Network computations in artificial intelligence

Network Computations in Artificial Intelligence

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op donderdag 29 juni 2017 om 16.00 uur

door

Decebal Constantin Mocanu

geboren te Mizil, Roemenië

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter:	prof.dr.ir. A.B. Smolders
Promotor:	prof.dr. A. Liotta
Co-promotor:	dr. G. Exarchakos
Leden:	prof.dr. P. Tran-Gia (Julius Maximilian University of Würzburg, Germany)
	dr. G. Di Fatta (University of Reading, UK)
	dr. M. Gibescu
	prof.dr. K. Tuyls (University of Liverpool, UK)
	prof.dr. M. Pechenizkiy

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN: 978-90-386-4305-2
NUR: 984
Title: Network Computations in Artificial Intelligence
Author: Decebal Constantin Mocanu
Eindhoven University of Technology, 2017.

Typeset using LaTeX, printed by Ipskamp Printing, Enschede, the Netherlands.

"*The greatest danger for most of us is not that our aim is too high and we miss it, but that it is too low and we reach it.*"

(*most probably*) Michelangelo

Acknowledgements

Four years passed extremely fast, bringing new people, places, challenges, disappointments, satisfactions, sadness, happiness, and a lot of knowledge. They end up with this PhD thesis. This, in fact, it is not an end, but a new beginning. I would like to take this opportunity to thank to many special persons who have been there for me, helping me to arrive at this milestone. There are many things to say, but I will mention just a few.

First, I would like to thank to my promoter, Antonio Liotta, for giving me the freedom of pursuing my own research plans and ideas, for supporting them, while having always a critical eye on my proposals. Besides that, I would like to thank him for his advices, collaboration, and our open discussions on both, professional and personal life. I continue by thanking to my co-promoter, Georgios Exarchakos, for our discussions on network science, when I first came across this domain. Also, I thank him for advising me to prepare a proposal for my PhD topic in the first months of my PhD. Some of those initial ideas are reflecting now in this thesis.

My special thanks go to my PhD colleague, Maria Torres Vega, for her unconditioned support during these four years, being a good collaborator and a great friend.

Furthermore, I would like to thank to the people from the Smart Networks group: Roshan Kotian, Hedde Bosman, Stefano Galzarano, Michele Chincoli, and Tim van der Lee. Also, I would like to express my appreciations to all the people from the ECO group, led by Ton Koonen. Many thanks to José Hakkens who helped me with all the bureaucratic matters.

To proper acknowledge this moment, I have to go back in time and to express my gratitude to my master thesis supervisor from Philips Research, Dietwig Lowet, who introduced me to deep learning. Moreover, I would like to thank to my former professors and collaborators from Maastricht University, Karl Tuyls, Kurt Driessens, Evgueni Smirnov, Mykola Pechenizkiy, and Gerhard Weiss, whom in the last period of my master studies have introduced me to academic research or, along time, gave me confidence to continue my path. Furthermore, I would like to thank to Haitham Bou Ammar, a friend and collaborator, from which I learned what means a PhD and how to handle it, before to start one. Going back even more through time, my special thanks go to my informatics teacher from high school, Luminița Râpeanu, whose teaching style opens in me the wish to pursue a career which combines computer science and mathematics.

Parts of this thesis are based on my experience gained in the research visits that I performed, and for which I am profoundly grateful to my four host groups. I enjoyed working and interacting with Eric Eaton and the people from his group at University of Pennsylvania, in my first contact with the USA academic world.

Then, at Julius Maximilian University of Würzburg in the group of Phuoc Tran-Gia, I had the opportunity to give my first international invited talk. From the same group, I thank to Thomas Zinner for opening my eyes at the begin of my PhD, and for becoming a friend. During my third research visit, at the University of Texas at Austin in the Webber Energy group led by Michael Webber and LARG group led by Peter Stone, besides many interesting experiences, I re-understood the importance of setting high goals. The latter visit would not been possible without the support of Madeleine Gibescu, and I would like to take this opportunity and thank her. Many thanks go to all my co-authors for the excellent collaborations that we had.

I would like to thank also to my friends for the good moments spent together and for their friendship. Costel, Iancu, Cătă, Pitu, Nas, Mișu, Urlă, Monica, Vali, Laura, Rică, Silvia, Bobo, Trotter, Gonzalo and all the ones not explicitly mentioned here, thank you so much.

Finally, to my family, who unconditionally supported me in all the moments of my life, I would like to express my gratitude. Many thanks to my parents, Toia and Traian, who raised me, tried to make their best for me, learned me to dream more, and always support me in my dreams. Then, to my family in law, Despina, Virgil, Maria, Emilia, Alex, and Andrei for the good times over the years and for believing in me. In this moments, some of my thoughts go to my grandparents, Elena, Anghel, Ioana, and Stere, who sadly passed away. The last but not the least, there are no words to express my appreciations and gratitude to Elena, my wife, for enlightening my life. She is always there for me, from moments of deep sadness to moments of great happiness, or for discussing theoretical research problems. Elena, thank you!

Decebal Constantin Mocanu
Eindhoven, the Netherlands, 2017

Summary

From understanding fundamental principles behind the world near us, to advancing state-of-the-art artificial intelligence to solve real-world problems, networks have shown to be a very powerful tool. For example, from a physics perspective, the amazing "structures of networks of networks" at micro and macro-scale, from the vigintillions of interacting atoms in the observable universe to the billions of persons who live on Earth, are studied using network science by the means of complex networks. From a computer science perspective, artificial neural networks (which are inspired to biological neural networks) represent nowadays the state-of-the-art in object recognition problems, zero-sum game playing (e.g. Chess and Go) and so on. Even when successful in real-world problems, it is intuitive that network algorithms may be affected by various scalability issues. This thesis starts from considering practical problems from emerging large-scale systems, which pose hard scientific challenges. Then we extrapolate fundamental challenges, including: (1) how to reduce computational complexity when assessing the importance of all the elements of a complex network, i.e. nodes and links; (2) how to reduce the excessive memory requirements in Artificial Neural Networks (ANNs) when they perform on-line learning; and (3) how to reduce computational complexity when training and exploiting artificial neural networks. All of these, with the hard constraint of not loosing accuracy when comparing with the traditional algorithms for the specific problem. These challenges led us to make fundamental theoretical contributions in the areas of artificial intelligence and network science, building new bridges between them, as follows.

Polylogarithmic centrality computations in complex networks. To compute the centrality of all elements (i.e. nodes and links) in a complex network is a difficult problem due to: (1) the difficulty of unveiling the hidden relations between all networks elements; (2) the computational time of state-of-the-art methods which many times are not practical in real-world networks that have in excess of billions of nodes. Herein, we introduce a new class of fully decentralized stochastic methods, inspired by swarm intelligence and human behavior, to compute the centralities of all nodes and links simultaneously in a complex network. The parallel time complexity of this approach is on the polylogarithmic scale with respect to the number of nodes in the network, while its accuracy is similar, and many times even better, than state-of-the-art centrality metrics. To give an impression on the magnitude of the computational problem at hand, if we were to consider one trillion Internet of Things devices (each one running the proposed protocol, over an unloaded network), and a transmission rate of 1 message per millisecond, then the centrality of all network elements (devices and the relations between them) would

be computed in less than 22 seconds. As a comparison, by using other state-of-the-art centrality metrics for the same problem, one would need (perhaps) months to compute the results.

Generative Replay: towards memory free online learning with artificial neural networks. Online learning with artificial neural networks is in many cases difficult due to the need of storing and relearning large amount of previous experiences. This limitation can be partially surpassed using a mechanism conceived in the early 1990s, named experience replay. Traditionally, experience replay can be applied in all types of ANN models to all machine learning paradigms (i.e. unsupervised, supervised, and reinforcement learning). Recently, it has contributed to improving the performance of deep reinforcement learning. Yet, its application to many practical settings is still limited by the excessive memory requirements, necessary to explicitly store previous observations. From a biological sense of memory, the human brain does not store all observations explicitly; it dynamically generates approximate reconstructions of those experiences for recall. Inspired by this biological fact, to remedy the experience replay downside, we propose a novel approach, dubbed generative replay. Generative replay uses the generative capabilities of restricted Boltzmann machines to generate approximations of past experiences, instead of recording them, as experience replay does. Thus, the restricted Boltzmann machine can be trained online, and does not require the system to store any of the observed data points. Moreover, generative replay is a generic concept which may be exploited in many combinations with ANNs to perform on-line learning.

Quadratic parameter reduction in artificial neural networks. Almost all of the artificial neural networks used nowadays contain fully connected layers which have a quadratic number of connections with respect to the number of neurons. This type of fully connected layers contain the most of the neural network connections. Because the weight corresponding to each connection has to be carefully optimized during the learning process, this leads to increased computational requirements, proportionally to the number of connections that need to be optimized. Inspired by the fact that biological neural networks are sparse, and even more, they usually have small-world and scale-free topologies, in this thesis we show that a striking amount of the connections from the fully connected layer in artificial neural networks is actually redundant. Furthermore, we demonstrate that we can safely decrease this number of connections from a quadratic relation to a linear relation, with respect to the number of neurons, at no decrease in accuracy (many times, even with an increase in accuracy). It is worth highlighting that the connections reduction is done in the design phase of the neural network, before training. First, we use a fixed scale-free connectivity pattern. Then, we take this idea further and, starting from a fixed sparse connectivity pattern and then using an evolutionary process during the training phase of the ANN model, we are capable to reach even better performance in terms of accuracy. Our results show that it is possible to replace the fully connected layers in artificial neural networks with quadratically faster counterparts in both phases, training and exploitation, and lead to the possibility of building ANN models with at least billions of neurons.

Thus, by looking at the synergy between network science, artificial intelligence, and biological neural networks, in this thesis we have been able to push the scalability bounds of various networks algorithms much beyond their state-of-the-art. Auxiliary, we have pioneered the bidirectional bridge between complex networks

and artificial intelligence. While most effort so far was put into trying to solve complex networks problems using artificial intelligence, we showed for the first time that artificial intelligence methods can be improved using complex networks paradigms.

Contents

CHAPTER 1

Introduction

Traditionally science is done using the reductionism paradigm. Artificial intelligence does not make an exception and it follows the same strategy. At the same time, network science tries to study complex systems as a whole. This Ph.D. thesis takes an alternative approach to the reductionism strategy, with the aim to advance both fields, advocating that major breakthroughs can be made when these two are combined.

1.1. Motivation

Most of the science done throughout the human evolution uses the traditional reductionism paradigm, which attempts to explain the behavior of any type of system by zooming in on its constituent elements [**11**] and by summing their behavior. Consequently, nowadays we have an abundance of specializations and specialized people but few scientist study complex systems, which are in fact all around us. In my work, I do not claim reductionism to be wrong. On the contrary, it has been the basis of scientific advances throughout centuries of methodic investigation. Yet, my ambition is to understand the hidden properties that underlie complexity.

The limitations of reductionism were hinted millenniums ago by the ancient Greeks, Aristotle wrote in Metaphysics that *"The whole is more than the sum of its parts"*. At a first thought, the whole should be the sum of its parts. Still, some times we do not know all the parts and, in many cases, it may even be difficult to identify all those parts, let alone their mutual interdependencies. For instance, think about the gravitational waves. Gravity was first postulated by Isaac Newton in the 17^{th} century. Yet, the gravitational waves could have not considered in his theory, since that would have assumed that physical interactions propagate at infinite speed. Still, it was not until more than two centuries later, that Albert Einstein has intuited and predicted the existence of gravitational waves [**56**]; and it took about another century of great technological advancements before the existence of gravitational waves was proven [**3**].

To overcome the limitations of reductionism, the 'complex systems' paradigm aims to study the systems and their mutual interactions as a whole, which requires a multidisciplinary research, as depicted in Figure 1.1. This approach was first pioneered by the Santa Fe institute [**112**].

A complete theory of complexity is very hard to devise, but Network Science (NS) offers many of the required mathematical tools (e.g. complex networks) necessary to overpass reductionism [**17**]. Complex networks are graphs with non-trivial topological features, which are typical in many real world systems from a variety

This chapter is partly based on: D.C. Mocanu: *On the synergy of network science and artificial intelligence*, International Joint Conference on Artificial Intelligence (IJCAI), 2016, New York, USA.

