the training patterns and 95.9% for the test patterns (i.e., we observe a decrease of less than 3%). This result corroborates the hypothesis that the distribution of letters and digits can be described using a common set of features, which can be effectively extracted from the data using a hierarchical generative model. More generally, this analysis also suggests how knowledge extracted from one domain can be readily transferred to perform tasks on related domains.

In order to investigate if knowledge transfer can also occur between two domains that are apparently more distant, we applied the same method to investigate transfer learning between different alphabets. In particular, we used the deep network trained as a generative model on Farsi letters to compute a high-level representation of the Latin digits contained in the MNIST dataset. Interestingly, as reported in Table 1 also in this case the read-out classification accuracy remains fairly high (approaching 95% in the test set), although it does not reach state-of-the-art performance.

	MNIST Digits	
	Train %	**Test %**
Deep network trained on Farsi letters	95.9 ± 0.1	94.9 ± 0.2
Raw data	86.8 ± 0.0	43.0 ± 0.0
Random deep network	79.5 ± 2.3	38.7 ± 1.0

Table 1. Classification accuracy for MINST digits obtained by a linear read-out module applied on different representations.

In summary, these results suggest that despite different writing systems have some distinguishing features, they also share many commonalities, some of which can be effectively captured by hierarchical generative models. In particular, our simulations show that the apparently marked visual differences between Farsi and Latin scripts might only be present at a “surface” level, while they might share many common features at a deeper, structural level. This computational framework might therefore be very useful to better understand the intimate nature of different alphabets, for example suggesting how bilinguals could exploit their abstract similarities to improve learning and knowledge transfer.

4.5 Learning sequences of letters: Spatial arrangement

In the remaining series of simulations, we extended the single-letter scenarios presented above by studying learning of multiple letters, which are combined together in order to create English words. In this section we consider sequences of letters organized into a spatial arrangement, which means that the entire word is given as input to the network at a single timestep, thereby allowing processing of all the letters in parallel.

4.5.1 Dataset and network architecture

Concatenating several letters encoded as raw pixel values (as done in Sections 4.3 and 4.4) would result in a significantly larger input pattern, which might require some form of convolutional architecture to be efficiently processed. One possible solution to mitigate this complexity is to simply replicate many single-letter networks at different spatial positions in order to separately process each letter of the word. The high-level, internal representations created by each network can then be concatenated together, forming the input pattern for a subsequent layer that would therefore learn the compositional structure of the letter sequence.

Another possibility, which we adopted in our experiments, is to directly exploit a more abstract, feature-based representation of each single letter, which allows to efficiently process the whole word as a single input pattern. We therefore encoded each letter as a combination of simple geometric features, which correspond to the "Siple font" used in the original IAM paper (McClelland & Rumelhart, 1981). This encoding scheme represents each letter using 14 basic features, as shown in Fig. 29A. However, we slightly increased the complexity of the task by encoding each pattern using a "pixelated" version of the Siple font, thereby implicitly representing geometric features as the simultaneous activation of several aligned pixels arranged into a 7x7 matrix. In this setting, each input letter was therefore represented using 49 visible units.

Figure 29. **(A)** The geometric "Siple font" used to encode each letter. **(B)** Network architecture.

We then created a dataset of four-letter words, which corresponded to the original set of 1180 English words used in the IAM simulations. The resulting input patterns were therefore given to the network using a total of (7x7)x4 = 196 visible units, and the hierarchical architecture consisted of two layers of, respectively, 120 (H1) and 200 (H2) hidden units (see Fig. 29B).

4.5.2 Simulations

After learning, the deep network was first tested on several inference tasks, in which the input patterns were disrupted by means of Gaussian noise, occlusions, or by randomly turning-off active pixels. The network was then asked to recover the original configuration by iteratively performing top-down reconstructions on the input values.

The network was then tested on two different discriminative problems. The first required the identification of each of the four letters composing a word, using as label a binary vector with one-hot (i.e., localistic) coding of the target letter. The second problem consisted in the identification of the word itself, using as label a binary vector with one-hot coding of the target word. To investigate the quality of the representations extracted by deep learning, we compared the classification accuracy on the representations learned at each of

the levels of the two-layer network with that of the representations learned by a single-layer RBM, with as many hidden units as the top layer of the deep belief network (H = 200 units). As a baseline, we also measured the classification accuracy obtained by trying to directly categorize the raw input vectors. Note that the read-out of the original data is trivial, due to lack of variability in the coding of letters and words (i.e., there is a unique pattern for each letter and word). Indeed, the raw data vectors are linearly separable as shown by the perfect accuracy of the read-out. However, if the input patterns are degraded by adding a certain amount of noise (i.e., by randomly turning to zero a certain amount of pixels), one should expect a progressive decrease of the classification accuracy when the input representation does not include high-level, invariant features.

4.5.3 Results and discussion

As a preliminary analysis, we investigated the type of representations encoded at different levels of the hierarchy by simply plotting the receptive fields of the hidden units (see upper part of Fig. 30A). It appears that units in the first hidden layer learn to encode simple geometric features resembling those used in the Siple font. At the second layer, these basic features are combined into more complex patterns, sometimes resembling entire letters or letter parts. We also tested a sparse variant of deep belief networks, which consists in forcing the internal representations to rely on a limited number of active hidden units. This variant can be easily implemented by adding a penalty term on the total amount of hidden activations (H. Lee et al., 2008). However, when applying this sparsity constraint it appears that the features became less selective, probably because each unit is forced to encode more information (see lower part of Fig. 30A).

The representations extracted by the deep network allowed to effectively perform inference in the reconstruction tasks, guaranteeing good reconstructions even in highly noisy conditions.

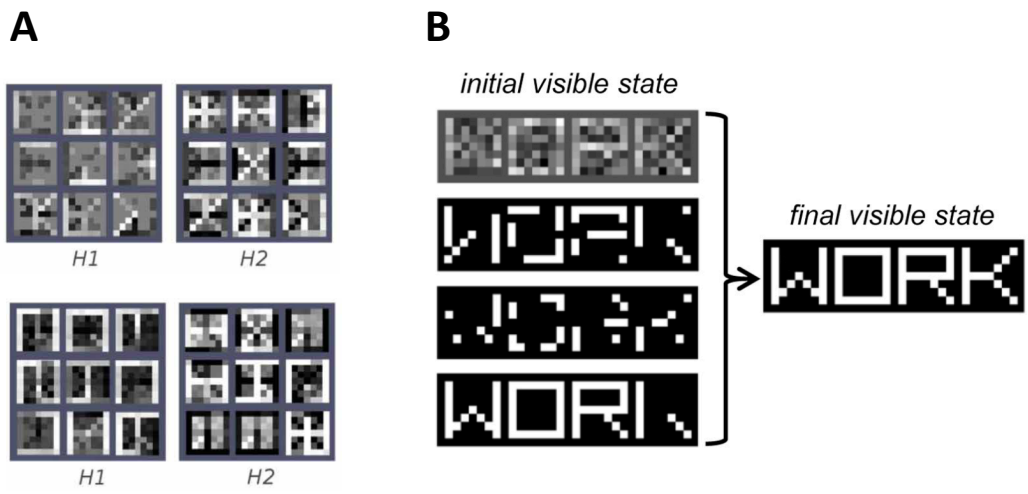

Figure 30. **(A)** Receptive fields obtained at different levels of the hierarchy. The lower panel shows the sparse variant. **(B)** Reconstruction ability of the network under a variety of noise conditions.

Fig. 30B shows the result of inference when four different noisy versions of the same word were given as input to the model: Gaussian random noise, moderate level of binary noise, high level of binary noise, and a simple occlusion. Note that the visible units always settled onto an activation state corresponding to the correct word image.

Results about letters and word discrimination tasks are reported in Fig. 31. When the input word image was corrupted by randomly setting to zero a certain percentage of its pixels, read-out accuracy on the raw pixel data dropped even with a small amount of noise (approaching zero in the word recognition task). As expected, the hierarchical network instead extracted robust internal representations that were less sensitive to noise. Indeed, both hidden layers supported good discrimination accuracy for letters, whereas only the deepest hidden layer adequately supported word discrimination. Notably, the shallow generative model (single-layer RBM) did not unfold word-level information, thereby failing to support robust word recognition (especially for larger noise levels).

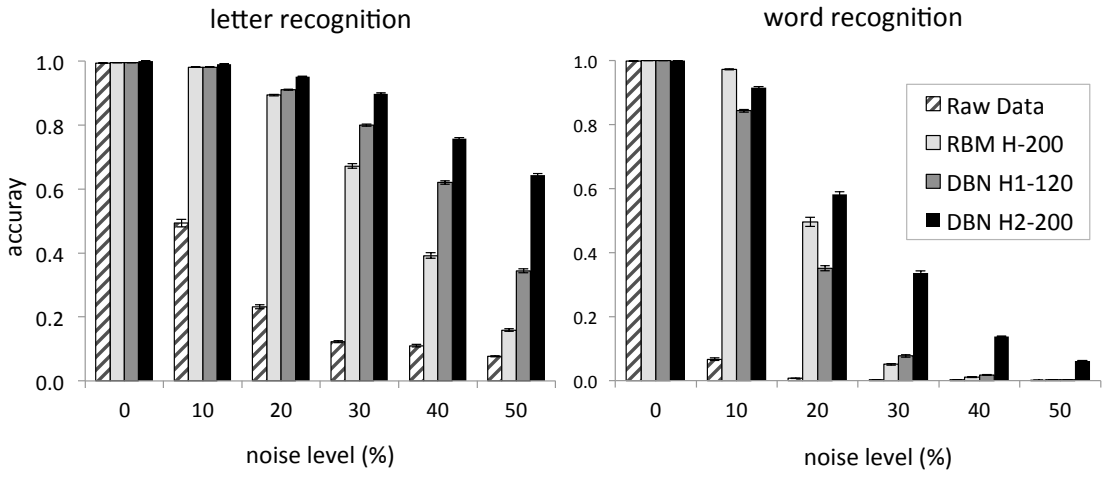

Figure 31. Recognition accuracy of different models in the task of **(A)** identifying each component letter or **(B)** the whole word encoded in the input pattern, as the level of noise increases. Raw data accuracy was obtained by directly training a linear read-out on the input patterns, while in the other cases the classifier was applied to the hidden layer of a single RBM (RBM H-200) or to each of the two hidden layers of a deep belief network (DBN).

These results suggest that a hierarchical architecture is more suited to effectively represent orthographic information contained in visual displays, with lower levels of the hierarchy extracting simpler information that can successively be combined into more abstract words representations. Despite its simplicity, the emergent properties of this model revisit key aspects of the seminal IAM and suggest a very promising research direction for developing a full-blown deep learning model of visual word recognition.

Indeed, up-scaling the present toy model is likely to be successful because deep learning is particularly suited to capture features hierarchies over large training datasets with great pattern variability. We therefore expect that a similar architecture could be exploited to study visual word recognition using more realistic stimuli, which would also allow a more direct comparison with empirical data. For example, a larger model would be useful to better understand how location-invariant representations of written words might

emerge by hierarchical generative learning over spatially arranged letter sequences, and what type of intermediate coding would better support abstraction of the identity and position of letters from their eye-centered locations (some preliminary results are reported in Di Bono & Zorzi, 2013).