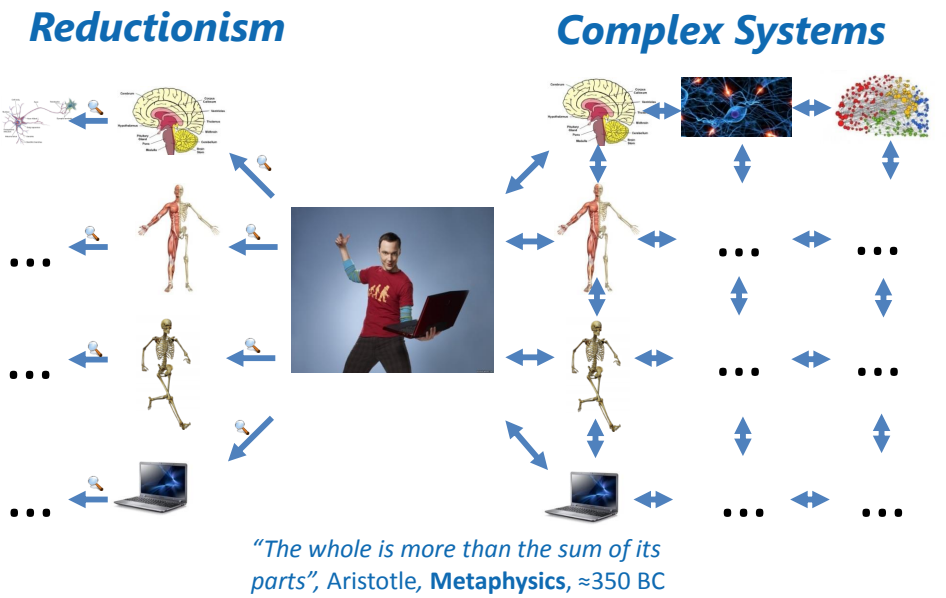


Figure 1.1 – Illustration of the reductionism and complex systems paradigms. It may be observed that while in the reductionism paradigm the main idea is to zoom in onto the various components of a system, the main emphasis in the complex systems paradigm is on unveiling connections among the various components and grasping the overall system behavior.

of research fields (e.g. neuroscience, astrophysics, biology, epidemiology, social and communication networks) [**157**].

At the same time, while the NS community has been trying to use Artificial Intelligence (AI) techniques to solve various NS open questions, such as in [**169**], the AI community has largely ignored the latest findings in network science. We argue that AI tends to follow the principles of reductionism and that new breakthroughs will need to go beyond it. In this thesis, we explore the potential arising from combining NS with AI, with emphasis on artificial neural networks [**110**] and evolutionary computation [**62**]). We set out with two long term research goals: (1) to better understand the fundamental principles behind the world near us, which may be modeled in amazing structures of networks of networks at micro and macro-scale, from the vigintillions of interacting atoms in the observable universe to the billions of persons in a social network; and (2) to advance the artificial intelligence field. These will ultimately help improving the general well-being of the human society, which is increasingly dependent upon intelligent software in complex systems of systems.

The remainder of this chapter is organized as follows. Sections 1.2 and 1.3 present background knowledge on network science and artificial intelligence, respectively, for the benefit of the non-specialist reader. Section 1.4 briefly introduces some of our novel approaches to solve real-world problems using network science and artificial intelligence. Section 1.5 discusses some common issues in state-of-the-art networks algorithms, and details the research questions addressed

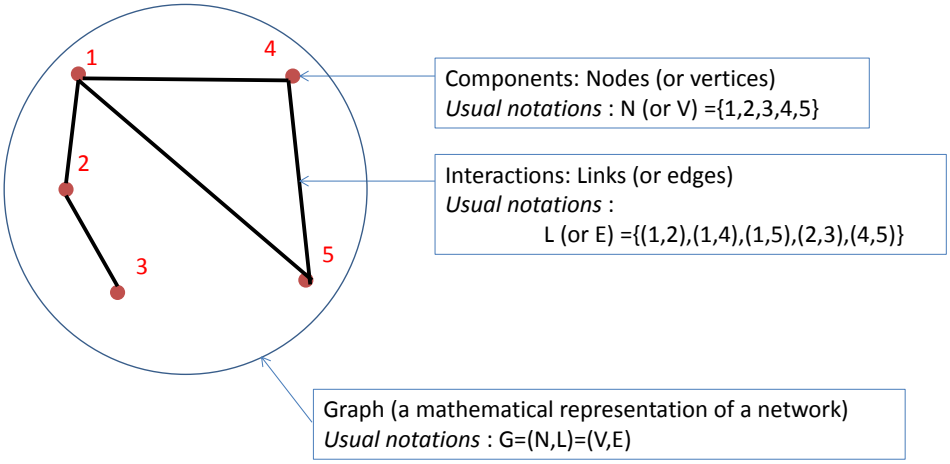


Figure 1.2 – Schematic representation of a complex network.

in this thesis. Section 1.6 presents an outline of this thesis contributions. Finally, Section 1.7 provides a guideline to the reader.

1.2. Network science

Network science is the academic field which studies complex networks [**157**, **192**]. Any real-world network formalized from a graph theoretical perspective is a complex network. For example, such networks can be found in many domains from technical to social ones, such as telecommunication networks [**8**, **134**], transportation networks [**46**], biology [**90**, **224**] (e.g. biological neural networks, protein interactions), neuroscience [**60**, **168**, **191**], astrophysics [**85**], artificial intelligence [**144**] (e.g. artificial neural networks), semantic networks, social networks [**63**, **87**], to mention but a few. In the study of complex networks, network science uses knowledge from many academic fields, such as mathematics (e.g. graph theory), physics (e.g. statistical mechanics), statistics (e.g. inferential modeling), computer science (e.g. data visualization, data mining), sociology (e.g. social structure) and so on.

Formally, a complex network is a graph with non-trivial properties, e.g. scale-free [**218**] or small-world [**19**]. It is composed by actors and the interactions between these actors. In general, the actors are named nodes (or vertices), and the interactions between them are named links (or edges). A schematic representation of a complex network is depicted in Figure 1.2. The usual notation for the graph is $G = (N, L) = (V, E)$, where N (or V) represents the set of nodes, and L (or E) represents the set of links. There are many open research questions in network science some of them coming from graph theory, others coming from the real-world challenges associated with the complex networks, such as community detection [**61**], controlling networks dynamics [**44**], finding the most influential nodes in a network using centrality metrics [**12**], spreading of the information through a network [**120**], and so on. What makes it difficult for the state-of-the-art algorithms to cope with these challenges is the size of complex networks, which may span from small networks with tens on nodes (e.g. a small dolphin community from New Zealand [**125**]), to medium size networks (e.g. Facebook users), up to

extremely large scales (e.g. the vigintillions of interacting atoms in the observable universe).

A complete review of the fundamental knowledge from network science and its open questions is beyond the goals of this thesis. The interested reader is referred to the specialist literature for a deeper understanding, such as [**18**, **157**]. Further on, the network science background necessary to understand the research done in this thesis is introduced gradually, where it is needed, in each chapter.

1.3. Artificial intelligence

Artificial intelligence is a subfield of computer science, which uses the concept of software intelligent agents to incorporate intelligence into machines [**88**]. The main research directions addressed by artificial intelligence are, mainly, knowledge representation, perception, learning, reasoning, and planning. These are, in fact, inspired to corresponding human cognitive functions. In this thesis, we address in more details, two subfields of artificial intelligence, namely machine learning and evolutionary computations.

Machine learning studies how to get machines to learn how to function directly from the data, rather than explicitly programming each individual instructions [**178**]. There are three main paradigms in machine learning:

(1) Supervised learning [**78**] - aims to build a general function (or model) based on data input-output pairs. Specifically, the function learns how to estimate any output based on its corresponding input. This type of learning assumes the existence of labeled data, where each data point (the input) has associated a label (the output) generated by expert knowledge. There are two main types of problems within the supervised learning paradigm, i.e. classification (where the output has discrete values), and regression (where the output has continuous values).

(2) Unsupervised learning [**78**] - aims to build functions (or models) which are capable by themselves to extract useful information from the input data, without having its corresponding output. Example of unsupervised learning problems are: clustering, density estimation, and dimensionality reduction.

(3) Reinforcement learning [**193**] - is a special type of learning inspired by psychology. Herein, an agent interacts dynamically with an environment having the goal of learning by itself how to take the optimal action in a specific state (situation) without knowing the ground truth. To learn the optimal choices, as the agent navigates through the environment it is provided with positive or negative feedback (also named reward) as a result of its actions.

Each learning paradigm can perform its own specific models. Among these, in the scope of this thesis, we explore Artificial Neural Networks (ANNs). ANNs are mathematical models, inspired by biological neural networks, which can be used in tandem with all three machine learning paradigms. ANNs are extremely versatile and powerful, as demonstrated by the remarkable success registered recently, for instance by Deep Artificial Neural Networks (in short, deep learning), which are the last generation of ANNs [**110**]. These have been demonstrated to be able to perform all three machine learning paradigms in many domains, from computer

vision [**110**] to game playing [**133**, **183**]. Briefly, just as their biological counterparts, ANNs comprise neurons and weighted connections between those neurons. Depending upon their purposes and architectures, several models of ANNs have been introduced, including restricted Boltzmann machines [**187**], multi layer perceptron [**173**], convolutional neural networks [**111**], and recurrent neural networks [**74**], to mention just a few. In general, working with ANN models involves two phases: (1) training (or learning), in which the weighted connections between neurons are optimized using various algorithms (e.g. backpropagation combined with stochastic gradient descent [**27**, **174**] or contrastive divergence [**81**]) to minimize a given loss function; and (2) exploitation, in which the optimized ANN model is used to fulfill its purpose.

Evolutionary computation [**55**] represents a class of algorithms inspired by the principles of biological evolution that tries to solve global optimization problems. In general, evolutionary algorithms have a metaheuristic or stochastic behavior. In a very broad sense, the basic idea is that, starting from a random generated initial population (set) of possible solutions, this population is refined during generations, mimicking the natural processes of evolution. At each generation the most unfitted solutions for the goal of the algorithm are removed, while new solutions (which can either derive from a measure of fitness or be picked randomly), are iteratively added to the general population. This procedure continues until the population contains acceptable solutions, aiming towards global optimum convergence. There are many types of evolutionary computing algorithms, mainly categorized by their biological counterparts, including genetic algorithms [**132**] (inspired by natural evolution), swarm intelligence [**22**] (inspired by the collective behavior of organisms living in swarms), ant colony optimization [**42**] (inspired by ants behavior), and so on.

A full review of artificial intelligence goals and methods is much beyond the goals of this thesis. The interested reader is referred to specialized books for more information [**21**, **78**, **193**]. For the benefit of the non-specialist reader, the background knowledge required in this thesis is outlined where needed.

1.4. Real-world challenges

In this section, we consider practical, real-world problems, which pose hard scientific challenges, explaining how we have addressed them - either through novel solutions or through novel application of existing methods.

1.4.1. Wireless sensor networks

With the emergence of sensors with wireless capability, most of current sensor networks consist of a collection of wirelessly interconnected units, each of them with embedded sensing, computing and communication capabilities [**115**]. Such sensor networks are referred to as Wireless Sensor Networks (WSNs) [**102**]. Due to their versatility, WSNs have been employed in a wide range of sensing and control applications [**49**], such as smart traffic control, environmental monitoring, security surveillance, and health-care [**64**]. As a consequence of cost, energy and spectrum constraints [**100**] sensors are prone to failure (hardware and transmission), as well as to data corruption [**164**]. A typical approach to tackle these issues is through smart autonomic methods [**26**, **65**, **118**, **170**, **214**].

1.4.1.1. *Redundancy reduction in WSN.* The dense, unpredictable deployment of sensors leads to substantial data and networks [**121**]. In these situations, identifying the redundant sources and connections can save considerable resources (energy, communication spectrum, data processing and storage). In turn, this can extend the network life-time and scale [**69**, **86**, **119**]. Redundancy reduction requires that the network stays fully connected to let the flow of information pass between any communication points.

In the scope of these arguments, in [**148**], we take advantage of the latest theoretical advances in complex networks, introducing a method that simplifies network topology based on centralized centrality metrics computations [**157**]. We can detect the redundant network elements (both nodes and links), which allows switching them off safely, that is without loss in connectivity. The experiments performed on a wide variety of network topologies with different sizes (e.g. number of nodes and links), using different centralized centrality metrics, validate our approach and recommend it as a solution for the automatic control of WSNs topologies during the exploitation phase of such networks to optimize, for instance, their life time.

1.4.1.2. *Predictive power control in WSN.* Besides that, prompt actions are necessary to achieve dependable communications and meet quality of service requirements in WSNs. To this end, the reactive algorithms used in the literature and standards, both centralized and distributed ones, are too slow and prone to cascading failures, instability and sub-optimality. In [**38**] we explore the predictive power of machine learning to better exploit the local information available in the WSN nodes and make sense of global trends. We aimed at predicting the configuration values that lead to network stability. We adopted Q-learning, a reinforcement learning algorithm, to train WSNs to proactively start adapting in face of changing network conditions, acting on the available transmission power levels. The results demonstrate that smart nodes lead to better network performance with the aid of simple reinforcement learning.

1.4.2. Computer security

Computer security [**67**] handles the protection of IT systems from malicious attacks. The aim is to avoid the stealing or damaging of their hardware, software, and of the information which they contain, likewise the misdirection of the services which they provide. In this area we have only just started to explore the benefits of artificial intelligence, as outlined below.

1.4.2.1. *ABAC Policy Mining from Logs.* Different languages may be used to specify security policies through a number of constructs. These are based on an underlying access control model that captures the security requirements. In other words, the selection of the policy language, and thus the model, determine expressiveness in encoding rules and simplicity in administration. Among the various models, Attribute-Based Access Control (ABAC) [**43**] has been shown to provide very expressive constructs; various tools have been developed to assist policy specifications with them [**202**,**203**]. In order to assist policy administrators when specifying ABAC policies, a particularly useful approach is to infer access control rules from existing logs.

In [**146**] we started exploring how to merge traditional AI approaches (i.e. inductive logic programming [**226**]) with deep learning techniques to infer access control rules. We take advantage of the excellent generalization capabilities of restricted Boltzmann machines [**187**] as density estimators to propose a technique that can produce a set of suitable candidate rules in a binary vector format, based on the knowledge extracted by RBMs from the processed logs. Further on, the candidate binary rules may be translated to the inductive logic programming format to take advantage of a human readable format.

1.4.3. Transfer learning

Reinforcement Learning (RL) methods often learn new problems from scratch. In complex domains, this process of tabula rasa learning can be prohibitively expensive, requiring extensive interaction with the environment. Transfer learning [**198**] provides a possible solution to this problem by enabling reinforcement learning agents to reuse knowledge from previously learned source tasks when learning a new target task.

1.4.3.1. *What to transfer.* In situations where the source tasks are chosen incorrectly, inappropriate source knowledge can interfere with learning through the phenomenon of negative transfer. To avoid this drawback, transfer learning agents must be able to automatically identify source tasks that are most similar to and helpful for learning a target task. In RL, where tasks are represented by Markov Decision Processes (MDPs), agents could use an MDP similarity measure to assess the relatedness of each potential source task to the given target. This measure should: (1) quantify the similarity between a source and a target task, (2) be capable of predicting the probability of success after transfer, and (3) be estimated independently from sampled data.

In [**10**], we formulate for the first time a mathematical framework to achieve these goals successfully, proposing a novel similarity measure, based on restricted Boltzmann machines, dubbed RBDist. This measure works for MDPs within a domain and it can be used to predict the performance of transfer learning. Moreover, this approach does not require a model of the MDP, but can estimate this measure from samples gathered through agents interaction with the environment. We demonstrate that the proposed measure is capable of capturing and clustering dynamical similarities between MDPs with multiple differences, including different reward functions and transition probabilities. Our experiments also illustrate that the initial performance improvement on a target task from transfer is correlated with the proposed measure - as the measured similarity between MDPs increase, the initial performance improvement on the target task similarly increases.

1.4.3.2. *How to transfer.* In transfer learning for RL, the source task and target task may differ in their formulations. In particular, when these have different state and/or action spaces, an inter-task mapping [**199**], which describes the relationship between the two tasks is needed. In [**28**] we introduce an autonomous framework for learning inter-task mappings based on three-way restricted Boltzmann machines, dubbed FTrRBM. The results demonstrate that FTrRBMs are capable of: (1) automatically learning an inter-task mapping between different MDPs, (2) transferring informative samples that reduce the computational complexity of a sample-based RL algorithm, and (3) transferring informative instances

which reduce the time needed for a sample-based RL algorithm to converge to a near-optimal behavior.

1.4.4. Computer vision

Computer vision [**16**] is a broad field which aims at making computers that extract high-level concepts from images or videos. There are many open research questions in this area, and in our work we have targeted a few of them, as follows.

1.4.4.1. *Human activity recognition.* Accurate activity recognition is needed in many domains [**36**, **228**], such as robotic support for elderly people [**122**, **127**]. This is a very difficult problem due to the continuous nature of typical activity scenarios, which makes the task highly similar to time series prediction. In [**141**] we propose a novel machine learning model, namely Factored Four Way Conditional Restricted Boltzmann Machine (FFW-CRBM) capable of both classification and prediction of human activity in one unified framework. An emergent feature of FFW-CRBM, so called self auto evaluation of the classification performance, may be very useful in the context of robotic companions. It allows the machine to autonomously recognize when an activity is undetected, triggering a retraining procedure. Due to the complexity of the proposed machine, the standard training method for DL models is unsuited. As a second contribution, in the same paper, we introduce Sequential Markov chain Contrastive Divergence (SMcCD), an adaptation of Contrastive Divergence (CD) [**81**]. We illustrate the efficacy and effectiveness of the model by presenting results performed on two sets of experiments using real world data originating from: (1) our previous developed smart companion robotic platform [**123**], and (2) a benchmark database for activity recognition [**161**].

1.4.4.2. *3D trajectories estimation.* Estimating and predicting trajectories in three-dimensional spaces based on two-dimensional projections available from *one* camera source is an open problem with wide-ranging applicability including entertainment [**182**], medicine [**171**], biology [**126**], physics [**89**], etc. Unfortunately, solving this problem is exceptionally difficult due to a variety of challenges, such as the variability of states of the trajectories, partial occlusions due to self articulation and layering of objects in the scene, and the loss of 3D information resulting from observing trajectories through 2D planar image projections.

In [**142**] we, first, propose the use of FFW-CRBMs to estimate 3D trajectories from their 2D projections, while at the same time being also capable to classify those trajectories. To achieve a better performance, we then propose an extension of FFW-CRBMs, dubbed Disjunctive FFW-CRBMs (DFFW-CRBMs). Our extension refines the factoring of the four-way weight tensor from FFW-CRBMs. This yields the sufficiency of a reduced training dataset for DFFW-CRBMs to reach similar classification performance to state-of-the-art methods while at least doubling the performance on real-valued predictions. Specifically, DFFW-CRBMs require limited labeled data (less than 10 % of the overall dataset) for: (1) simultaneously classifying and predicting three-dimensional trajectories based on their two-dimensional projections, and (2) accurately estimating three-dimensional postures up to an arbitrary number of time-steps in the future. We validate our

approach in two sets of experiments: (1) predicting and classifying simulated three-dimensional ball trajectories (based on a real-world physics simulator) thrown from different initial spins, and (2) human activity recognition.

1.4.5. Quality of experience

Quality of Experience (QoE) [**131**,**222**] aims at assessing the quality perceived by a user, while experiencing a service (e.g. video streaming services, web browsing, phone or video calls, server based enterprise software at the work environment and so on). Even though QoE is human centric, in general, due to the exponential increase of services, it is not practical to employ humans to assess the services quality. Thus, objective computational methods capable to assess the quality of those services such as humans do are needed [**118**].

1.4.5.1. *Objective image quality assessment.* Objectively measuring the quality degradation of images yielded by various impairments of the communication networks during a service is a difficult task, as there is often no original images to be used for direct comparisons. To address this problem, in [**136**] we proposed a novel reduced-reference QoE method, dubbed Restricted Boltzmann Machine Similarity Measure (RBMSim), that measures the quality degradation of 2D images, without requiring the original images for comparisons. Moreover, in [**137**] we take this work further, proposing a new reduced-reference QoE method to measure the quality degradation of 3D images using factored third order restricted Boltzmann machines [**130**], dubbed Q3D-RBM. What is interesting is that both, RBMSim and Q3D-RBM, perform just unsupervised learning taking advantage of RBM performance as density estimator. So, they do not need the ground truth, this being an important advantage for quality of experience methods. The experiments performed on benchmark datasets demonstrate that both methods achieve a similar performance to full reference objective metrics when benchmarked with subjective studies.

1.4.5.2. *Objective video quality assessment.* For obvious reasons, video quality assessment, is more difficult and more important than image quality assessment [**138**, **213**]. In [**145**, **208**–**210**, **212**] we take further our work on images, proposing new no-reference and reduced-reference QoE methods to assess the quality degradation suffered by videos during streaming services. We use various models of artificial neural networks, from restricted Boltzmann machines to deep neural networks, using both unsupervised and supervised learning. The results show that, in general, the variants of artificial neural networks used achieve very good performance, comparable with state-of-the-art objective full-reference metrics for video quality assessment, while not requiring the original videos for comparisons. An example on how to use artificial neural networks to perform objective video quality assessment is described in Appendix A.

1.4.5.3. *Objective quality of experience in enterprise and working environments.* While most of the QoE studies aim at understanding the QoE impact of waiting times in controlled laboratories or in the user's domestic environment, the enterprise and working environments have been largely ignored. This happens due to the IT environment, which is highly complex and hard to analyze, and incurs high costs. In [**23**], by using a non-intrusive application monitoring of response times and subjective user ratings on the perceived application, we employ

deep neural networks and other machine learning models to estimate the users QoE. The results show that we can successfully build machine learning models to estimate the QoE of specific users, but do not allow us to derive a generic model for all users.

1.4.6. Smart grid

The smart grid is a broad intensive research area nowadays, which studies the future of the actual power grid, incorporating knowledge from computer science, information and communication technologies, and machine learning [**152**, **153**]. The ultimate goal is to improve quality of life, while taking into consideration several technological, ecological, and social constraints.

1.4.6.1. *Real-time energy disaggregation in buildings.* Within the smart grid context, the identification and prediction of building energy flexibility is a difficult open question [**151**], paving the way for new optimized behaviors from the demand side. The latest smart meters developments help us to monitor in real-time the power consumption level of the home appliances, with the aim of obtaining an accurate energy disaggregation, as explained next. Due to practical constraints, it is unrealistic to expect that all home appliances are equipped with smart meters. In [**135**] we propose a hybrid approach, which combines sparse smart meters with artificial neural network methods. Using energy-consumption data collected from a number of buildings, we created a database on which we trained two deep learning models, i.e. Factored Four-Way Conditional Restricted Boltzmann Machines (FFW-CRBMs) [**141**] and Disjunctive FFW-CRBM [**142**]. The aim was to show how these methods could be used to accurately predict and identify the energy flexibility of buildings unequipped with smart meters, starting from their aggregated energy values. The experiments performed on a real database, namely the Reference Energy Disaggregation Dataset [**98**], validated the proposed method, showing that Disjunctive FFW-CRBM outperformed FFW-CRBMs on the prediction problem; whereas both were comparable on the classification task.