4.6 Learning sequences of letters: Temporal arrangement

In this section we still consider the problem of learning sequences of letters, but to differ from the scenario discussed above here the letters have a temporal arrangement, which means that only one letter is processed at a single timestep. This input regimen does not allow processing all the letters of a word in parallel, because the orthographic structure must be discovered by extracting the transitional probabilities occurring over a dynamic input stream (Testolin, Stoianov, Sperduti, & Zorzi, *under review*).

At a first analysis, this setting might appear unrelated to the traditional notion of orthographic processing, which often assumes a parallel processing of the letters constituting a word. However, several connectionist models of visual word recognition and reading aloud entail a serial processing mechanism (Perry, Ziegler, & Zorzi, 2007; Plaut, 1999; Sibley, Kello, Plaut, & Elman, 2008), and sequential processing of letters is prominent during reading acquisition in childhood, whereby phonological decoding bootstraps the development of orthographic representations (Share, 1995; Ziegler, Perry, & Zorzi, 2014). Sequential generation of letters is also a prominent feature of written spelling, and it is a key aspect in popular computational models of spelling (Glasspool & Houghton, 2005; Houghton, Glasspool, & Shallice, 1995). Moreover, extracting temporal structures from sequences of elements is a ubiquitous feature of language-related processes, which might play an important role in subsequent stages of orthographic processing (for example, by providing support for high-level mappings between phonological and orthographic representations).

Due to the temporal nature of the task, here we did not use deep belief networks, but we applied the Temporal Recurrent Restricted Boltzmann Machine (RTRBM) discussed in Section 3.4. Also in this case, computation was significantly speeded-up by using graphic processors, and the complete source code is publicly available for download[3]. Unlike models that use slot-based

[3] http://ccnl.psy.unipd.it/research/RTRBM

representations on which visible units encode position-specific elements of a sequence, recurrent neural networks learn to gradually integrate temporal information over time, generalizing knowledge about letters across positions by encoding their statistical relations in the hidden layer. In this way, the internal representations created in the hidden layer can implicitly encode distal temporal interactions that can span an arbitrary number of elements. A recurrent network might therefore in principle be able to build fixed-width, internal representations of whole sequences as static activation patterns (Sibley et al., 2008; Stoianov, 1999). However, as we will discuss below, it appears that in the RTRBM very similar words are internally represented by very similar activation patterns, which might not guarantee a perfectly discriminable, static encoding of whole sequences (Testolin, Sperduti, Stoianov, & Zorzi, 2012).

In order to learn the sequential structure of words, the RTRBM was exposed to a corpus of English monosyllables, with each word presented one letter at a time. After learning, the generative model was expected to have inferred the orthographic structure underlying the training data. To assess this, we first evaluated the accuracy of context-dependent predictions (simulation 1). Moreover, the model should be able to reproduce the training sequences and generalize, thus producing well-formed pseudowords (simulation 2).

4.6.1 Dataset and network architecture

We used a dataset of 6,670 English monosyllables of variable length (from 3 to 7 letters) extracted from CELEX (Baayen, Piepenbrock, & van Rijn, 1993), which is an electronic corpus that comprises general lexicons for British English, German and Dutch. The dataset was randomly split into a training set of 5,300 words and a test set of 1,370 words. Words were codified as sequences of letters, represented with fixed-size binary orthogonal vectors of 27 units (one for each possible letter, plus a termination symbol). Therefore, in this case we did not use neither a realistic, pixel-based or a simplified, feature-based representation to encode single letters, which were assumed to be already abstract elements. This

simplified representation scheme allows to isolate the problem of finding the positional relation among letters from the problem of building an abstract representation of each element.

Learning parameters were tuned with the aim of obtaining high prediction accuracy on the training data. In particular, the resulting architectures had 200 hidden units, and learning was stopped when no significant improvements occurred on the training set (after approximately 300 epochs). The performance of the RTRBM was compared with that of another popular connectionist model used to learn temporal data, the Simple Recurrent Network (SRN; Elman, 1990), and with other two popular families of probabilistic generative models for sequential data, that is *n*-grams (P. Brown, DeSouza, Mercer, Della Pietra, & Lai, 1992) and Hidden Markov Models (HMMs; Rabiner, 1989).

SRNs are feed-forward neural networks composed by three layers. The input layer contains both the current element of the sequence that is being processed and the contextual information encoded by the network, which is simply a copy of the hidden layer activities at the previous timestep. At the beginning of a sequence, the activations of context units are usually set to zero. An output layer is then used to perform a prediction of the next element of the sequence, and learning is performed by back-propagating the mismatch error between the network prediction and the target value. In SRNs, therefore, supervised learning is used to establish a mapping between the input (i.e., the current element of the sequence plus contextual information) and a separate output representation (i.e., the prediction of the following element). Processing in SRNs is thereby inherently deterministic and essentially input-driven (i.e., bottom-up), while RTRBMs can autonomously produce top-down activations on the sensory units from internal representations through their intrinsically stochastic dynamics. As for the RTRBM, learning parameters of the SRN were tuned to maximize accuracy on the training data. The resulting network had 200 hidden neurons, thereby the two connectionist models had about the same number of connections (the SRN had slightly more connections due to the

additional set of hidden-to-output weights, as also shown in Fig. 33A). This implies that both models have approximately the same complexity (i.e., the same number of parameters to be fit). Learning rate was set to 0.01 and training was performed for 250 epochs.

The *n*-gram models have been implemented as look-up tables, where each row contains the successor distribution computed from the training data for each possible context (i.e., the *n-1* preceding letters), with *n* varying between 2 and 4 (see Fig. 32A). These models therefore treat two sequences as equivalent if they end in the same *n - 1* letters: assuming a value $k \geq n$, it holds that

$$P(l_k \mid l_1^{k-1}) = P(l_k \mid l_{k-n+1}^{k-1}) \qquad (21)$$

where l_1^{k-1} represents the sequence of letters $l_1 l_2 \ldots l_{k-1}$ and l_k is the *k*-th letter of a word. Although this might seem a somewhat crude approximation, *n*-grams have demonstrated very good performances (P. Brown et al., 1992) and still constitute a reference framework for language modeling. One of the major drawbacks of *n*-grams is caused by data sparsity: items not present in the training set will be given a probability of zero, which motivates the use of smoothing techniques. In our study, we used a simple form of additive smoothing (Chen & Goodman, 1996).

We tested HMM of first- and second-order, with a number of states ranging from 7 to 60 according to a previous study (Sang & Nerbonne, 1999). HMMs assume that the system being studied can be modeled as a Markov process with a certain number of unobserved (i.e., hidden) states. In first-order models, the probability of being in a certain state at the current timestep only depends on the state of the model at the immediately preceding timestep. In second-order models, instead, this dependence is extended to the last two states. Each state has an associated emission distribution that describes the probability of emitting (i.e., observing) each symbol of the alphabet from that state. The transition distribution, instead, specifies the probability of moving from each state to any

other. If two states are not connected, the corresponding transition probability will be zero. Finally, the initial state distribution specifies the probability of starting the generation of a sequence from each of the states of the model. The parameters of an HMM can be estimated using an iterative procedure, known as Baum-Welch algorithm (Rabiner, 1989), which adjusts the probability distributions in order to raise the likelihood of the training data using an expectation-maximization method. As for the other models, HMM hyper-parameters were tuned to maximize accuracy on the training data. In particular, the highest performance was obtained using a first-order model with 40 hidden states, trained for 10000 iterations with a likelihood cut-off of 0.001 and 1000 steps in the Baum-Welch algorithm.

4.6.2 Simulation 1: context-dependent predictions

In the first set of simulations we evaluated the performance of the considered models on predicting the next element of a sequence, given a certain context (see Fig. 32A). Accuracy was measured as mean prediction error on both training and test sets using a computationally efficient procedure that exploits a tree-based data structure. In particular, we evaluated the response of each model across all possible left contexts in the evaluation sets (i.e., variable-length, initial parts of words). To this aim, we created a *k*-tree data structure, where *k* is the size of the alphabet (26 letters plus one termination symbol). Words are encoded as paths in the tree, starting from the root. Every node in the tree represents a left context (which is the path from the root to the current node) and it might have a number of children or alternatively constitute the end of a word. It is possible to efficiently compute the *empirical successor distribution* of each context in the dataset by counting the frequency of each child of a node (i.e., the frequency of each letter following that context) and normalizing the resulting vector to sum up to 1. Once the empirical successor distribution has been computed for all the variable-length contexts in the dataset, each model is probed with all possible contexts in order to compute the *predicted successor*

distributions. The vectors of empirical and predicted successor distributions can then be compared according to some metric in order to measure the discrepancy between observed and predicted values. We used the Kullback-Leibler (KL) divergence as distance metric (see Eq. 10). We preferred the KL-divergence instead of other metrics (e.g., Euclidean distance or cosine similarity) because of its sound probabilistic interpretation and its direct link to the notions of cross-entropy and perplexity, which are two other metrics commonly used to assess language models. Nevertheless, it is worth noting that the results reported below are robust with respect to the type of metric (see Testolin et al., 2012, for a preliminary study based on the Euclidean distance measure). The total error of each model was the average KL-divergence across all possible contexts in the evaluation datasets.

In order to compute the predicted successor distribution for the RTRBM, a response was collected by sequentially clamping the visible units on the given letters (i.e., left context) and letting the network generate visible-layer activations. The normalized activations (i.e., summing up to 1) constitute the predicted successor distribution *M*, which corresponds to the conditional probability distribution of all letters in the alphabet given the context $v_1, v_2, \ldots, v_{t-1}$ encoded by the hidden unit activation h_{t-1}:

$$M \leftarrow P(v_t \mid v_1, v_2, \ldots, v_{t-1}) \approx P(v_t \mid h_{t-1}) \qquad (22)$$

The vector representing the predicted successor distribution for each context was also obtained for the other models tested. For SRNs, we collected the output values (normalized to sum up to 1) in response to a given context. For *n*-grams, the successor distribution corresponded to the row associated with a particular context. For HMMs, the optimal sequence of hidden states (i.e., the one with the highest probability under the current context) was first computed using the Viterbi algorithm, and then the successor distribution was read out from the emission probabilities of the last state of the sequence.

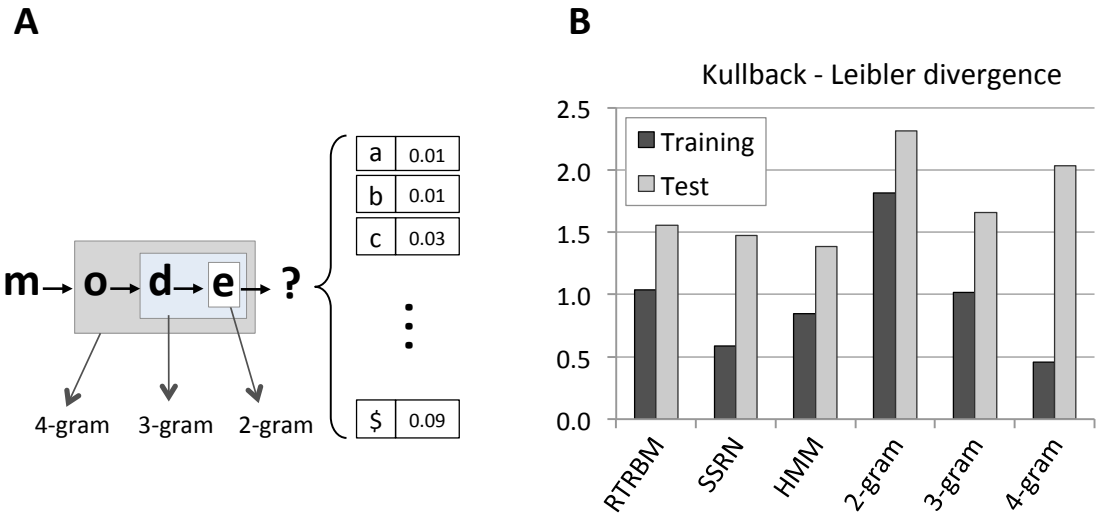

Figure 32. **(A)** A prototypical prediction problem, on which a certain context is given (i.e. the first four letters of a word) and the aim is to predict the probability distribution of the following letter. **(B)** Prediction error on training set (gray) and test set (black) for different models, measured as the KL-divergence between predicted and empirical distributions (small is better).