1.4.6.2. *On-line building energy optimization.* An optimal resource allocation of end-users patterns based on daily smart electrical devices profiles may be used to facilitate the demand response, while taking into consideration the future energy patterns and the supply of variable sources, such as solar and wind. In [**150**] we explore for the first time in the smart grid context the benefits of using deep reinforcement learning, a hybrid type of methods which combines reinforcement learning with deep learning, to perform on-line optimization of scheduling for building energy services. Specifically, we extend two methods, Deep Q-learning and Deep Policy Gradient, to perform optimally multiple actions in the same time. The proposed approach was validated on the large-scale Pecan Street Inc. database. The results show that these on-line energy scheduling strategies could be used to provide real-time feedback to consumers to encourage a more efficient use of electricity.

1.5. Research questions and objective

Following the study of a range of real-world problems, as outlined in Section 1.4, we realized the enormous potential of both network science and machine learning. In all cases, scalability was the key limiting factors. With the aim of

increasing the scalability bounds of various networks algorithms, we extrapolate a number of fundamental challenges, presented below as the *theoretical research questions* of this doctoral thesis:

(1) How to reduce the computational complexity when assessing the importance of all the elements of a complex network, i.e. nodes and links.
(2) How to reduce the excessive memory requirements in artificial neural networks when they perform on-line learning.
(3) How to reduce the computational complexity when training and exploiting artificial neural networks.

In this thesis, while trying to answer to these three research questions, we follow one single common *objective*:

- Any new method, which is to fulfill one of the three research questions above, will have to be comparably as accurate as its state-of-the-art counterparts.

1.6. Thesis contributions and outline

Overall, we have discovered that the key to addressing the three research questions stated above lies in methods that combine artificial intelligence with network science methods, rather than employing them independently [**140**]. We elaborate on this claim through a selection of contributions included in Chapters 2 to 5, while Chapter 6 provides a summary and discussion of the main research findings and presents further research directions. The core thesis contributions are summarized next.

1.6.1. Chapter 2

Polylogarithmic centrality computations in complex networks [**134**, **143**]. To compute the centrality of all elements (i.e. nodes and links) in a complex network is a difficult problem due to: (1) the difficulty of unveiling the hidden relations between all networks elements; (2) the computational time of state-of-the-art methods, which many times are not practical in real-world networks that are in excess of billions of nodes. Herein, we introduce a new class of fully decentralized stochastic methods, inspired by swarm intelligence and human behavior, to compute the centralities of all nodes and links simultaneously in a complex network. The parallel time complexity of this approach is on the polylogarithmic scale with respect to the number of nodes in the network, while its accuracy is similar, and many times even better, than state-of-the-art centrality metrics. To give an impression on the magnitude of the computational problem at hand, if we were to consider one trillion Internet of Things devices (each one running the proposed protocol, over an unloaded network), and a transmission rate of 1 message per millisecond, then the centrality of all network elements (devices and the relations between them) would be computed in less than 22 seconds. As a comparison, by using other state-of-the-art centrality metrics for the same problem, one would need (perhaps) months to compute the results.

1.6.2. Chapter 3

Generative Replay: towards memory-free online learning with ANNs [**147**]. Online learning with artificial neural networks is in many cases difficult due to the need of storing and relearning large amount of previous experiences. This limitation can be partially surpassed using a mechanism conceived in the early 1990s, named experience replay. Traditionally, experience replay can be applied in all types of ANN models to all machine learning paradigms (i.e. unsupervised, supervised, and reinforcement learning). Recently, it has contributed to improving the performance of deep reinforcement learning. Yet, its application to many practical settings is still limited by the excessive memory requirements, necessary to explicitly store previous observations. From a biological sense of memory, the human brain does not store all observations explicitly, but instead it dynamically generates approximate reconstructions of those experiences for recall. Inspired by this biological fact, to remedy the experience replay downside, we propose a novel approach dubbed generative replay. Generative replay uses the generative capabilities of restricted Boltzmann machines to generate approximations of past experiences, instead of recording them, as experience replay does. Thus, the RBM can be trained online, and does not require the system to store any of the observed data points. Furthermore, generative replay is a generic concept which may be used in combination with other types of generative artificial neural network models to serve dynamic approximations of past experiences to any ANN model that performs on-line learning.

1.6.3. Chapters 4 and 5

Quadratic parameter reduction in artificial neural networks [**139**, **144**]. Almost all of the artificial neural networks used nowadays contain fully connected layers, which have a quadratic number of connections with respect to the number of neurons. This type of fully connected layers contain the most of the neural network connections. Because the weight corresponding to each connection has to be carefully optimized during the learning process, this leads to increased computational requirements, proportionally to the number of connections that need to be optimized. Inspired by the fact that biological neural networks are sparse, and even more, they usually have small-world and scale-free topologies, in these two chapters we show that a striking amount of the connections from the fully connected layer in artificial neural networks is actually redundant. Furthermore, we demonstrate that we can safely decrease the number of connections from a quadratic relation to a linear relation, with respect to the number of neurons, at no decrease in accuracy (many times, even with an increase in accuracy). It is worth highlighting that the connections reduction is done in the design phase of the neural network, i.e. before training. In Chapter 4 [**144**], we use a fixed scale-free connectivity pattern. Furthermore, in Chapter 5 [**139**], we take this idea further and, starting with a random sparse connectivity pattern and adding an evolutionary process during the training phase of the ANN model, we are capable to reach even better performance. Our results show that it is possible to replace the fully connected layers in artificial neural networks with quadratically faster

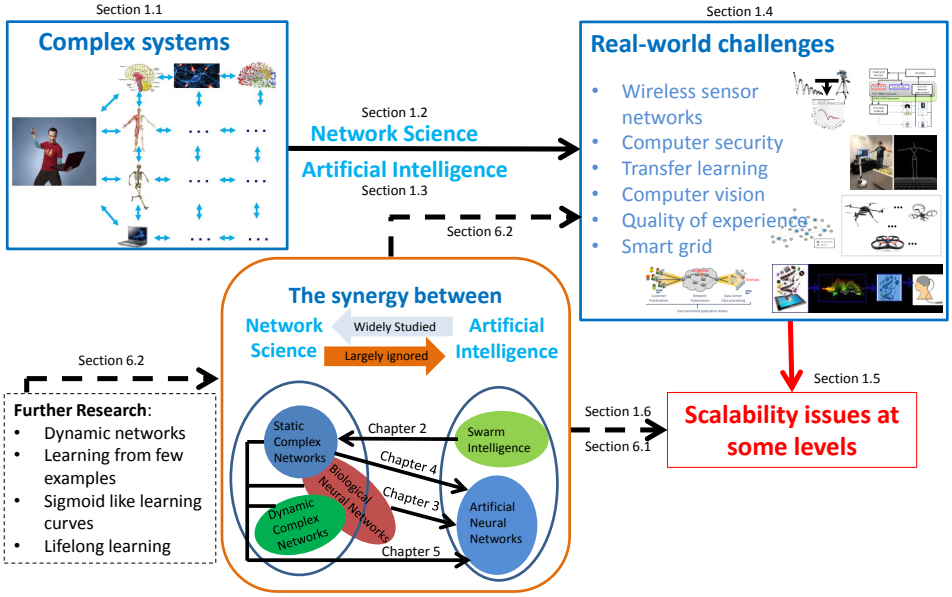


Figure 1.3 – Thesis storyline.

counterparts in both phases, training and exploitation, and lead to the possibility of building ANN models in excess of billions of neurons.

1.7. How to read this thesis

We tried to make as much as possible the chapters of this thesis self-contained. Thus, it is not necessary to read them in a strict succession, although this will lead to a more gradual introduction to the proposed concepts. An outlook to the thesis is depicted in Figure 1.3.

CHAPTER 2

Polylogarithmic centrality computations in complex networks

In this chapter we present the first core contribution of this thesis, showing how artificial intelligence can be used to improve network science algorithms. Specifically, we tackle the difficult problem of understanding and controlling complex networks. This is due to the very large number of elements in such networks, on the order of billions and higher, which makes it impossible to use conventional network analysis methods. Herein, we employ artificial intelligence (specifically swarm computing), to compute centrality metrics in a completely decentralized fashion. More exactly, we show that by overlaying a homogeneous artificial system (inspired by swarm intelligence) over a complex network (which is a heterogeneous system), and playing a game in the fused system, the changes in the homogeneous system will reflect perfectly the complex network properties. Our method, dubbed Game of Thieves (GOT), computes the importance of all network elements (both nodes and edges) in polylogarithmic time with respect to the total number of nodes. Contrary, the state-of-the-art methods need at least a quadratic time. Moreover, the excellent capabilities of our proposed approach, it terms of speed, accuracy, and functionality, open the path for better ways of understanding and controlling complex networks.

2.1. Introduction

In any real-world system, at micro and macro-scale, from the vigintillions of interacting atoms in the observable universe, to the billions of persons who live on Earth, there are amazing structures of networks of networks. These networks can be studied, understood, and controlled by the means of network science and complex networks [**192**], leading to advances in many domains, including neuroscience [**60**,**168**,**191**], astrophysics [**85**], biology [**90**,**224**] epidemiology [**97**], social networks [**63**,**87**], transportation networks [**46**], communication networks [**8**,**134**], and artificial intelligence [**144**] (to mention but a few). Yet, unveiling the complex networks hidden patterns and computing even their most basic properties is far from trivial, due to the massive number of node entangles that interact in non-obvious ways, evolving and unfolding continuously [**32**].

This chapter is integrally based on:
D.C. Mocanu, G. Exarchakos, A. Liotta: *Node Centrality Awareness via Swarming Effects*, Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2014, San Diego, USA.
D.C. Mocanu, G. Exarchakos, A. Liotta: *Decentralized dynamic understanding of hidden relations in complex networks*, 2017 (submitted for journal publication).

Among all these network properties, the centrality (or importance) of nodes and links is fundamental to understanding things such as: biological neural networks [**60**, **168**, **191**], cosmic structures [**85**], biological networks [**90**], how viruses spread or can be contained [**167**]; which people or news are influencing opinions and decisions the most [**12**]; how to protect computer systems from cyber-attacks [**217**]; or how to relay data packets in the one-trillion Internet-of-Things network of the future. While there is ample literature on node centrality computation [**108**], the existing methods do not scale to the size and dynamics of practical complex networks, which operate at the tunes of millions to trillions nodes. Besides that, the state-of-the-art centrality metrics are designed for specific goals, and one metric which performs well for one goal is suboptimal for another [**24**]. Furthermore, existing methods focus on finding the most important network elements (i.e. nodes or links), but fail to capture the hidden relations across the whole network links and nodes. The centralized algorithms consider the topology as a whole, overlooking many of the local features [**108**].

Per contra, the decentralized methods are usually based on local computations to construct statistics of network elements (as in [**219**]), but fail to capture the overall network structure. In fact, the most effective decentralized methods nowadays still fail to capture all the relations between the networks elements, and this is our main target. In addition, current methods have technological constraints that have to be surpassed. To tackle the scale as well as dynamics of real-world networks, we need to compute centrality metrics not only accurately but also timely, based on the existing computational capabilities.

To tackle all of the above constraints and limitations, in this chapter we propose a new viewpoint to model and understand complex networks. The basic idea is fairly simple. First, we overlay a homogeneous artificial system (a system created in such a way that all its elements ponder equally) over a complex network, which is a heterogeneous system - its level of heterogeneity being given by its topology. We then start a gaming process, whereby the artificial system entities start interacting with the network. What's interesting is the artificial system evolves in different ways, depending on the features of the complex network. In turn, network features, specifically the centrality metrics, start emerging. Our viewpoint is inspired to a basic principle of physics. If one would like to measure the volume of an irregular-shape object then one solution would be analytical, by measuring its dimensions and by solving some complicated triple integrals. An alternative much faster and ingenious solution, which needs just middle school knowledge, is the water displacement method coming from the Ancient Greeks, i.e. Archimedes of Syracuse. One would need just to submerge that irregular object in a graduated cylinder filled with water and to measure the water displacement. Further on, this easy to obtain volume can be used to measure other properties of the object, e.g. density.

Keeping the proportion, in the case of complex networks, the artificial homogeneous system represents the water, and the centrality represents the volume, while the game represents the action of submerging the irregular object. With the complex networks constraints in mind, our proposed homogeneous system follows four stratagems:

(1) completely decentralized computations, so that all nodes contribute simultaneously to the calculation of centrality;

(2) computational simplicity, so that the algorithm may be executed in thin nodes, such as the low-resources sensors of the Internet of Things;
(3) nature-inspired, swarm computations [**22**], to pursue global convergence through localized, stochastic actions;
(4) human-behaviour like computations [**179**](namely, egoistic behaviour), to gain an insight on the topological features of the network.

Altogether, the above four stratagems are confined in a novel algorithm, dubbed Game of Thieves (GOT).

The remaining of this chapter is organized as follows. Section 2.2 presents background knowledge on complex networks. Section 2.3 presents the intuition behind our proposed method and its mathematical formulation. Section 2.4 makes an analysis of GOT in terms of scalability and optimal parameters choice. Section 2.5 describes the experiments performed and analyzes the results. Finally, Section 2.7 concludes the chapter and presents directions of future research.

2.2. Background

In this section we briefly introduce some background information about complex networks, for the benefit of the non-specialist reader.

2.2.1. Complex networks

Complex networks [**157**] are graphs characterized by non-trivial features. Formally, any arbitrary network is an object which contains nodes (or vertices) and directed or undirected links (or edges) between nodes. Mainly, based on their properties, there are three classes of networks, as presented next.

2.2.1.1. *Erdös-Rényi random graphs.* In this type of networks, any node pair is connected with the same probability, $p \in [0, 1]$, by an edge [**57**]. By using this property, and creating the Erdös-Rényi Random Graphs dynamically, one may obtain a graph that has no particular structures. Due to the assumption that each edge is independent, it might be inappropriate to model real-world phenomena with Erdös-Rényi Random Graphs, and they are usually used for theoretical demonstrations of graph properties. Hence, the "Scale-Free" and "Small-World" models, discussed next, are more widely used in real networks modeling.

2.2.1.2. *Scale-Free networks.* In these networks the degree distribution follows a power law [**19**]. For instance, in addition to World Wide Web, electric power grids, transportation networks, many other networks have been found to be scale-free topologies [**157**]. An algorithm to build scale-free graphs (based on preferential attachment) has been proposed in [**19**]. In short, this means that the nodes with high degree, commonly referred to as "hubs" in the literature, are favored to obtain new connections when a new node is added to the graph.

2.2.1.3. *Small-World networks.* These are graphs in which each node can be reached from any other node in a small number of steps [**218**]. Typically, the shortest path between any two nodes has a length of $\propto log(n)$.

2.2.2. Centrality in complex networks

Centrality is a measure to assess how important individual nodes (or links) are in a network and how they can affect their neighborhood or even the whole network. However, there is no clear way to define "centrality" in graphs. In the literature, there are several methods to calculate node's centrality, each one focused on specific features. Broadly, there are two main approaches: centralized and decentralized methods. We exemplify these approaches, through four state-of-the-art centrality metrics, as summarized in Table 2.1.

2.2.2.1. *Betweenness Centrality (BC).* BC and its variants are among the most utilized metrics to assess the nodes' importance [**30**]. It quantifies how a node lies on the path between other nodes. Formally, for a node $n \in \mathbf{V}$, where V is the set of all nodes, this can be written as:

$$C_{be}(n) = \sum_{w,u \in \mathbf{V}} \frac{\sigma_{w,u}(n)}{\sigma_{w,u}} \tag{2.1}$$

where $\sigma_{w,u}(n)$ represents the number of shortest paths from node w to node u which pass through the node n, and $\sigma_{w,u}$ represents the total amount of shortest paths from w to u. The computational complexity of the original algorithm is $\mathcal{O}(n^3)$, making it unsuitable for large networks. For this reason, in the last period, several BC approximations have been proposed (see [**29**] and references therein).

2.2.2.2. *Current Flow Betweenness Centrality (CFBC).* It was proposed in [**159**], and is inspired to how the electric current flows into an electric network. In comparison to BC, CFBC does not make the assumption that only the shortest paths are important to compute the node centralities. It considers all the possible paths in a network, by making use of random walks. In general, CFBC is considered to reflect centrality more accurately than BC, but it is slower.

2.2.2.3. *Second Order Centrality (SOC).* It is a novel form of node's centrality metric, calculated in a decentralized way, and proposed by Kermarrec et al. in [**96**]. The algorithm is based on a random walk in the graph, which starts from a random chosen node, and runs continuously. After the random walk has visited all nodes at least three times, the standard deviation of the number of steps required to reach each of the nodes is computed. The authors demonstrate why this value reflects the centrality of nodes.

2.2.2.4. *DACCER.* It is a decentralized algorithm to measure the centrality of nodes in networks, proposed by Wehmuth and Ziviani in [**219**]. The main idea is that each node is computing its own centrality, based on the information acquired from its vicinity. The authors showed that a two-hop vicinity reflects well the closeness centrality.

2.3. Game of Thieves (GOT)

2.3.1. Intuition

Intuitively, GOT mimics the egoistic behaviour of a multitude of thieves faced with the prospect of easy-to-steal diamonds - from here comes its name. Our homogeneous artificial system has two virtual elements: a group of wandering thieves (in game theory: the actors) and a set of virtual diamonds or vdiamonds (in game theory: the resources). At start, each node is artificially endowed with vdiamonds

which are nomadic, reusable and non-replicable virtual resources, generalizing and virtualizing the concept from [**59**, **134**]. Likewise, each node is endowed with wandering thieves, mobile actors which act stochastic (they wander in search of vdiamonds to steal) and egoistic (as soon as they have an opportunity, they steal vdiamonds and take them back to their home node).

A thief has two states: "empty" (i.e. it does not carry any vdiamond) and "loaded" (i.e. it carries one vdiamond). Besides that, he has three functionalities: he wanders from one node to a neighbour, picked randomly (chaotic behaviour), to search for vdiamonds; when he finds vdiamonds, the thief fetches one (egoistic behaviour); he brings it to his home node by following back the same path previously used to find the vdiamond. Like any other vdiamond, this newly homed vdiamond becomes immediately available for the other wandering thieves to steal it. When GOT starts, all nodes host the same given number of thieves and vdiamonds. Then the game proceeds in epochs. At each epoch, all thieves hop from their current location to the next one, changing state when they find or deposit a new vdiamond.

Comparing with classical swarm computational methods, in GOT the thieves do not communicate directly among them - they are independent actors in the game. Nodes, links and thieves perform just local actions, while the interactions at global level are ensured by the vdiamonds migration. In turn, the vdiamonds migration is driven by the network topology (a heterogeneous system), since the resources tend to be drawn more rapidly from the better connected nodes and tend to be accumulated in the less connected nodes. It is through this migration process that the network elements strengths (node and link centralities) gradually emerge from the vdiamonds distribution.

2.3.2. Formalism

Let us consider $G = (V, E)$ to be an undirected graph (G) containing a set of nodes (V) and a set of edges (E). Φ_0^n is the initial amount of vdiamonds in node $n \in V$ (at time zero). Similarly, Φ_T^n denotes the number of vdiamonds in node $n \in V$ at time T(i.e. after the game has run for T epochs). Φ_T^l is the number of "loaded" thieves traversing link $l \in E$ at epoch T. The average number of vdiamonds present at a node (n), after the game has run for a duration of T epochs, can be computed as:

$$\bar{\Phi}_T^n = \frac{1}{T} \sum_{e=0}^{T} \Phi_e^n \tag{2.2}$$

The average number of "loaded" thieves passing through link (l) after T epochs will be:

$$\bar{\Psi}_T^l = \frac{1}{T} \sum_{e=0}^{T} \Psi_e^l \tag{2.3}$$

Counterintuitively, a smaller $\bar{\Phi}_T^n$ value reflects a more important node, while a higher $\bar{\Phi}_T^n$ value indicates a less important one, as the more central nodes have higher chances to be visited by thieves and they will be depleted first. Intuitively, higher $\bar{\Psi}_T^l$ values reflect more important links, while lower $\bar{\Psi}_T^l$ values point to the less important links.

2.3.3. Thieves behaviour

A key ingredient in the success of GOT is the behavior of thieves (the agents) within the network. Before going into details, let us add the following notations: Γ^n is the set of nodes that are connected by a link with node n, $\forall n \in V$; with $\Omega^{nm} \geq 0$ the weight of the link which connects the nodes $n \in V$ and $m \in V$; and with Υ_a a dynamic list with the nodes visited by thief a, useful to keep the path of a in his search for vdiamonds.

So, a thief a in the "empty" state will always perform successively the following operations in any epoch e:

- It randomly picks a node $m \in \Gamma^n$, where n is its actual location, with the following probability $p_a^{nm} = \frac{\Omega^{nm}}{\sum_{v \in \Gamma^n} \Omega^{nv}}$; and it moves to node m. It is clear that unweighted networks are just a particular case of weighted networks, by setting the weights of all links from the networks to 1.
- If $m \in \Upsilon_a$ then all the nodes situated after m in the list are removed from Υ_a, to avoid the apparition of cycles in the list.
- If $m \notin \Upsilon_a$ then m is added to the end of Υ_a.
- If node m has vdiamonds then the thief a takes one and it changes his state to "loaded", while node m decreases Φ_e^m by one vdiamond.

At the same time, a thief a in the "loaded" state will always perform successively the following operations in any epoch e:

- It moves from the last node n from Υ_a, which is his actual location, to the last but one node m from Υ_a, and after that it removes n from Υ_a.
- Link l from n to m increases Ψ_e^l by one.
- If m is the home node of a, the thief unloads the vdiamond, and sets his state to "empty", while node m increases Φ_e^m by one vdiamond.

2.3.4. Algorithm and functionality illustration

GOT is presented in Appendix B, Algorithm B.1, while Figure 2.1 shows snapshots of GOT in operation at eight different times, on a simplistic 10-node network. Notably, after just 5 exchanges the $\bar{\Phi}_T^n$ values already reflect the nodes centrality. Being a purely stochastic process, GOT rapidly leads to well-organized patterns in the resource distribution, as visible from the evolution of the colour codes over the eight epochs. This behaviour agrees with diffusion-limited aggregation processes [**223**] and ensures that the most connected (central) nodes lose their resources first, while the least connected nodes (e.g. leaves) will tend to accumulate resources more rapidly (Figures 2.1g and 2.1h). This also follows the intuition that nodes with higher centrality have higher chances of being visited by thieves. This observation is also compatible with a similar phenomenon discovered by Saavedra et al. in the context of real-world biological and economical networks, whereby the strongest network contributors were found to be the most prone to extinction [**175**].

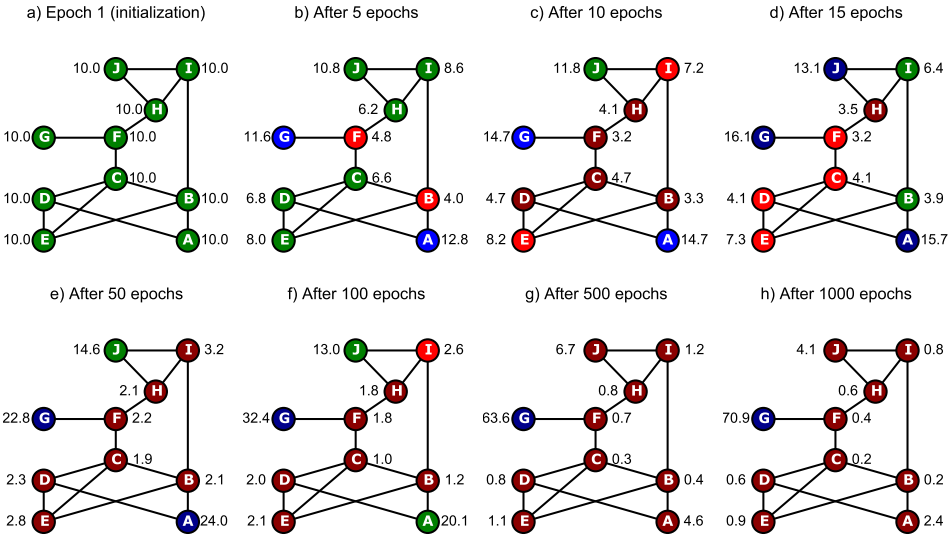


Figure 2.1 – Thieves in action. Snapshots with the illustration of GOT behavior over epochs on a simple unweighted network with 10 nodes. Initially, we set $\Phi_0^n = 10$ vdiamonds and three thieves per node, and we let the game to run for 1000 epochs. For each node n the colors symbolize: $0 \leq \bar{\Phi}_T^n \leq 2$ (dark red); $3 \leq \bar{\Phi}_T^n \leq 6$ (red); $7 \leq \bar{\Phi}_T^n \leq 13$ (green); $14 \leq \bar{\Phi}_T^n \leq 18$ (blue); $18 \leq \bar{\Phi}_T^n \leq 100$ (dark blue), while he numbers on the side of each node show $\bar{\Phi}_T^n$, where T=1, 5, 10, 15, 50, 100, 500, and 1000 epochs in subplots a, b, c, d, e, f, g, and h respectively.