4.6.3 Results and discussion

Prediction errors for the different models are plotted in Fig. 32B. The prediction accuracy of the RTRBM on the test set was higher than that of all *n*-gram models, and just slightly lower than that of the SRN and HMM. It can be noted that the 2-gram model is inadequate for accurately predicting which letter will follow a given context, because it only takes into account the last letter as context. Other models reach better accuracy thanks to their ability to consider longer context when making predictions. Moreover, the results confirmed a critical limitation of *n*-gram models, which is their poor generalization. The longer the context of the *n*-gram, the greater was the prediction accuracy on the training data. However, on test data the performance of the 4-gram model significantly decreased due to coding of too specific contexts. This limitation could be alleviated by using more sophisticated smoothing techniques.

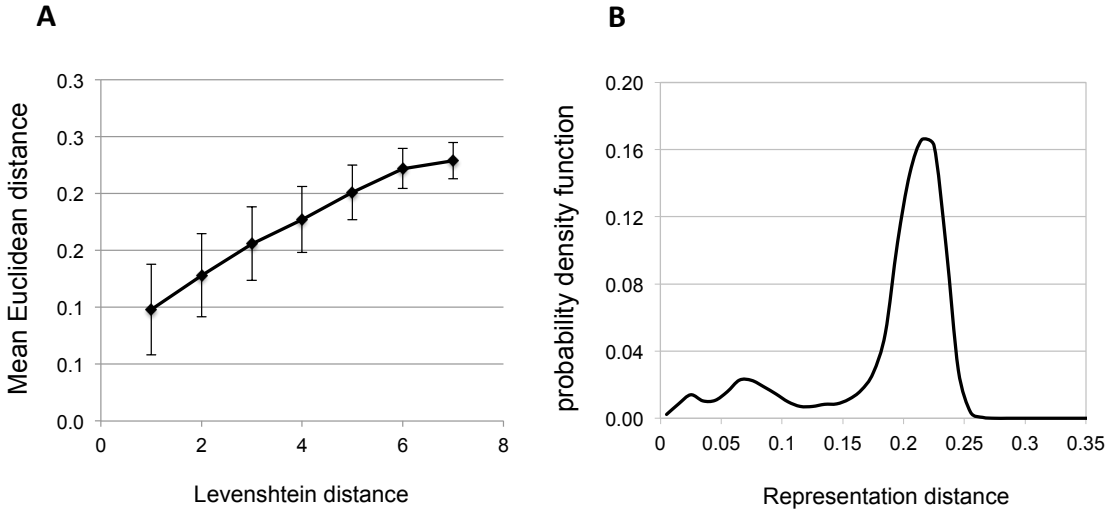

Figure 33. **(A)** Correlation between internal representations similarity and Levenshtein distances of corresponding words. **(B)** Probability density function of mean Euclidean distances between internal representations.

On the other hand, both connectionist models avoid the problem of specificity by exploiting distributed representations of the context, which turns into good generalization performance. Overall, the results show that both the RTRBM and the SRN successfully learned the transition probabilities between the letters of English words, yielding a level of prediction accuracy that is comparable to that of other popular sequential language models.

Analysis of the internal (i.e., hidden layer) representations, generated after the production of the last letter of a word, revealed that the similarity between the representations (calculated as Euclidean distances) is correlated with the similarity between the corresponding sequences (measured with the Levenshtein distance), with a correlation coefficient *r* of 0.42 (see Fig. 33A). This means that similar words, for example those that have Levenshtein distance 1 (i.e., they differ only by one single-character edit, where an edit can be an insertion, a deletion or a substitution) are internally represented by highly similar activation patterns. Inspection of the frequency distribution of mean Euclidean distances between patterns (Fig. 33B) revealed a bimodal distribution, best fit by a mixture of two Gaussians: G1 ($\mu1 = 0.08$, $\sigma1 = 0.04$) and G2 ($\mu2 =$

0.21, σ2 = 0.02) with mixing coefficients *p1* = 0.20 and *p2* = 0.80. This means that a consistent number of sequences are encoded using highly similar representations (corresponding to G1), and this happens to be the case for the majority of words with Levenshtein distance of 1. Although it is not clear how this high similarity affects the discriminability between words, it suggests that the RTRBM is not always producing static, holistic representations of entire sequences.

4.6.4 Simulation 2: generative abilities

In a second set of simulations, we went on to investigate the ability of the considered models to autonomously generate well-formed sequences of letters. Indeed, a generative model will produce a sequence of letters even when there is no external context to drive the generation.

Stochastic simple recurrent networks (SSRNs)

Due to the deterministic, input-driven nature of the SRN, its basic version cannot be used to autonomously generate sequences. We therefore propose a simple extension of the model that allows to produce sequences from the learned probability distribution. The proposed extension can be applied to tasks exploiting a localistic, one-hot encoding of the input pattern. In this scenario, an external stochastic process can be used to sample the next element following a given sequence according to learned conditional probabilities, thereby allowing to autonomously generate sequences from the model. A graphical representation of this Stochastic Simple Recurrent Network (SSRN) is given in Fig. 34A, while Fig. 34B illustrates the generation process of a three-elements sequence when the network is unfolded in time. Stochastic sampling can be implemented by first transforming the SRN's output activations to a (conditional) probability distribution. A straightforward way to realize this is to calculate the corresponding cumulative distribution, and then select the letter corresponding to a random number drawn from the interval [0,1].

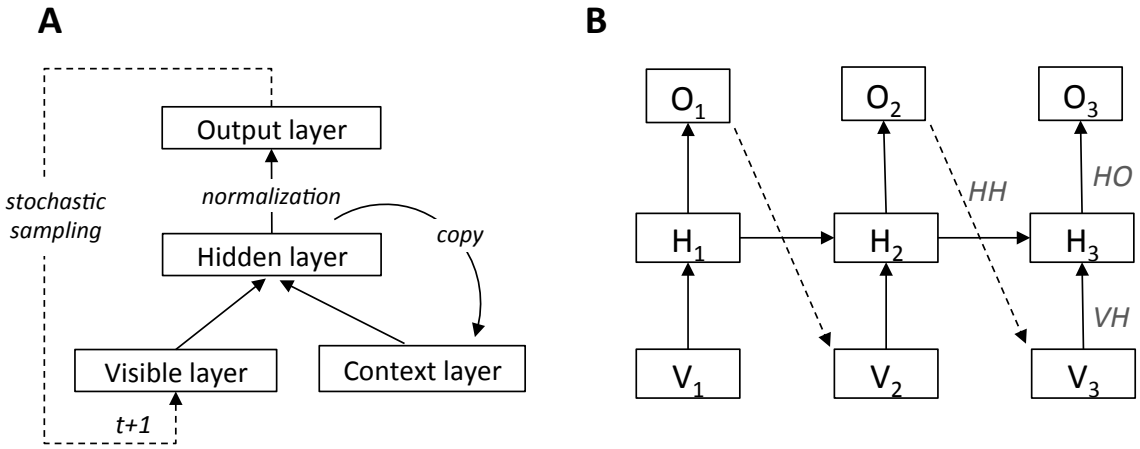

Figure 34. **(A)** Graphical representation of the Stochastic Simple Recurrent Network (SSRN). The output activations are first normalized, and an external stochastic process then samples the next element of the sequence, which is given as input to the network at the following timestep. **(B)** Schematic diagram of the SSRN generating a 3-element sequence. Note that this architecture requires an additional set of hidden-to-output (*HO*) weights compared to the RTRBM.

The selected element is given as input to the network at the next timestep, and this process is repeated until the termination symbol is produced. Even if we used logistic sampling, it should be noted that other approaches can be used to obtain a probability distribution in the output units, for example by using a softmax function (see McClelland, 2013, for a discussion about the relation of logistic and softmax sampling and their Bayesian interpretation).

We note that, although RTRBMs and SSRNs have some common characteristics, they differ in several fundamental aspects. In SSRNs, supervised learning is used to establish a mapping between the input (i.e., the current element of the sequence plus contextual information) and a separate output representation (i.e., the prediction of the following element), as illustrated in Fig. 34A. In contrast, RTRBMs use a common layer for encoding both the input and the model's prediction (see Fig. 13A), and learn to process sequential information in a completely unsupervised way by trying to accurately reproduce (i.e., generate) the training sequences. Processing in SSRNs is thereby inherently

deterministic and essentially input-driven (i.e., bottom-up), while RTRBMs can autonomously produce top-down activations on the sensory units from internal representations through their intrinsically stochastic dynamics.

4.6.5 Evaluation procedure

We evaluated the quality of the letter sequences generated by the learning models by comparing them with those contained in the training and test sets, and with those produced by two published pseudoword generator algorithms. We considered the ARC pseudowords database (Rastle, Harrington, & Coltheart, 2002) and Wuggy (Keuleers & Brysbaert, 2010). The ARC database contains 310,000 non-pseudohomophonic monosyllabic pseudowords, built using a hand-crafted grammar that defines phonological constraints on monosyllables. A set of phoneme-to-grapheme correspondences extracted from CELEX is used to derive possible spellings of legal phonological strings, which are then converted back to phonological representations using a set of grapheme-phoneme correspondences. Finally, phonological strings that differ from the initial phonologies are excluded from the database. The Wuggy pseudoword generator takes a different approach. Instead of combining subsyllabic elements like in the ARC database, it starts from a given set of words, which are syllabified and used to build a bigram chain. Pseudowords are then generated by recursively iterating through the chain. Wuggy is particularly interesting for our comparison, because it does not use phonological representations and it starts the generation from a reference list of words. Thus, we could generate pseudowords using the same training set of the RTRBM.

Each model was used to generate an arbitrary number of sequences, which was chosen to be 300 times the size of the training set (i.e. 1,590,000 samples). We then calculated two indexes: *completeness* of the generation, computed as the ratio between the number of sampled sequences that also appeared in the training set (without repetitions) and the total number of sequences contained in the training set; and *fidelity* of the generation, computed as the ratio between

the sampled sequences that also appeared in the training set (possibly repeated) and the total number of sampled sequences. The first indicator describes the ability of the model to regenerate the training sequences and it depends on the sampling size. Augmenting the number of the samples generally increases the completeness of the generation. The second indicator does not depend on the size of the sampling and gives an idea about the model tendency to generate new wordforms instead of reproducing only previously seen sequences: the lower the fidelity, the greater this tendency.