2.3.5. Stopping criterion

The algorithm stopping criterion of GOT is reached when just a small number of nodes still changes their rank of importance from one epoch to the next successive ones using the scores assigned by the GOT algorithm. Formally, let us define a vector Λ_e for any epoch e. Each element $\Lambda_{e,n} \in \Lambda_e$ is the rank of importance given by GOT to node $n \in V$. Note that all elements of Λ_e are unique natural numbers between 1 and $|V|$. Thus, a general stopping criterion for GOT can be expressed as:

$$\frac{1}{H}\Big(\sum_{e=T-H}^{T} \sqrt{\sum_{n=1}^{|V|}(\Lambda_{e,n} - \Lambda_{e-1,n})^2} \Big) < \epsilon |V| \tag{2.4}$$

where T is the actual epoch, H is the number of past epochs taken into consideration, and $\epsilon \in (0, 1)$ is a subunitary real number. In practice, we found that satisfactory results are achieved by setting $H = 10$, and $\epsilon = 0.02$.

In other words, the aforementioned settings mean that a maximum 2% of the nodes change their rank over 10 consecutive epochs (SC_2), and we validate this stopping criterion throughout the chapter. Furthermore, Figure 2.2 reflects GOT stopping criterion for networks with 100 nodes and $H = 10$ over 1000 epochs.

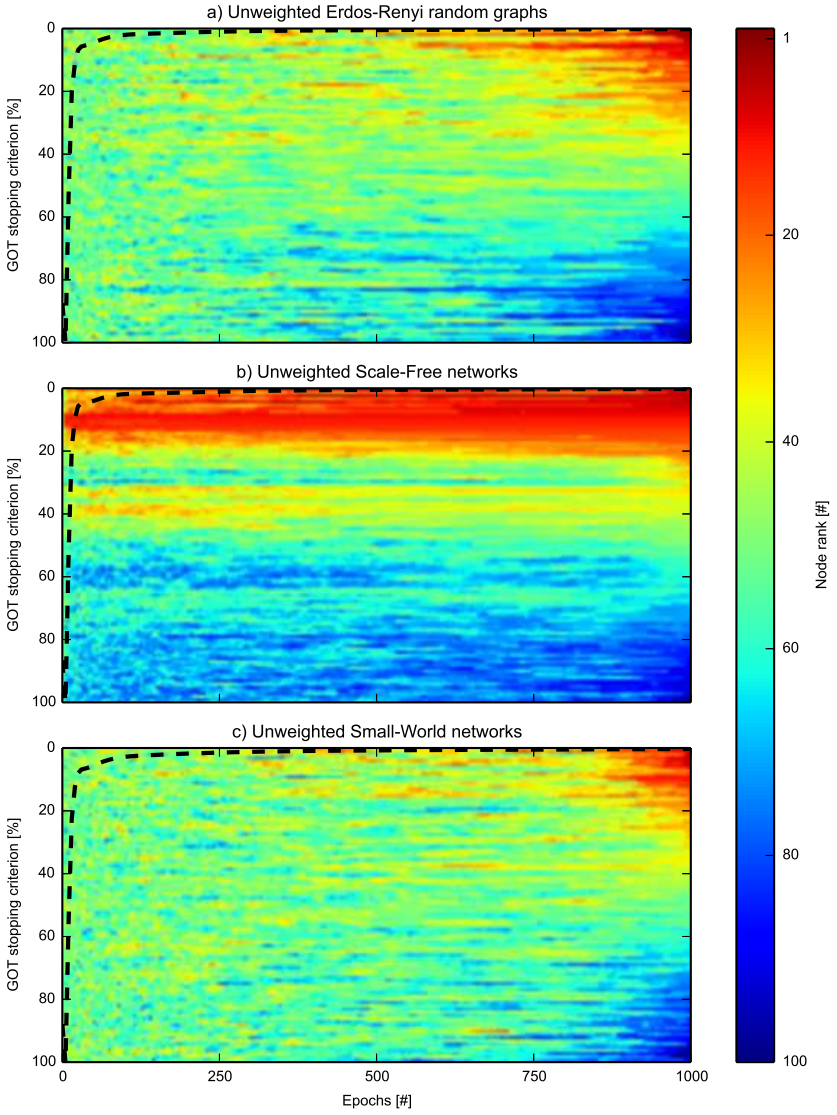


Figure 2.2 – GOT visualization. Nodes rank assigned by GOT in each epoch, while it runs for 1000 epochs in random generated networks with 100 nodes and between 500 and 1000 links. The results are averaged on 10 different networks for each network type. The dash lines show GOT stopping criteria as a percentage of the total number of nodes, at any epoch T.

2.4. GOT analysis

2.4.1. Visualization

To begin with, we have tested GOT in small scale simulations, mainly to visualise its operation. We simulated ten Erdős-Rényi Random Graphs [**57**], ten

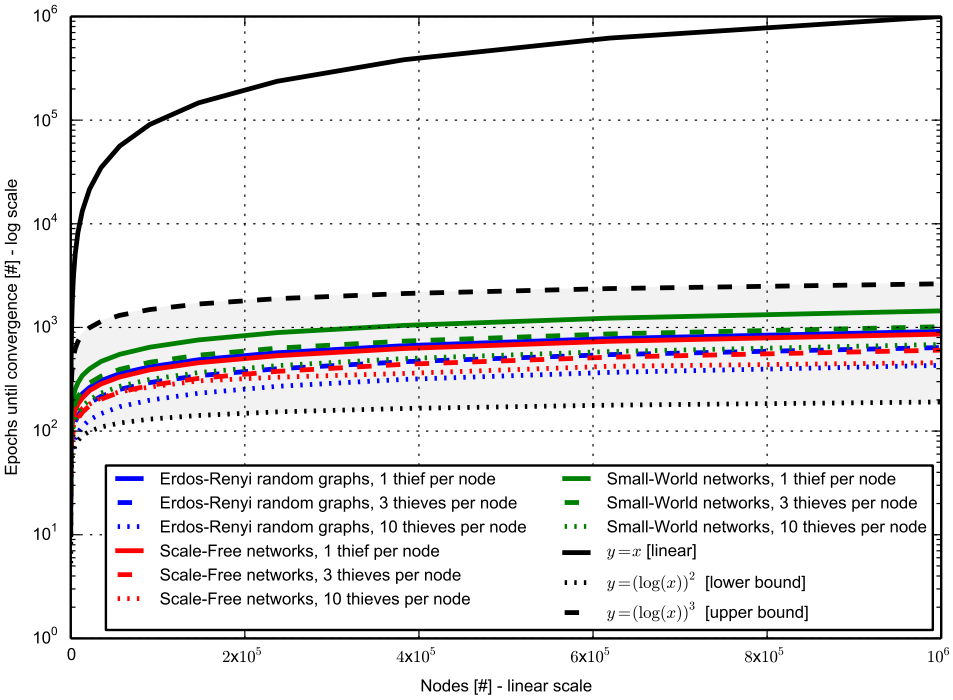


Figure 2.3 – GOT scalability. The plot shows the number of epochs needed by GOT to converge. For each network used, the number of edges is between 5 and 10 times bigger than the number of nodes. Independently of the network model, or the number of agents used per node (i.e. 1, 3, or 10), GOT convergences in a number of epochs empirically lower-bounded by $log^2|V|$ and upper-bounded by $log^3|V|$, which is on the polylogarithmic scale with respect to the total number of nodes in the network, $|V|$.

Scale-Free networks [**19**], and ten Small-World networks [**218**], each being unweighted, and including 100 nodes and 500 to 1,000 links. The game started with 1 thief and 100 vdiamonds per node and run for 1,000 epochs. At that point we averaged the results on each network type. Figure 2.2 shows both the node ranking (following a colour scheme) and GOT's convergence level (dotted line). Remarkably, after just a few hundred epochs GOT stabilizes, indicating that the striking majority of node ranks have been established. It is interesting to see that scale-free networks stabilize significantly faster (in just a few epochs), as it was expectable by the peculiar node degree distribution on such network types.

2.4.2. Scalability

To study the ability of GOT to scale, we have conducted extensive simulations on a variety of networks, up to one million nodes. We consider three types of randomly generated networks, Erdős-Rényi Random Graphs, Scale-Free and Small-World networks, both weighted and unweighted. Simulations are randomized, repeated and averaged to ensure statistical significance. We look at the

Table 2.1 – Comparison of five centrality algorithms using different performance criteria (i.e. functional, computational efficiency, and accuracy). The bold values represent the best performer for specific performance criteria.

Algorithm	Functional Performance	Computational Efficiency Performance		Accuracy Performance
		Architecture	Time complexity	
GOT with SC_2	**Nodes and Links**	**Fully Distributed**	$\mathbf{O(\log^2 \mathbf{|V|}) < O(GOT) < O(\log^3 \mathbf{|V|})}$	**83.4 %**
CFBC [**159**]	Nodes or Links	Centralized	$O(I(\|V\|-1)+\|V\|\|E\|log\|V\|)$	8.3 %
BC [**29**]	Nodes or Links	Centralized	$O(\|V\|\|E\|)$	8.3 %
SOC [**96**]	Nodes	Partially Distributed	$O(\|V\|^2) < O(SOC) < O(\|V\|^3)$	0 %
DACCER [**219**]	Nodes	**Fully Distributed**	n/a	0 %

number of epochs required for GOT to converge, using the stopping criterion described in Section 2.3.5, dubbed SC_2 (i.e. maximum 2% of the nodes change their rank from one epoch to another in the last 10 epochs). Therein we shall also discuss why SC_2 is satisfactory for the assessment of node and link centrality. We simulate networks ranging from 10 to 10^6 nodes, having a number of links comprised between six and ten times the number of nodes. We also tried different starting conditions, with 1, 3 and 10 thieves per node, setting $\Phi_0^n = |V|$.

Empirically, we found that the number of epochs needed for convergence is on the polylogarithmic scale of the network size. Figure 2.3 depicts this sub-linearly relation for each network type. More exactly, the parallel time complexity of GOT convergence, $O(GOT)$, is bounded by $log^2|V| < O(GOT) < log^3|V|$. To our best knowledge, this represents a breakthrough compared to the state-of-the-art centrality algorithms which have at least a quadratic time complexity (see Table 2.1).