Given that all models generated a consistent amount of new wordforms (as reported below), we inspected the quality of the generated strings that did not belong to the training set. All models produced a number of real English words that were not part of the training set, which we excluded from the analysis in order to allow a fair comparison with the pseudoword generator algorithms. We therefore analyzed the 20,000 most frequently generated pseudowords composed by at least 3 letters. We randomly selected the same number of pseudowords (with at least 3 letters) from the ARC database for a comparison with its underlying generation algorithm. The Wuggy pseudoword generator was supplied with the words of the training dataset as input to build the bigram chain, and we selected the 20,000 most frequently generated pseudowords using the following parameter set: maximal number of candidates was set to 15, maximal search time was set to 10 seconds, and all output restrictions were required (i.e., match length of subsyllabic segments, match letter length, match transition frequencies and match subsyllabic segments). The set of pseudowords generated by each model or algorithm was evaluated in terms of the following statistical features (Duyck, Desmet, Verbeke, & Brysbaert, 2004): *sequence length*, that we expected to be close to the average length of words in the training set; *orthographic neighborhood*, the number of orthographic neighbors that a string has (i.e., an orthographic neighbor is a word of the same length that differs from the original string by only one letter; for example, given the pseudoword ‘at’, the words ‘bat’, ‘fat’, ‘cat’ and ‘tab’ are

orthographic neighbors); and *constrained bigrams and trigrams frequency*, which are the averaged type frequency of constrained bigrams (trigrams) for the wordform. A constrained bigram is defined as a specific two-letter combination in a specific position and specific word length. That is, 'es' in 'best' is considered the same as in 'nest', but is different both in 'yes' (different length) and in 'does' (different position).

4.6.6 Results and discussion

The results of the generation process in terms of completeness and fidelity are shown in Table 2.

	RTRBM	**SSRN**	**HMM**	**2-gram**	**3-gram**	**4-gram**
Completeness	91%	98%	92%	67%	96%	99%
Fidelity	16%	37%	13%	5%	21%	58%

Table 2. Completeness and fidelity of the generation process for each tested model.

With regard to the RTRBM and the SSRN, Fig. 35 shows that both indicators improved as training proceeded and that eventually both networks were able to generate a large fraction of the words in the training set. Nevertheless, the low fidelity suggests that both models are not encoding entire sequences, but they rather exploit local transition rules during the generative process. Indeed, the networks generated many legal sequences that were not present in the training set (a few samples generated by the RTRBM are reported in Fig. 36A). Some of these sequences were in fact real English words that were not part of the training set, and a similar pattern was found for the other models. It is worth noting that the generated pseudowords might be composed by "legal" bigrams (i.e., combinations of two subsequent letters that are observed in the training set) or by novel (and potentially illegal) bigrams.

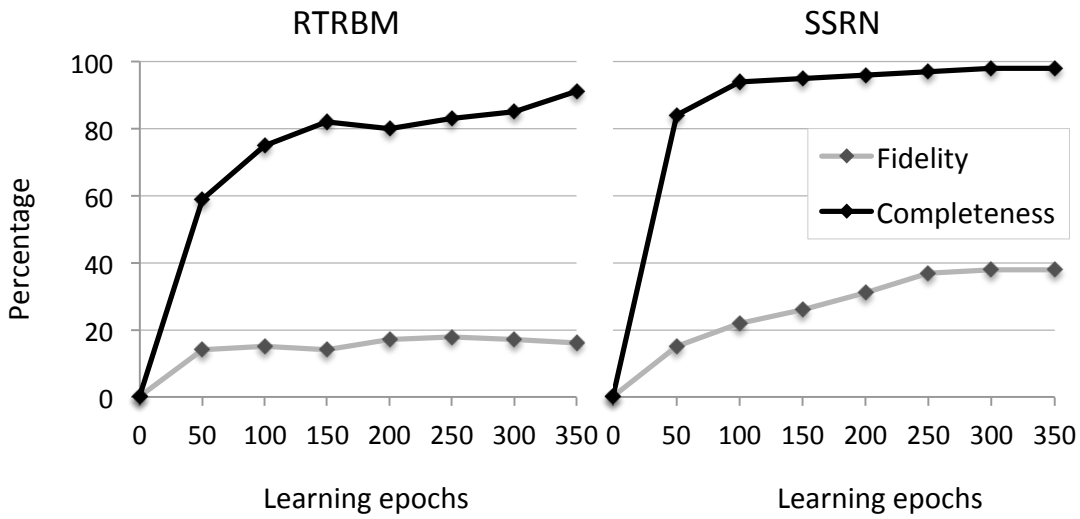

Figure 35. RTRBM and SSRN completeness and fidelity (percentage) of generation as a function of training time (in epochs).

To investigate this point, we computed the fraction of illegal bigrams produced by the RTRBM with respect to the total generated bigrams in the considered set of pseudowords. We found a ratio of 0.2%, which confirms that the RTRBM generated many novel pseudowords without introducing illegal bigrams.

Results of the analysis of pseudoword quality for all models are shown in Fig. 36. The number of pseudowords shorter than 3 letters that was excluded from the analysis was negligible for all the models. As expected, the average length of the pseudowords generated by the different models was similar to that of the words in the training set, except for the 4-gram model, which produced longer sequences. Interestingly, only the RTRBM and Wuggy never generated words longer than 7 letters, which is the maximum length of the words in the training data (note that Wuggy was explicitly required to respect this constraint). All the other models, instead, generated a certain number of pseudowords longer than 7 letters. This ranged from just 16 for the SSRN to several thousand for the 4-gram model.

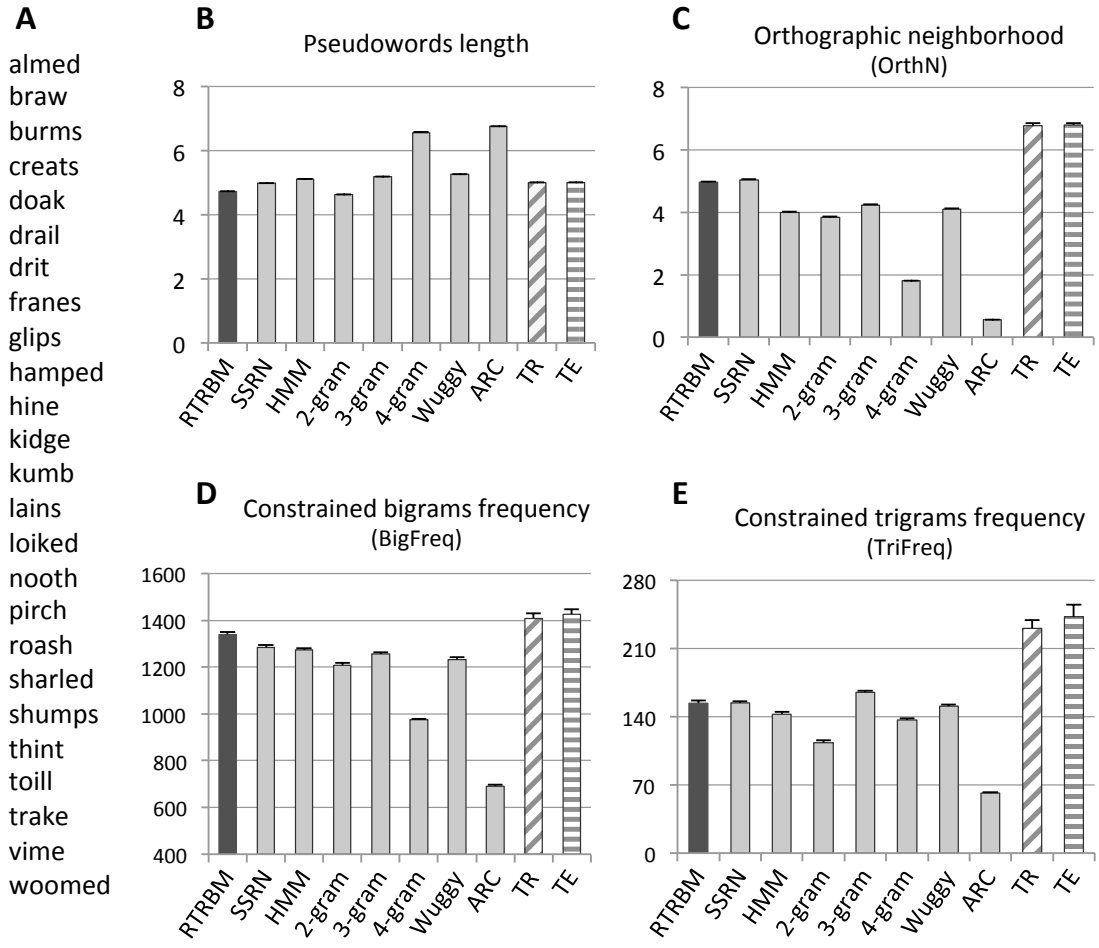

Figure 36. **(A)** Sample of pseudowords generated by the RTRBM. Average length **(B)**, average orthographic neighborhood **(C)**, constrained bigrams frequency **(D)** and constrained trigrams frequency **(E)** collected over 20,000 pseudowords generated by different models and over words of the training (TR) and test (TE) datasets.

Importantly, pseudowords generated by connectionist models had the highest mean orthographic neighborhood (4.96 for the RTRBM and 5.03 for the SSRN), followed by those generated by the 3-gram model and Wuggy; see Fig. 36C. On the other hand, ARC pseudowords had the lowest orthographic neighborhood (0.56). The mean constrained bigram frequency (Fig. 36D) was higher for the RTRBM compared to all other models, approaching the value of the words in the training set. Both these measures indicate that the RTRBM produced high-quality pseudowords.

Note also that the statistics computed over the training set are very close to those computed over the test set (compare the last two columns in Fig. 36B-E), thereby showing that these values are representative of the statistical distribution in English monosyllabic words. It is also interesting to note that the RTRBM, the SSRN and Wuggy generated pseudowords with similar statistics, even if the latter exploits a sophisticated algorithm based on bigram chains that are carefully constructed taking into account linguistic information and that are processed using an optimized search procedure (Keuleers & Brysbaert, 2010).

It should finally be emphasized that our assessment of the generative ability of the network was not performed over a sample of pseudowords generated by human subjects. This would constitute an important research direction, because there is no study to our knowledge that has investigated the spontaneous production of pseudowords by humans. An empirical study on human pseudoword generation would provide a very informative baseline to measure the quality of different models.

5 Conclusions

5.1 Discussion and summary of contributions

Generative neural networks constitute a promising modeling approach to study cognitive processes within a parallel distributed processing framework. In particular, Restricted Boltzmann Machines (RBMs) are a powerful class of recurrent neural networks that can learn a probabilistic generative model from a set of high-dimensional patterns in a completely unsupervised fashion. The recent introduction of efficient learning algorithms and the possibility to implement these models on high-performance parallel computing machines allow to effectively train large-scale networks in a reasonable amount of time. Moreover, single-layer RBMs can be easily stacked together in order to build multi-layer architectures known as deep learning systems, which implement hierarchical generative models by extracting increasingly more abstract representations from the sensory input (Hinton, 2007). Other extensions of single-layer RBMs use a set of delayed recurrent connections to propagate information over time, thereby allowing to extract temporal structures from sequences of patterns (Sutskever et al., 2008).