2.4.3. Optimal parameter choice

In total, GOT has three parameters: i.e. the number of epochs to run the game, the initial amount of vdiamonds which have to be set in each node, and the number of thieves in each node. In terms of accuracy, these parameters do not affect the algorithm performance, if the game is ran until the SC_2 criteria is fulfilled. Thus, we studied them just in term of computational efficiency and how they affect the SC_2 criteria. Previously, we demonstrated that independently of the network size GOT converges to SC_2 in a bounded number of epochs. So, we consider a safe practice to set the number of epochs to run the game to the lower bound of SC_2, $O(log^2|V|)$, if one needs the results faster, or to the upper bound, $O(log^3|V|)$, if a better accuracy is needed. To find the best value for the initial amount of vdiamonds per node, we performed extra experiments on different network types and sizes. We found that this parameter does not significantly affect the convergence time of the algorithm if it is set to non trivial values, e.g. 1, 2, 3 vdiamonds per node. Our experiments showed us that best practice is to set this parameter to the total number of nodes in the network. We should, in fact, mention that the initial value of vdiamonds is not the crucial one, since it has negligible computational costs. Finally, we have analyzed how the number of thieves per node influences the number of epochs needed by the algorithm to converge considering different network types and sizes. In all cases, independently on the number of thieves, the game converged within the bounds of SC_2. To conclude, we consider that by setting just one thief per node is enough, due to the fact that it achieves

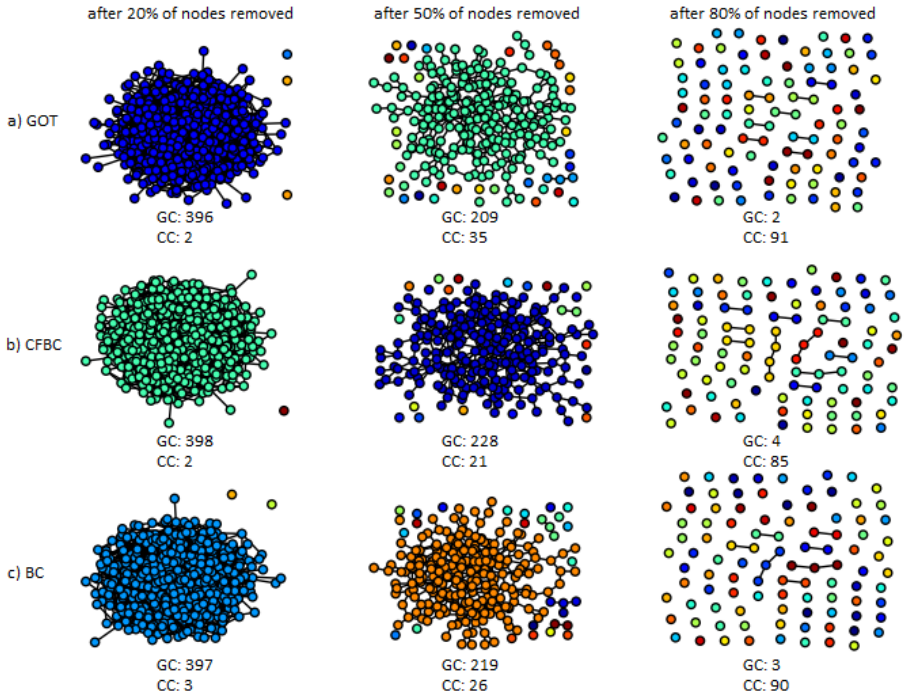


Figure 2.4 – NRP procedure - nodes. Snapshots during the NRP procedure for nodes in a random generated network with 500 nodes. At the bottom of each subplot, the number of connected components (CC) and the size of the giant component (GC) are shown.

fast convergence time, independently of the cases studied, while being the fastest option in terms of the total number of messages exchanged in the network.

2.5. Experiments and results

2.5.1. Evaluation method

We have assessed GOT both on simulated and real-world networks, against state-of-the-art centrality metrics, i.e. Betweenness Centrality (BC) [**29**], Current Flow Betweenness Centrality [**159**] (CFBC), DACCER [**219**], and Second Order Centrality [**96**] (SOC), as detailed further.

2.5.1.1. *Evaluation metric.* In the experiments, we have used a standard procedure to assess the accuracy of the nodes centrality metrics, namely the Node Removal Procedure (NRP) [**96**], as described next. After a centrality metric assigns scores for each node of the graph, all the nodes are sorted according to their scores, starting with the most important one, and ending with the less important one. Furthermore, the nodes from this sorted list are removed one by one from the graph, and after each removal the size of the Giant Component (GC) and the number of Connected Components (CC) in the remaining graph are measured. A node centrality metric is considered to be better if the number of connected components is as big as possible, while the size of the giant component is as small

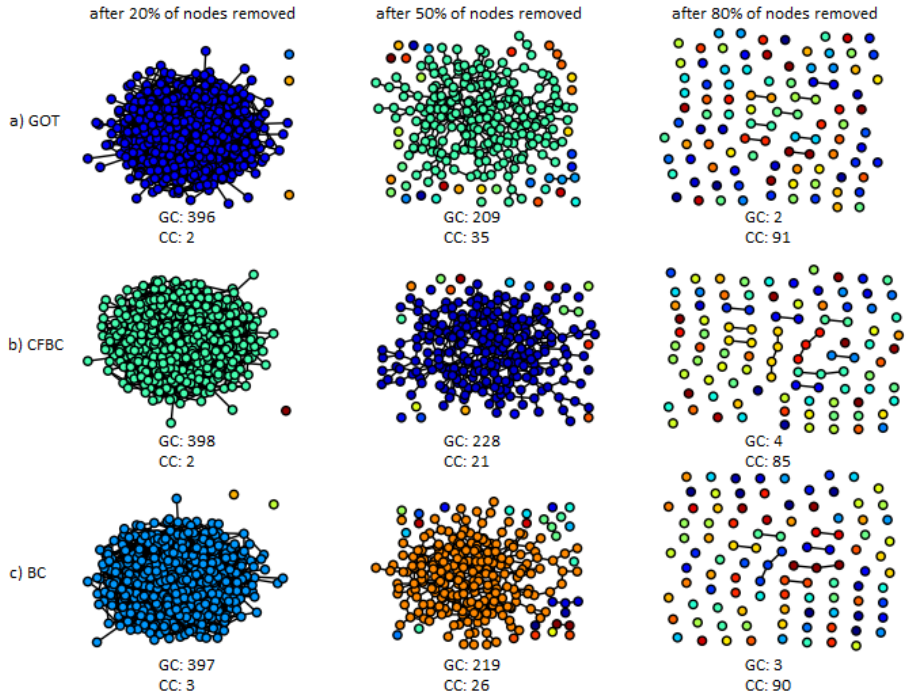


Figure 2.5 – NRP procedure - links. Snapshots during the NRP procedure for links in a random generated network with 100 nodes. At the bottom of each subplot, the number of connected components (CC) and the size of the giant component (GC) are shown.

as possible, during this NRP procedure. Similarly, NRP can be applied for links, if the links are sorted according with their importance and after that they are removed one by one. In Figure 2.4, we have illustrated some snapshots during the NRP procedure for nodes in a random network with 500 vertices. In Figure 2.5 we have illustrated the NRP procedure for links in a random network with 100 vertices.

2.5.1.2. *Implementation.* For all the experiments performed in this chapter we used Python and the NetworkX library [**75**]. Furthermore, for BC and CFBC we used the standard implementations offered by the aforementioned library, while GOT, DACCER and SOC were fully implemented by us. Moreover, we used NetworkX to generate the simulated networks, to work with the real-world networks under scrutiny, and to compute the size of the giant component and the number of connected components during the NRP procedure. The hardware platform utilized was a typical desktop computer (i.e. Intel Core i7, 32 GB RAM).

2.5.2. Performance on simulated networks

To assess GOT's accuracy in identifying the correct node centrality (while validating SC_2), we used three classes of simulated networks: Erdős-Rényi Random

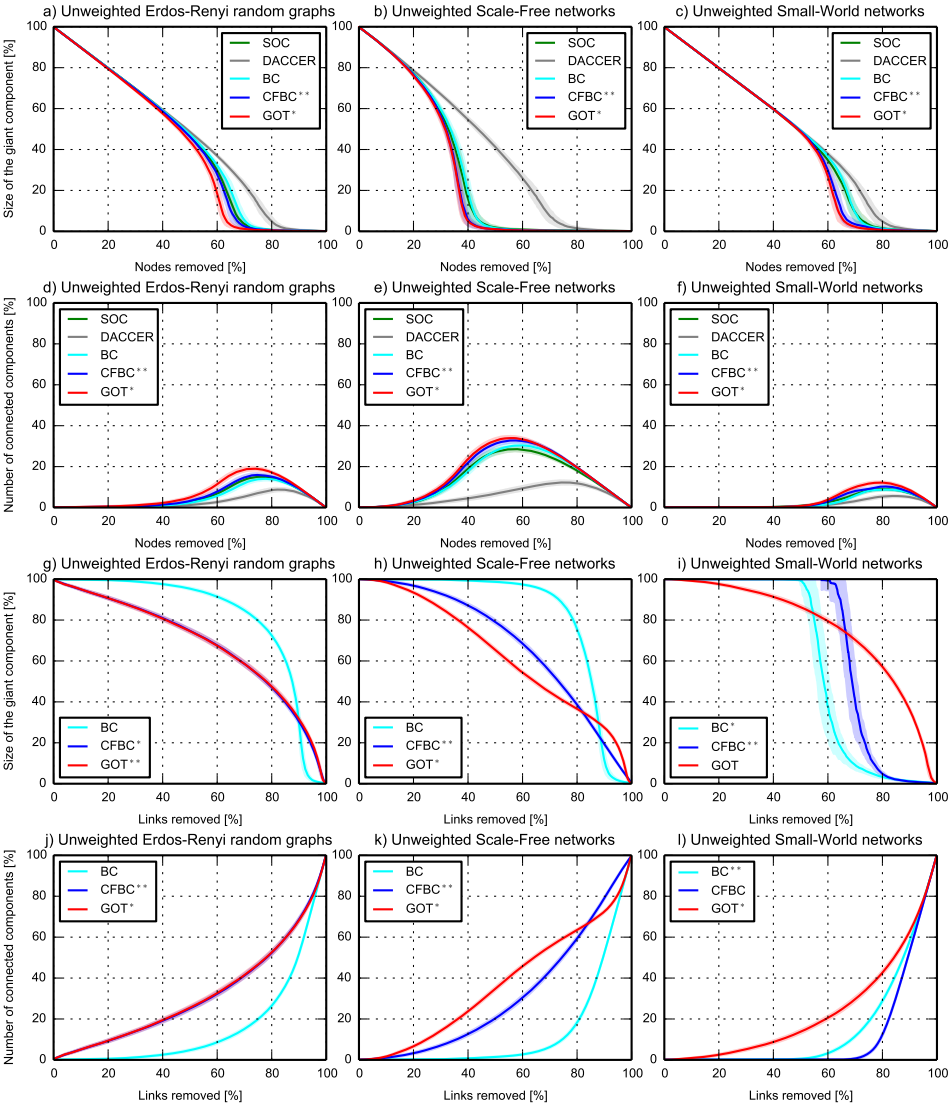


Figure 2.6 – GOT accuracy - random generated unweighted networks. The evolution of the size of the giant component and of the number of connected components with mean (the straight line) and standard deviation (the shadow area) in unweighted networks during the NRP procedure, averaged over 100 networks in each subplot. The y-axes give figure of merit, while the x-axes represent percentage of node and links removals, respectively. In the top subplots, nodes centrality is assessed, while in the bottom subplots, the links centrality is evaluated.