Generative neural networks reach impressive performance on many challenging machine learning tasks, ranging from visual object recognition (Krizhevsky et al., 2012) and speech perception (Mohamed et al., 2012) to completely different domains, such as optimization of wireless systems (Badia et al., 2014; Testolin et al., 2014). However, their application as cognitive models is still very limited (e.g. Di Bono & Zorzi, 2013; Stoianov & Zorzi, 2012; Zorzi et al., 2013). In this thesis, we presented several techniques that can be used to study deep learning systems from a cognitive modeling perspective. We also discussed some critical features of this type of generative networks that make them particularly appealing to model cortical processing. In particular, the presence of top-down generative connections, the unsupervised nature of the learning

algorithm and the interactive propagation of signals across the processing hierarchy suggest that these models constitute an important step forward to improve current connectionist models of cognitive functions. Moreover, the probabilistic interpretation of these recurrent networks in terms of graphical models improves the probabilistic characterization of cognitive systems by proposing how Bayesian computations might be actually implemented in neural circuits.

The modeling framework proposed in this thesis is also consistent with other recent brain theories that describe nervous systems as a sophisticated type of predicting machines (Clark, 2013; Friston, 2010; Huang & Rao, 2011), which not only build a probabilistic internal model of the environment, but also optimize information encoding by only transmitting prediction errors caused by a mismatch between model's expectations and sensory signals. Furthermore, this modeling framework is perfectly aligned with the growing experimental literature highlighting the statistical nature of learning processes in human infants and many other animal species (e.g., Fiser & Aslin, 2002; Fiser, Berkes, Orbán, & Lengyel, 2010; Krogh, Vlach, & Johnson, 2013; Saffran, Aslin, & Newport, 1996; Toro & Trobalón, 2005).

The empirical research performed in this thesis tested the capability of generative neural networks within the prototypical domain of orthographic processing. Learning to recognize written patterns is a remarkable challenge for neural circuits, which must cope with the high variability in shapes and writing styles that characterize writing systems. A seminal proposal to model orthographic processing exploited a hierarchy of hand-coded visual features (McClelland & Rumelhart, 1981), while more recent learning models have shown how supervised learning applied to convolutional architectures lead to very high recognition accuracy (LeCun et al., 1998). Here we demonstrated that impressive recognition accuracy could also be achieved by means of unsupervised generative learning in deep architectures, which extract abstract letter representations by learning a hierarchy of increasingly complex visual

features. We started our research work by first proposing an efficient, parallel implementation of deep belief networks on graphic processing units (GPUs), which allows to significantly reduce computational times required by deep learning simulations by distributing the computation over many simple processors equipped with a shared, large-bandwidth memory. This efficient implementation makes it possible to learn large-scale models, which are trained using big digital datasets containing thousands of high-dimensional patterns.

We then tested the capability of deep networks to build invariant representations of printed letters by performing unsupervised learning over a large dataset of visual patterns containing Latin letters printed with a variety of fonts, sizes and styles. After learning, the model was tested on a classification task by training a linear read-out module on different levels of representation. The linear classifier obtained high recognition accuracy on the deepest internal representations, while it exhibited a much lower accuracy when applied directly on the raw input patterns. Moreover, we tested the performance of deep networks in various generation tasks, where top-down connections are used to produce accurate class prototypes in the sensory layer. However, a careful inspection of the type of receptive fields developed by a deep network at different levels revealed that unsupervised learning did not produce an optimal hierarchical model of the training data, because the complexity of visual features did not always increased along the hierarchy. A possible explanation for this phenomenon is that the variability associated with printed patterns might not be enough to build a proper hierarchy of features. Indeed, better features can be extracted from datasets containing handwritten patterns, whose visual appearance is more variable compared to printed ones. At the same time, an important limitation of using handwritten patterns is that we do not obtain a clear and controlled experimental setting, because the latent factors of variations that are explicit in the printed form (e.g., font type, letter size, etc.) cannot be directly manipulated when using handwritten characters. Despite this limitation, in a second set of simulations we tested the recognition performance

of deep networks on a large dataset containing handwritten Farsi characters, which represent a particularly challenging script due to the presence of fine-grained, cursive visual features and due to the high similarities among certain letters. Confirming the expectations discussed above, in this case the deep network was indeed able to extract a good hierarchy of visual features: receptive fields of the first hidden layer resembled on-centered and off-centered combinations of Gaussian filters, while in the second layer these low-level features were combined in order to form edge detectors and Gabor filters. In the deepest layer receptive fields became even more complex, encoding structured visual features and letter shapes. Notably, also here a linear read-out module trained on the deepest internal representations exhibited very high recognition accuracy, suggesting that hierarchical generative models constitute a promising tool for modeling written patterns belonging to different scripts. Moreover, we tested the hypothesis that different alphabets share a common underlying visual structure, which can be readily captured by deep networks. To this aim, we used a deep network trained in an unsupervised way on handwritten Farsi letters to compute high-level representations of both Farsi and Latin handwritten digits. Such representations were then read-out by a linear classifier, which still exhibited surprisingly high recognition accuracy, thus supporting the idea that visual features can be transferred across different writing domains.

In a separate set of simulations, we also tested the ability of single-layer RBMs to extract low-level visual features from a large dataset containing patches of natural images. We found that unsupervised learning shaped the receptive fields of hidden units in a way that closely matches the outcome of other types of statistical learning algorithms, such as independent component analysis (Bell & Sejnowski, 1997; van Hateren & van der Schaaf, 1998) or sparse coding (Olshausen & Field, 1996). In particular, hidden units learned to encode low-level visual features resembling those observed in the primary visual cortex, such as Gaussian filters, edge detectors and Gabor filters of different spatial orientation and frequency. However, a quite surprising result was that RBMs

also extracted some qualitatively different types of filters, which might be classified as ridgelets and gratings, and which have shown to emerge when using highly overcomplete sparse codes (Olshausen, 2013). Furthermore, we showed that this type of visual features can be readily used to also represent the dataset of Latin printed letters used in the simulations discussed above, and that a linear read-out module applied on these representations reach an impressive recognition accuracy. This interesting result provides further evidence supporting the hypothesis that the shapes of written patterns are not arbitrary, but instead they have been culturally selected to closely match the statistical structure of visual stimuli occurring in our natural environments (Changizi et al., 2006; Dehaene & Cohen, 2007).

We then proceeded by investigating how deep networks could learn patterns composed by multiple letters, which can be arranged in either a spatial or in a temporal configuration. In a set of simulations we exposed a two-layers deep network to a dataset of English words composed by four letters, which were encoded using a simple, geometrical font and arranged in a visual display. The entire letter sequence was therefore processed in parallel. Analyses on the model suggested that the network was building increasingly more complex representations along the hierarchy, with the first layer mostly encoding information about simple features and letter identities, and the second layer combining this information in order to build word-level representations. The network was also successfully tested on a variety of reconstruction and denoising tasks, which showed how top-down generative connections could be used to disambiguate sensory information.

In a final set of simulations, we assessed the performance of a temporal extension of RBMs, called the Recurrent Temporal Restricted Boltzmann Machine (RTRBM), which can process sequential information by using a set of delayed recurrent connections. In this setting, orthographic information had to be extracted in an unsupervised way from English monosyllables presented one letter at a time. After learning, we tested the performance of the network on a

prediction task, where some letters were given as a context and the model was asked to predict the probability distribution of the successive letter. We compared the performance of the network with that of another widely used connectionist architecture for sequential data, the simple recurrent network (SRN; Elman, 1990) and with that of other popular probabilistic generative models (i.e., hidden Markov models and *n*-grams). We also tested the generative abilities of the considered models, by making them generate a large number of sequences that were confronted with a set of pseudowords generated by popular algorithms used in psycholinguistic studies. In order to autonomously generate sequences from the SRN, we extended its basic formulation by introducing an external, stochastic sampling process that select the next letter to generate according to the learned conditional probabilities. We found that both the RTRBM and the stochastic version of SRNs exhibited good prediction and generation performance. On the one hand, this is not surprising because both connectionist models try to predict the next element of a sequence by learning conditional probabilities from the training data. Indeed, there is a tight formal relationship between probabilistic graphical models and recursive neural networks (Baldi & Rosen-Zvi, 2005). However, the two architectures also differ in several fundamental aspects. Learning in the SSRN is concerned with establishing a mapping function between the current input (plus temporal context) and a separate output representation. In contrast, the RTRBM learns an internal model of the data (i.e., the hidden causes that generated the input patterns) by trying to accurately reproduce the incoming information through feedback (i.e., top-down) connections. That is, sequential information is learned by trying to re-generate the training sequences on the same layer that is used for providing the input. Moreover, the SSRN relies on two additional operations, one that performs a non-local normalization over the activations of output units and another that samples the predicted element exploiting an external, *ad-hoc* stochastic process. In contrast, autonomous sequence generation from the RTRBM is an intrinsic feature of the network: there is no need to perform

normalization and to sample from the corresponding distribution, because the probabilistic behavior is caused by the stochastic dynamics that is also a crucial part of the learning process. Nevertheless, the SSRN might be also appealing as a cognitive modeling architecture due to its much simpler formulation and its close relationship to the widely used SRNs. On the other hand, the SSRN cannot be applied when the learning task involves multimodal, distributed representations as input to the network (as in Sutskever et al., 2008) instead of the simpler, one-hot localist scheme that we adopted in our simulations.

In summary, the theoretical contributions and the empirical results presented in this research work provide support for a computational characterization of cognitive systems based on generative neural networks. Though the simulations discussed in this thesis must be expanded and carefully validated against published experimental data, they constitute a promising starting point to develop a comprehensive computational description of complex cognitive processes such as those underlying orthographic processing.

5.2 Future research directions

The theoretical and methodological framework discussed in this thesis needs to be further extended in order to address many challenging and interesting research questions.

Regarding the empirical results on recognition of single characters, simulations should be more accurately evaluated according to the great number of experimental studies that have been conducted during the last decades. In particular, the internal representations and the response errors of a deep network can be readily used to computer letter similarities, which can then be compared to data collected on human subjects (e.g., Gervais, Harvey, & Roberts, 1984; Grainger et al., 2008). Moreover, the performance of the network can be tested on a variety of experimental settings, for example by adding different types of noise in the visual display in order to more precisely identify which are

the most relevant visual features used to recognize each letter (e.g., Fiset et al., 2008; Pelli, Burns, Farell, & Moore-Page, 2006).

The multiple-letters scenario can be extended to cover more challenging input representations, for example by studying recognition of English words encoded at the pixel level. The behavior of a deep network can then be compared to existing data collected on human subjects, or even on non-human primates (e.g., Grainger et al., 2012). Moreover, the deep architecture used to process spatially arranged sequences of letters might be combined with the temporal extension of RBMs, which would allow to effectively process sequences of letters exploiting both serial and parallel processing mechanisms. This would make it possible to better investigate the extent to which unsupervised statistical learning could generate novel word-like units (Saffran, 2001), for example by showing how syntactic structures could emerge by first segmenting words from a continuous stream, and subsequently discovering the permissible orderings of the words (Saffran & Wilson, 2003). In this respect, it is worth mentioning that there exist some other recent generative models, like the Conditional RBM (Taylor, Hinton, & Roweis, 2011), which also allow processing of temporal information and which might be more easily stacked into a hierarchical system.

It should also be noted that orthographic processing is an essential component of reading, thereby incorporating our framework into popular computational models of reading (Zorzi, 2005) would be an important step forward to improve our current understanding of this complex cognitive task. Indeed, reading is a fundamental skill that children must properly acquire in order to be successful in subsequent educational levels, but despite the central relevance of reading for modern societies its neural bases are not yet completely understood (Dehaene, 2009). More comprehensive computational simulations would therefore constitute a valuable tool, possibly allowing to also improve educational practices and to design more effective reading devices. The computational framework proposed in this thesis can serve as a starting point to also explore other interesting phenomena, for example by suggesting how neural

circuits adapted to encode particular statistical features can be re-used to also process other type of information, thereby providing important insights into the cortical recycling hypothesis (Dehaene & Cohen, 2007).

From a broader perspective, it should be noted that sequential statistical learning is a general phenomenon that is found across different sensory modalities (Conway & Christiansen, 2005). Although it is not yet clear whether the underlying mechanism is unitary or modality-constrained, it is interesting to note that the temporal extension of RBMs used in this thesis has been previously applied to modeling video sequences (Sutskever et al., 2008). The RTRBM was able to capture the high-dimensional, multimodal nature of the pixels distribution across subsequent frames, despite the complex nonlinearities characterizing the movies of the dataset. Moreover, an extension of the RTRBM has also been recently used to model temporal dependencies in polyphonic music (Boulanger-Lewandowski, Bengio, & Vincent, 2012), thereby supporting the intriguing hypothesis that music, language and statistical learning might be tightly linked (McMullen & Saffran, 2004; Patel, 2003). Interestingly, one of the possible common threads underlying all these cognitive phenomena might be the involvement of anticipatory, *attentive* processes, which allow to effectively use neural resources by actively predicting the sensory stream. The language of Bayesian statistics has already provided some initial insights about how attention can be formalized within a probabilistic framework: for example, eye movements used to explore a visual scene have been modeled by simulating a cognitive agent that tries to minimize a "surprise" measure (Itti & Baldi, 2009), where the surprise of a stimulus was defined as the Kullback-Leibler divergence between the probabilistic estimate of a hypothesis before and after the observation of some data (if the divergence is high, it means that the observed evidence was unexpected). A promising research direction would therefore be to use generative, sequential neural networks to study how such attentional mechanisms might be implemented in neural circuits, for example by investigating how expectations are propagated over time, while the system is

constantly integrating incoming evidence to update the current internal model of the environment. In this respect, it might be interesting to explore a recent variant of deep belief networks, called deep Boltzmann machine (Salakhutdinov & Hinton, 2012), which implements a fully recurrent deep architecture that uses top-down feedback at each inferential step, thereby allowing to better propagate uncertainty within the system. In a recent study (Reichert, Series, & Storkey, 2013), deep Boltzmann machines have been used to model some symptoms observed in the Charles Bonnet syndrome, where loss of vision leads to vivid visual hallucinations of objects, people, and whole scenes. The authors suggest that such sensory hallucinations might be caused by an unbalanced regulation of bottom-up and top-down neural activity, which might result in false perceptions generated according to expectations produced by the internal model.

Furthermore, by also considering the possibility to *act* on the environment, this approach might be further extended to incorporate some form of active perception, on which the system is not passively waiting for the evidence to come, but it also actively explores the hypothesis space in order to collect the most informative samples. Within this broader framework, the role of attention might be to act at the interplay of perception, action and cognition, at the same time guaranteeing an efficient use of the available processing resources.

It should also be mentioned that in the last few years there have been other significant improvements in training recurrent neural networks, which allow to more efficiently learn temporal models from sequences of patterns. In particular, some authors proposed to extract long-term dependencies from a data stream by exploiting Hessian-free optimization algorithms (Martens & Sutskever, 2011), while other approaches exploit a simplified, linear model to pre-train the recurrent connections, which are successively fine-tuned to capture more distant dependencies (Pasa & Sperduti, 2014; Pasa, Testolin, & Sperduti, 2014, 2015).

Finally, testing this type of models in more realistic situations would also require to further improve their hardware and software implementation, for

example by relying on more efficient parallel computing architectures. Graphic processors constitute a promising technology to significantly scale-up simulations of parallel processing biological systems, because they provide an enormous computational power with an affordable cost. However, it is likely that we will have to resort to even more efficient computing machines in order to build truly large-scale simulations, where rich, high-dimensional sensory inputs are processed in real-time by complex neural networks composed by billions of artificial neurons. Within this respect, a promising research direction is to exploit *neuromorphic* hardware, which relies on advanced processing units that more closely mimic the physical dynamics observed in biological synapses (e.g., Jo et al., 2010; Strukov, Snider, Stewart, & Williams, 2008).

In summary, current research on artificial neural networks and probabilistic models is an extremely active, ever expanding field, which will hopefully improve our understanding of cognition from a computational perspective. However, although the path toward these promising research directions might appear straightforward, it requires the integration of many different, complementary approaches and research methodologies. In order to succeed, researchers should therefore never forget the fundamentally interdisciplinary nature of cognitive science, which takes advantage from a variety of disciplines and schools of thought. In this thesis we focused on the computational modeling framework, which has some distinguishing features that make it particularly appealing to investigate many cognitive phenomena. However, we should also keep in mind that computational models are primarily tools for exploring the implication of ideas (McClelland, 2009), and that they must always be complemented with other research frameworks and perspectives if we really want to grasp the amazing complexity of nervous systems.

References

Abbott, L. F. (2008). Theoretical neuroscience rising. *Neuron*, *60*(3), 489–95.

Ackley, D., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*(1), 147–169.

Anderson, J. (1983). *The architecture of cognition*. Cambridge, MA: Hardward University Press.

Andrieu, C., De Freitas, N., Doucet, A., Jordan, M. I., & Freitas, N. De. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50*(1), 5–43.

Ashby, W. R. (1956). *An introduction to cybernetics.* London, UK: Chapman and Hall.

Atick, J., & Redlich, A. (1992). What does the retina know about natural scenes? *Neural Computation*, *4*, 196–210.

Baayen, R. H., Piepenbrock, R., & van Rijn, H. (1993). The CELEX lexical database [CD-ROM]. Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania.

Badia, L., Munaretto, D., Testolin, A., Zanella, A., Zorzi, M., & Zorzi, M. (2014). Cognition-based Networks: Applying Cognitive Science to Multimedia Wireless Networking. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. Sydney (AUS).

Baldi, P., & Rosen-Zvi, M. (2005). On the relationship between deterministic and probabilistic directed Graphical models: from Bayesian networks to recursive neural networks. *Neural Networks*, *18*(8), 1080–1086.

Barlow, H. (1961). Possible principles underlying the transformation of sensory messages. *Sensory Communication*.

Bassett, D. S., & Gazzaniga, M. S. (2011). Understanding complexity in the human brain. *Trends in Cognitive Sciences*, *15*(5), 200–9.

Beer, R. (2000). Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, *4*(3), 91–99.

Bell, A. J., & Sejnowski, T. J. (1997). The “independent components” of natural scenes are edge filters. *Vision Research*, *37*(23), 3327–38.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127.

Bengio, Y., & Lamblin, P. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, *19*, 153–170.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*, 157–166.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Botvinick, M. M., & Plaut, D. C. (2004). Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action. *Psychological Review*, *111*(April), 395–441.

Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning*.

Brown, M., Hua, G., & Winder, S. (2011). Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 43–57.

Brown, P., DeSouza, P., Mercer, R., Della Pietra, V., & Lai, J. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*(4), 467–479.

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, *106*(3), 551–581.

Candès, E., & Donoho, D. (1999). Ridgelets: A key to higher-dimensional intermittency? *… of the Royal …*, *357*(1760), 2495–2509.

Caruana, R. (1998). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Cattell, J. M. (1886). The Time it Takes to See and Name Objectsm. *Mind*, *11*, 220–242.

Changizi, M., Zhang, Q., & Ye, H. (2006). The structures of letters and symbols throughout human history are selected to match those found in objects in natural scenes. *The American Naturalist*.

Chen, S., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (pp. 310–318).

Chikkerur, S., Serre, T., Tan, C., & Poggio, T. (2010). What and where: a Bayesian inference theory of attention. *Vision Research*, *50*(22), 2233–47.

Chomsky, N. (1957). *Syntactic Structures*. Mouton De Gruyter.

Churchland, P. S., Ramachandran, V., & Sejnowski, T. J. (1994). A Critique of Pure Vision. In *Large-scale neuronal theories of the brain* (pp. 22–60).

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. MIT press.

Ciresan, D., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *Neural Computation*, *22*(12), 3207–3220.

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, *36*(3), 181–204.

Cohen, J. D., Dunbar, K., & McClelland, J. L. (1990). On the control of automatic processes: a parallel distributed processing account of the Stroop effect. *Psychological Review*, *97*(3), 332–410.

Cohen, L., & Dehaene, S. (2004). Specialization within the ventral stream: the case for the visual word form area. *NeuroImage*, *22*(1), 466–76.

Cohen, L., Dehaene, S., Naccache, L., Lehericy, S., Dehaene-Lambertz, G., Henaff, M., & Michel, F. (2000). The visual word form area: Spatial and temporal characterization of an initial stage of reading in normal subjects and posterior split-brain patients. *Brain*, *123*, 291–307.

Conway, C. M., & Christiansen, M. H. (2005). Modality-constrained statistical learning of tactile, visual, and auditory sequences. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *31*(1), 24–39.

Cottrell, G. W. (2008). Looking around the backyard helps to recognize faces and digits. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). Ieee.

Daugman, J. G. (2001). Brain metaphor and brain theory. In W. Bechtel, P. Mandik, J. Mundale, & R. Stufflebea (Eds.), *Philosophy and the neurosciences: A reader* (pp. 23–36). Blackwell.

Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, *7*(5), 889–904.

De Filippo De Grazia, M., Stoianov, I., & Zorzi, M. (2012). Parallelization of Deep Networks. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 621–626). Bruges, Belgium.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V, … Ng, A. Y. (2012). Large Scale Distributed Deep Networks. *Advances in Neural Information Processing Systems*, *24*, 1–9.

Dehaene, S. (2009). *Reading in the brain: The new science of how we read.* Penguin.

Dehaene, S., & Cohen, L. (2007). Cultural recycling of cortical maps. *Neuron*, *56*(2), 384–398.

Dehaene, S., Cohen, L., Sigman, M., & Vinckier, F. (2005). The neural code for written words: a proposal. *Trends in Cognitive Sciences*, *9*(7), 335–41.

Di Bono, M. G., & Zorzi, M. (2013). Deep generative learning of location-invariant visual word recognition. *Frontiers in Psychology*, *4*(September), 635.

DiCarlo, J. J., & Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, *11*(8), 333–41.

DiCarlo, J. J., Zoccolan, D., & Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, *73*(3), 415–34.

Dong, D., & Atick, J. (1995). Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus. *Network: Computation in Neural Systems*, *6*(2), 159–178.

Dupuy, J.-P. (2009). *The mechanization of the mind: On the origins of cognitive science.* Bradford Book.

Duyck, W., Desmet, T., Verbeke, L. P. C., & Brysbaert, M. (2004). WordGen: a tool for word selection and nonword generation in Dutch, English, German, and French. *Behavior Research Methods*, *36*(3), 488–99.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Elman, J. L., Bates, E., Johnson, M., Karmiloff-smith, A., Parisi, D., & Plunkett, K. (1996). *Rethinking Innateness: A Connectionist Perspective on Development*. (J. L. Elman, Ed.). Cambridge, MA: MIT Press.

Falchier, A., & Clavagnier, S. (2002). Anatomical Evidence of Multimodal Integration in Primate Striate Cortex. *The Journal of Neuroscience*, *22*(13), 5749–5759.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*(1), 1–47.

Ffytche, D. H., & Zeki, S. (1996). Brain activity related to the perception of illusory contours. *Neuroimage*, *3*(3), 104–108.

Fiser, J., & Aslin, R. N. (2002). Statistical learning of higher-order temporal structure from visual shape sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *28*(3), 458–467.

Fiser, J., Berkes, P., Orbán, G., & Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, *14*(3), 119–30.

Fiset, D., Blais, C., Ethier-Majcher, C., Arguin, M., Bub, D., & Gosselin, F. (2008). Features for identification of uppercase and lowercase letters. *Psychological Science*, *19*(11), 1161–8.

Fodor, J. (1975). *The language of thought.* Cambridge, MA: Hardward University Press.

Friston, K. J. (2010). The free-energy principle: a unified brain theory? *Nature Reviews. Neuroscience*, *11*(2), 127–38.

Fukushima, K. (1980). Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *202*.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741.

Gerstner, W., Sprekeler, H., & Deco, G. (2012). Theory and simulation in neuroscience. *Science*, *338*(6103), 60–5.

Gervais, M. J., Harvey, L. O., & Roberts, J. O. (1984). Identification confusions among letters of the alphabet. *Journal of Experimental Psychology: Human Perception and Performance*, *10*(5), 655–66.

Glasspool, D. W., & Houghton, G. (2005). Serial order and consonant-vowel structure in a graphemic output buffer model. *Brain and Language*, *94*(3), 304–30.

Goldstein, J. (1999). Emergence as a construct: History and issues. *Emergence*, 49–72.

Goodale, M. a., & Milner, A. D. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, *15*(1), 20–25.

Graham, D., & Rockmore, D. (2011). The packet switching brain. *Journal of Cognitive Neuroscience*, 267–276.

Grainger, J. (2008). Cracking the orthographic code: An introduction. *Language and Cognitive Processes*, *23*(1), 1–35.

Grainger, J., Dufau, S., Montant, M., Ziegler, J. C., & Fagot, J. (2012). Orthographic processing in baboons (Papio papio). *Science*, *336*(6078), 245–8.

Grainger, J., Rey, A., & Dufau, S. (2008). Letter perception: from pixels to pandemonium. *Trends in Cognitive Sciences*, *12*(10), 381–7.

Griffiths, T. L., Kemp, C., & Tenenbaum, J. B. (2008). Bayesian models of cognition. In R. Sun (Ed.), *Cambridge handbook of computational cognitive modeling* (pp. 59–100). Cambridge, MA: Cambridge University Press.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: II. feedback, expectation, olfaction, illusions. *Biological Cybernetics*, *202*, 187–202.

Hebb, D. (1949). *The Organization of Behavior: A Neuropsychological Theory*.

Hertz, J. A., Krogh, A. S., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Weasley.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences*, *11*(10), 428–34.

Hinton, G. E. (2010a). A practical guide to training Restricted Boltzmann Machines. *Technical Report UTML TR 2010-003, University of Toronto*, *9*, 1.

Hinton, G. E. (2010b). Learning to represent visual input. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *365*(1537), 177–84.

Hinton, G. E. (2013). Where Do Features Come From? *Cognitive Science*, 1–24.

Hinton, G. E., & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *352*(1358), 1177–1190.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 77–109). MIT Press, Cambridge, MA.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–7.

Hinton, G. E., & Sejnowski, T. J. (1999). *Unsupervised Learning: Foundations of Neural Computation*. Cambridge, MA: MIT Press.

Hinton, G. E., & Shallice, T. (1991). Lesioning an attractor network: investigations of acquired dyslexia. *Psychological Review*, *98*(1), 74–95.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(April), 2554–2558.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Houghton, G., Glasspool, D. W., & Shallice, T. (1995). *Spelling and serial recall: Insights from a competitive queuing model*. (G. D. A. Brown & N. C. Ellis, Eds.) (pp. 365–404). Chichester: John Wiley.

Huang, Y., & Rao, R. P. N. (2011). Predictive coding. *Wiley Interdisciplinary Reviews. Cognitive Science*, *2*(5), 580–593.

Hubel, D., & Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*, 106–154.

Hyvärinen, A., Hurri, J., & Hoyer, P. O. (2009). *Natural Image Statistics* (Vol. 39). London: Springer London.

Insel, T., Landis, S., & Collins, F. (2013). The NIH brain initiative. *Science*, *340*(May), 687–688.

Itti, L., & Baldi, P. (2009). Bayesian surprise attracts human attention. *Vision Research*, *49*(10), 1295–306.

Jain, A. K., Duin, P. W., & Mao, J. (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 4–37.

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., & Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters*, *10*(4), 1297–301.

Joanisse, M., & Seidenberg, M. S. (1999). Impairments in verb morphology after brain injury: A connectionist model. *Proceedings of the National Academy of Sciences of the United States of America*, *96*(June), 7592–7597.

Jones, M., & Love, B. C. (2011). Bayesian Fundamentalism or Enlightenment? On the explanatory status and theoretical contributions of Bayesian models of cognition. *Behavioral and Brain Sciences*, *34*(4), 169–88; disuccsion 188–231.

Jordan, M. I., & Sejnowski, T. J. (2001). *Graphical Models: Foundations of Neural Computation*. (M. I. Jordan & T. J. Sejnowski, Eds.)*Pattern Analysis & Applications* (Vol. 5). Cambridge, MA: MIT Press.

Kanan, C., & Cottrell, G. W. (2010). Robust classification of objects, faces, and flowers using natural image statistics. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2472–2479). Ieee.

Kanizsa, G. (1979). *Organization in vision: Essays on Gestalt perception.* New York: Praeger.

Kastner, S., & Ungerleider, L. G. (2000). Mechanisms of visual attention in the human cortex. *Annual Review of Neuroscience*, 315–341.

Keuleers, E., & Brysbaert, M. (2010). Wuggy: a multilingual pseudoword generator. *Behavior Research Methods*, *42*(3), 627–33.

Khosravi, H., & Kabir, E. (2007). Introducing a very large dataset of handwritten Farsi digits and a study on their varieties. *Pattern Recognition Letters*, *28*(10), 1133–1141.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simmulated annealing. *Science*, *220*(4598), 671–680.

Knill, D. C., & Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, *27*(12), 712–9.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *69*, 59–69.

Kok, P., Jehee, J. F. M., & de Lange, F. P. (2012). Less is more: expectation sharpens representations in the primary visual cortex. *Neuron*, *75*(2), 265–70.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. Cambridge, MA: The MIT Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *24*, 609–616.

Krogh, L., Vlach, H. a, & Johnson, S. P. (2013). Statistical learning across development: flexible yet constrained. *Frontiers in Psychology*, *3*(January), 598.

Kruger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J., … Wiskott, L. (2013). Deep Hierarchies in the Primate Visual Cortex: What Can We Learn

For Computer Vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1847–1871.

Kullback, S., & Leibler, R. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Lange, S., & Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Ieee.

Le Roux, N., & Bengio, Y. (2008). Representational power of Restricted Boltzmann Machines and Deep Belief Networks. *Neural Computation*, *20*(6), 1631–1649.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 384–398.

LeCun, Y., & Cortes, C. (1998). MNIST Optical Character Database at AT&T Research. *Http://yann.lecun.com/exdb/mnist*.

Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief net models for visual area V2. *Advances in Neural Information Processing Systems*, *20*, 873–880.

Lee, T. S., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, *20*(7), 1434.

Lee, T. S., & Nguyen, M. (2001). Dynamics of subjective contour formation in the early visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, *98*(4), 1907–11.

Lehky, S. R., & Sejnowski, T. J. (1988). Network model of shape-from-shading: neural function arises from both receptive and projective fields. *Nature*, *333*(6172), 452–4.

Lochmann, T., & Deneve, S. (2011). Neural processing as causal inference. *Current Opinion in Neurobiology*, *21*(5), 774–81.

Markram, H. (2012). The Human Brain Project. *Scientific American*, *306*(6), 50–55.

Marr, D., & Poggio, T. (1979). A Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, *204*(1156), 301–328.

Marslen-Wilson, W. D., & Welsh, A. (1978). Processing interactions and lexical access during word recognition in continuous speech. *Cognitive Psychology*, *10*(1), 29–63.

Martens, J., & Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In *International Conference on Machine Learning*.

Maturana, H., & Varela, F. (1980). *Autopoiesis and cognition: the realization of the living.* Dordrecht, NL: Reidel Publishing Company.

McCandliss, B. D., Cohen, L., & Dehaene, S. (2003). The visual word form area: expertise for reading in the fusiform gyrus. *Trends in Cognitive Sciences*, *7*(7), 293–299.

McCarthy, J. (1959). Recursive functions of symbolic expressions and their computation by machines. *Communications of the ACM*.

McClelland, J. L. (2009). The place of modeling in cognitive science. *Topics in Cognitive Science*, *1*(1), 11–38.

McClelland, J. L. (2010). Emergence in cognitive science. *Topics in Cognitive Science*, *2*, 751–770.

McClelland, J. L. (2013). Integrating probabilistic models of perception and interactive neural networks: a historical and tutorial review. *Frontiers in Psychology*, *4*(August), 503.

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 in Cognitive Sciences*, *14*(8), 348–56.

McClelland, J. L., & Elman, J. L. (1986). The TRACE Model of Speech Perception. *Cognitive Psychology*, *18*(1), 1–86.

McClelland, J. L., McNaughton, B., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*(3), 419–457.

McClelland, J. L., Mirman, D., Bolger, D. J., & Khaitan, P. (2014). Interactive Activation and Mutual Constraint Satisfaction in Perception and Cognition. *Cognitive Science*, *38*, 1139–1189.

McClelland, J. L., & Rogers, T. T. (2003). The parallel distributed processing approach to semantic cognition. *Nature Reviews. Neuroscience*, *4*(April), 310–322.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. An account of basic findings. *Psychological Review*, *88*(5), 375–407.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

McGurk, H., & MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, *264*(December), 746–748.

McMullen, E., & Saffran, J. R. (2004). Music and language: A developmental comparison. *Music Perception: An Interdisciplinary Journal*, *21*(3), 289–311.

Minsky, M. (1961). Steps Toward Artificial Intelligence. *Proceedings of the IRE*, (January).

Minsky, M., & Papert, S. (1969). *Perceptrons: an introduction to computational geometry*. *MIT Press, Cambridge, Mass*. Cambridge, MA: MIT Press.

Mnih, A., & Hinton, G. E. (2007). Three new graphical models for statistical language modelling. In *International Conference on Machine Learning* (pp. 641–648). New York, New York, USA: ACM Press.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems* (pp. 1–9).

Mohamed, A., Dahl, G. E., & Hinton, G. E. (2012). Acoustic Modeling Using Deep Belief Networks. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 14–22.

Morowitz, H. J. (2002). *The emergence of everything: How the world became complex.* Oxford University Press.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Nerbonne, J., & Stoianov, I. (2004). Learning phonotactics with simple processors. *On the Boundaries of Phonology and Phonetics*, 89–121.

Newell, A., & Simon, H. (1961). Computer simulation of human thinking. *Science*, *134*(3495), 2011–2017.

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *Queue*, *6*(2), 40.

Niell, C. M. (2013). Vision: more than expected in the early visual system. *Current Biology*, *23*(16), R681–4.

Noble, D. (2006). *The music of life: biology beyond the genome*. Oxford: Oxford University Press.

Norris, D. (2006). The Bayesian reader: explaining word recognition as an optimal Bayesian decision process. *Psychological Review*, *113*(2), 327–57.

Olshausen, B. A. (2013). Highly overcomplete sparse coding. In *IS&T/SPIE Electronic Imaging* (Vol. 8651).

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609.

Orban, G. a. (2008). Higher order visual processing in macaque extrastriate cortex. *Physiological Reviews*, *88*(1), 59–89.

Owens, J., & Houston, M. (2008). GPU computing. *Proceedings of the IEEE*, *96*(5), 879–899.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Pasa, L., & Sperduti, A. (2014). Pre-training of Recurrent Neural Networks via Linear Autoencoders. *Advances in Neural Information Processing Systems*, 3572–3580.

Pasa, L., Testolin, A., & Sperduti, A. (2014). An HMM-based Pre-training Approach for Sequential Data. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 467–472).

Pasa, L., Testolin, A., & Sperduti, A. (2015). Neural networks for sequential data: a pre-training approach based on Hidden Markov Models. *Neurocomputing*, 1–38.

Patel, A. D. (2003). Language, music, syntax and the brain. *Nature Neuroscience*, *6*(7), 674–81.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA: Morgan Kaufmann.

Pelli, D. G., Burns, C. W., Farell, B., & Moore-Page, D. C. (2006). Feature detection and letter identification. *Vision Research*, *46*(28), 4646–74.

Perry, C., Ziegler, J. C., & Zorzi, M. (2007). Nested incremental modeling in the development of computational theories: The CDP+ model of reading aloud. *Psychological Review*, *114*(2), 273–315.

Peterson, C., & Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, *1*(5), 995–1019.

Pinker, S. (1999). How the mind works. *Annals of the New York Academy of Sciences*, 119–127.

Plaut, D. C. (1999). A Connectionist Approach to Word Reading and Acquired Dyslexia: Extension to Sequential Processing. *Cognitive Science*, *23*(4), 543–568.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., & Patterson, K. (1996). Understanding normal and impaired word reading: computational principles in quasi-regular domains. *Psychological Review*, *103*(1), 56–115.

Plunkett, K., & Marchman, V. (1993). From rote learning to system building: acquiring verb morphology in children and connectionist nets. *Cognition*, *48*(1), 21–69.

Pylyshyn, Z. W. (1984). *Computation and cognition.* Cambridge, MA: MIT Press.

Quian Quiroga, R., Reddy, L., Kreiman, G., Koch, C., & Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, *435*(7045), 1102–7.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *International Conference on Machine Learning* (pp. 1–8). New York, New York, USA: ACM Press.

Rao, R. P. N. (2004). Bayesian Computation in Recurrent Neural Circuits. *Neural Computation*, *16*(1), 1–38.

Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*(1), 79–87.

Rastle, K., Harrington, J., & Coltheart, M. (2002). 358,534 nonwords: The ARC nonword database. *The Quarterly Journal of Experimental Psychology: Section A*, *55*(4), 1339–1362.

Reicher, G. M. (1969). Perceptual recognition as a function of meaningfulness of stimulus material. *Journal of Experimental Psychology*, *81*(2), 275–280.

Reichert, D. P., Series, P., & Storkey, A. (2013). Charles Bonnet syndrome: evidence for a generative model in the cortex? *PLoS Computational Biology*, *9*(7), e1003134.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–25.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.

Rumelhart, D. E. (1977). Toward an interactive model of reading. In S. Dornic (Ed.), *Attention and performance VI* (pp. 573–603). Hillsdale, NJ: Erlbaum.

Rumelhart, D. E., Hinton, G. E., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Rumelhart, D. E., & McClelland, J. L. (1986a). On learning the past tenses of english verbs. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Volume 2: Psychological and Biological Models* (pp. 216–271). Cambridge, MA.

Rumelhart, D. E., & McClelland, J. L. (1986b). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations* (Vol. 1). Cambridge, MA: MIT Press.

Sadeghi, Z., & Testolin, A. (n.d.). *Farsi character recognition with deep neural networks and transfer learning*.

Saffran, J. R. (2001). Words in a sea of sounds: the output of infant statistical learning. *Cognition*, *81*(2), 149–69.

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926–8.

Saffran, J. R., & Wilson, D. P. (2003). From Syllables to Syntax: Multilevel Statistical Learning by 12-Month-Old Infants. *Infancy*, *4*(2), 273–284.

Salakhutdinov, R., & Hinton, G. E. (2012). An efficient learning procedure for deep Boltzmann machines. *Neural Computation*, *2006*, 1967–2006.

Sang, E. F. T. K., & Nerbonne, J. (1999). Learning simple phonotactics. In *Proceedings of the Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Processing, ML2 workshop at IJCAI* (Vol. 99, pp. 41–46).

Selfridge, O. (1959). Pandemonium: A paradigm for learning. *Proceedings of the Symposium on Mechanisation of Thought Processes*, *1*, 511–529.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(3), 411–26.

Share, D. L. (1995). Phonological recoding and self-teaching: sine qua non of reading acquisition. *Cognition*, *55*(2), 151–218; discussion 219–26.

Sharma, G., & Martin, J. (2008). MATLAB®: A Language for Parallel Computing. *International Journal of Parallel Programming*, *37*(1), 3–36.

Sibley, D. E., Kello, C. T., Plaut, D. C., & Elman, J. L. (2008). Large-Scale Modeling of Wordform Learning and Representation. *Cognitive Science*, *32*(4), 741–754.

Sillito, A. M., Cudeiro, J., & Jones, H. E. (2006). Always returning: feedback and sensory processing in visual cortex and thalamus. *Trends in Neurosciences*, *29*(6), 307–16.

Simoncelli, E., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience*, *24*, 1193–1216.

Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory. In *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 194–281).

Stoianov, I. (1999). Recurrent Autoassociative Networks. In L. Medsker (Ed.), *Recurrent Neural Networks: Design and Application* (pp. 205–242). CRC Press.

Stoianov, I., & Zorzi, M. (2012). Emergence of a "visual number sense" in hierarchical generative models. *Nature Neuroscience*, *15*(2), 194–6.

Stoianov, I., Zorzi, M., Becker, S., & Umiltà, C. (2002). Associative arithmetic with Boltzmann Machines: The role of number representations. In J. Dorronsoro (Ed.), *Lecture Notes in Computer Science: ICANN 2002* (pp. 277–283). Berlin: Springer.

Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). The missing memristor found. *Nature*, *453*(7191), 80–3.

Sutskever, I., Hinton, G. E., & Taylor, G. (2008). The recurrent temporal restricted Boltzmann machine. *Advances in Neural Information Processing Systems*, *20*, 1601–1608.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning*. Cambridge, MA: MIT Press.

Tanaka, K. (1996). Inferotemporal cortex and object vision. *Annual Review of Neuroscience*, *19*, 109–139.

Taylor, G., Hinton, G. E., & Roweis, S. (2011). Two Distributed-State Models For Generating High-Dimensional Time Series. *Journal of Machine Learning Research*, *12*, 1025–1068.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: statistics, structure, and abstraction. *Science*, *331*(6022), 1279–85.

Testolin, A., Sperduti, A., Stoianov, I., & Zorzi, M. (2012). Assessment of Sequential Boltzmann Machines on a Lexical Processing Task. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 275–280). Bruges, Belgium.

Testolin, A., Stoianov, I., De Filippo De Grazia, M., & Zorzi, M. (2013). Deep unsupervised learning on a desktop PC : A primer for cognitive scientists. *Frontiers in Psychology*, *4*(May), 251.

Testolin, A., Stoianov, I., Sperduti, A., & Zorzi, M. (n.d.). *Learning Orthographic Structure with Sequential Generative Neural Networks*.

Testolin, A., Zanforlin, M., De Filippo De Grazia, M., 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. In *IEEE IFIP Annual Mediterranean Ad Hoc Networking Workshop*. Piran (SL).

Tieleman, T. (2010). Gnumpy: an easy way to use GPU boards in Python. *Technical Report UTML TR 2010-002, University of Toronto*.

Toro, J. M., & Trobalón, J. B. (2005). Statistical computations over a speech stream in a rodent. *Perception & Psychophysics*, *67*(5), 867–75.

Van Hateren, J. H., & van der Schaaf, a. (1998). Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings. Biological Sciences / The Royal Society*, *265*(1394), 359–66.

Von Foerster, H. (1984). *Observing systems.* Seaside: Intersystems Publications.

Von Helmholtz, H. (1925). Physiological optics. *Optical Society of America*, *3*(318).

Wandell, B. a. (2011). The neurobiological basis of seeing words. *Annals of the New York Academy of Sciences*, *1224*, 63–80.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*.

Widrow, B., Greenblatt, A., Kim, Y., & Park, D. (2013). The No-Prop algorithm: a new learning algorithm for multilayer neural networks. *Neural Networks : The Official Journal of the International Neural Network Society*, *37*, 182–8.

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. In *IRE WESCON Convention Record* (pp. 96–140).

Wiener, N. (1948). *Cybernetics or Control and Communication in the Animal and the Machine. Vol. 25. MIT press, 1965.* MIT Press, Cambridge, MA.

Williams, R., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *280*(1987), 1–10.

Wilson, D. R., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429–51.

Winsberg, E. (2009). Computer Simulation and the Philosophy of Science. *Philosophy Compass*, *4*(5), 835–845.

Yuille, A., & Kersten, D. (2006). Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, *10*(7), 301–8.

Ziegler, J. C., Perry, C., & Zorzi, M. (2014). Modelling reading development through phonological decoding and self-teaching: implications for dyslexia. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *369*(December 2013).

Zipser, D., & Andersen, R. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*.

Zorzi, M. (2005). Computational models of reading. In G. Houghton (Ed.), *Connectionist models in cognitive psychology* (pp. 403–444). London: Psychology Press.

Zorzi, M. (2010). The connectionist dual process (CDP) approach to modelling reading aloud. *European Journal of Cognitive Psychology*, *22*(5), 836–860.

Zorzi, M., Houghton, G., & Butterworth, B. (1998). Two routes or one in reading aloud? A connectionist dual-process model. *Journal of Experimental Psychology: Human Perception and Performance*, *24*(4), 1131–1161.

Zorzi, M., Stoianov, I., & Umiltà, C. (2005). Computational modeling of numerical cognition. In J. Campbell (Ed.), *Handbook of mathematical cognition* (pp. 67–84). New York, USA: Psychology Press.

Zorzi, M., Testolin, A., & Stoianov, I. (2013). Modeling language and cognition with deep unsupervised learning: a tutorial overview. *Frontiers in Psychology*, *4*(August), 515.

Zou, Y., Testolin, A., & McClelland, J. L. (n.d.). *Numerosity Judgement in Deep Neural Networks: Roles of Experience and Corpus Stucture in Development*.