Graphs, Scale-Free and Small-World networks. For each class, we randomly generated 100 weighted networks with weights generated randomly between 1 and 10, and 100 unweighted networks. Each network had 1,000 nodes and between 4,500 and 5,500 links. Comparing GOT to the literature was tricky, because nobody so far has managed to compute node and link centrality rankings simultaneously,

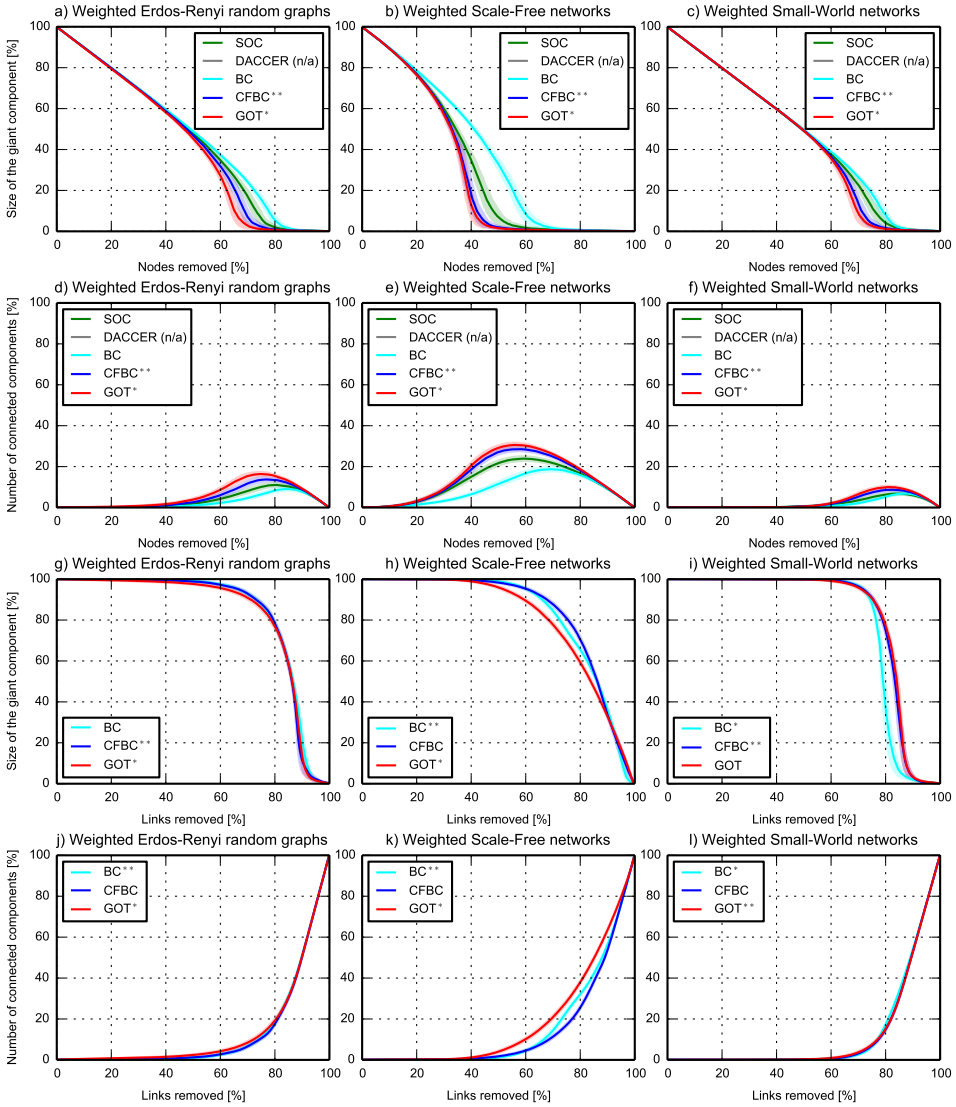


Figure 2.7 – GOT accuracy - random generated weighted networks. The evolution of the size of the giant component and of the number of connected components with mean (the straight line) and standard deviation (the shadow area) in weighted networks during the NRP procedure, averaged over 100 networks in each subplot. The y-axes give figure of merit, while the x-axes represent percentage of node and links removals, respectively. In the top subplots, nodes centrality is assessed, while in the bottom subplots, the links centrality is evaluated.

as we do. We compared to two centralized methods, Brandes' algorithm [**29**] for Betweenness centrality and Current flow betweenness centrality [**159**], which have variants for vertices and edges. We ran these multiple times to allow the comparison with GOT. Also, we compared GOT with two decentralized algorithms, DACCER [**219**] and Second order centrality [**96**], for nodes centrality. DACCER

and SOC do not have variants for links centrality, and DACCER is not capable to assess nodes centrality in weighted networks. For GOT, we set 1 thief and $\Phi_0^n = 1000$ vdiamonds per node and we ran the algorithm until SC_2 convergence was achieved. To assess the accuracy of all metrics used, we used the NRP procedure [**96**]. Figure 2.6 and 2.7 depict the generality of GOT, which has a better accuracy than all the other centrality metrics for nodes, while for links it outperforms its competitors in 8 of 12 scenarios, staying very close to the best performer (BC or CFBC) in the remaining 4 scenarios. But we should note that BC and CFBC are only used to compare centrality accuracies - these are centralized algorithms and would not scale in massive-scale networks (which is the ultimate goal of GOT). In all scenarios, SC_2 was fulfilled on average after 274 ± 45 epochs, this being within the previous discussed bounds. More than that, in both figures, it can be observed that GOT performs better because it is capable to discover well the centrality of the medium important nodes and links, while the other algorithms fail to do that.

2.5.3. Performance on real-world networks

We have validated GOT using three real-world networks (from different domains): the "Dolphins social network", an undirected social network of the most frequent associations between a community of 62 dolphins living in Doubtful Sound, New Zealand [**125**]; the "Internet", a symmetrized snapshot of the structure of the Internet created by Mark Newman from BGP tables posted by the University of Oregon in 2006 ; and the "High Energy theory collaborations, a weighted disconnected network with the co-authorships between scientists posting preprints on the High-Energy Theory E-Print Archive between 1 January 1995 and 31 December 1999 [**156**]. For GOT, we set 1 thief and $\Phi_0^n = |V|$ vdiamonds per node and we ran it for $log^2|V|$ epochs (i.e. the lower bound of GOT with SC_2) to avoid the overhead introduced by the SC_2 computing. By using the same NRP procedure as before, Figure 2.8 shows that GOT achieves a better accuracy than the other approaches in 10 out of 12 situations, while in the other 2 cases it stays very close to the best performer (CFBC) - again, CFBC is used only for comparison, being a centralized algorithm which would not be usable in massive-scale networks.

We emphasize that in the case of the "Internet" network, which was the biggest real-world network used in this chapter (i.e. 22,963 nodes, 48,436 links) a Python sequential implementation of GOT ran in 88 seconds and assessed both, nodes and links centrality, at the same time, while the cumulative times for the next two performers, BC and CFBC using their NetworkX [**75**] implementations, were 6,322 seconds and 66,977 seconds, respectively. These running times are at least two orders of magnitude larger than GOT. DACCER and SOC, using our own Python implementation, were a bit faster than BC and CFBC, and ran in 574 and 3,213 seconds, respectively, but their accuracy was much lower. Besides that, they were able to compute just nodes centrality.

The "High Energy" network was particularly interesting to show another singular feature of GOT: its ability to compute centrality in disconnected networks. This is not possible with existing distributed methods, so we use the centralized algorithm BC for the sake of performance comparison.

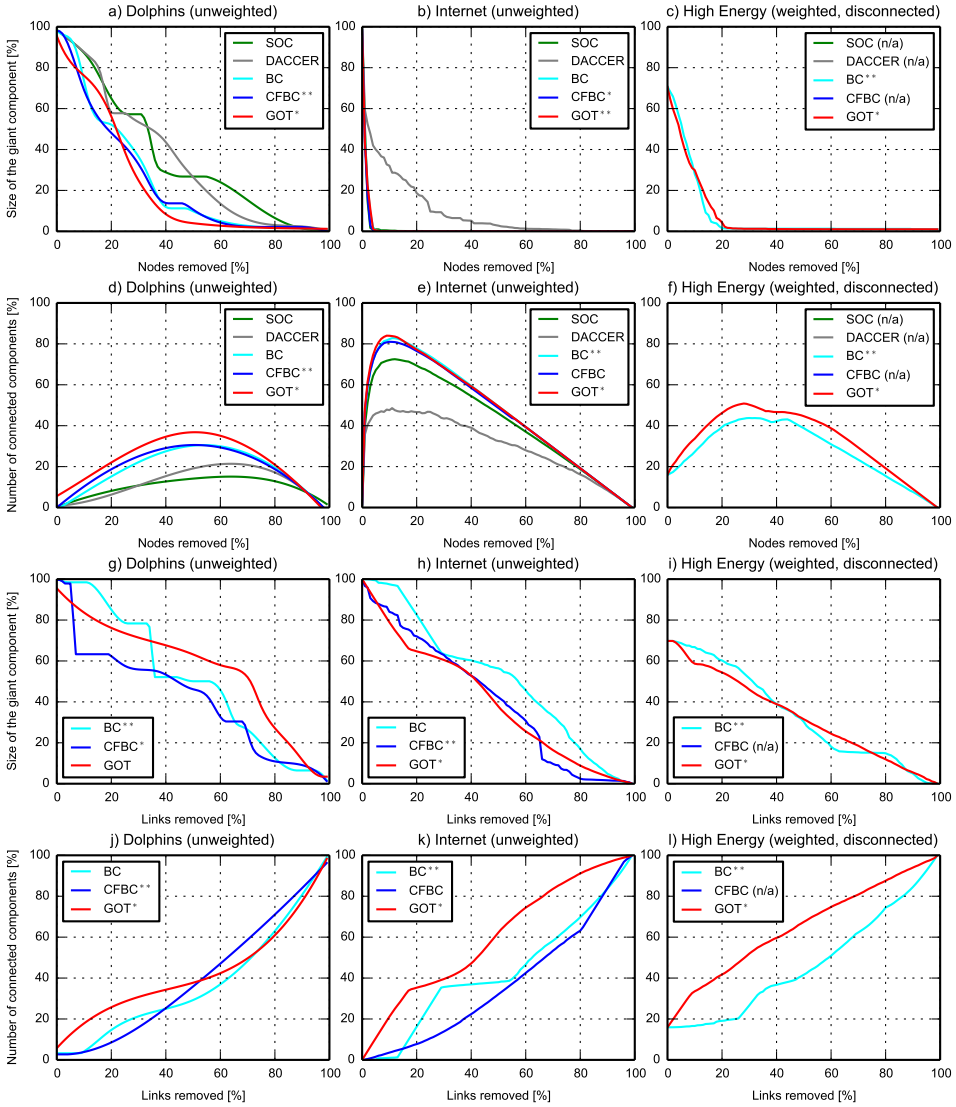


Figure 2.8 – GOT accuracy - real-world networks. The evolution of the size of the giant component and of the number of connected components during the NRP procedure in three real-world networks: Dolphins (62 nodes, 159 links, unweighted), Internet (22963 nodes, 48436 links, unweighted), and High Energy (8361 nodes, 15751 links, weighted, disconnected). The y-axes give figure of merit, while the x-axes represent percentage of node and links removals, respectively. The top subplots depict the performance of nodes centrality metrics. The bottom subplots show the links centrality metrics.

As a curiosity, looking at the High Energy network we found that prof. Jan Ambjorn was the most important researcher. Considering that this database was 16 years old, we found a strong correlation of GOT results with a recent Google scholar profile of prof. Jan Ambjorn (i.e. 15,664 citations, 67 h-index) on 18th

December 2016. We can then speculate that centrality algorithms may even be used to make future extrapolations on networks.

2.6. Discussion

GOT is a new approach to profiling complex networks using a fully decentralized method. It outperforms state-of-the art algorithms on three different performance criteria (i.e. functional, accuracy, and computational efficiency), as summarized in Table 2.1. Functionally, it is capable of assessing at the same time nodes and links importance in weighted, unweighted or disconnected networks. More than that, it outperforms state-of-the art algorithms in terms of accuracy, being capable to accurately capture the underlying relations between the network elements and to detect well all shades of centrality, including the most difficult entities - i.e. the one of medium importance. All of these are detailed in Table 2.2, which summarizes all the accuracy experiments by computing the area under the curve for each metric from each subplot of Figures 2.6, 2.7, and 2.8. Overall, GOT was the best performer in terms of accuracy in 30 out of 36 scenarios, while in the remaining 6 it was the second best performer or very close to the best performers - but these are centralized, thus unscalable methods.

Besides that, in terms of computational complexity, GOT is much faster and scalable (in terms of both number of nodes and number of links) compared to existing methods. The worst-case implementation of GOT is sequential (i.e. it emulates all network actions in sequence in a single computer). Yet this is bounded up by $O(|V|log^3|V|)$, which is much faster than the next three followers in terms of accuracy BC, CFBC, and SOC. These have computational complexity of $O(|V||E|)$ [**29**] , $O(I|V|-1)+|V||E|log|V|$ (where $O(I|V|-1)$ is the time necessary to compute the inverse Laplacian) [**31**], and at least $O(|V|^2)$ [**96**], respectively.

Even more strikingly, when GOT is implemented in distributed systems, its execution will proceed in parallel across all nodes. This natively decentralized version of GOT has a parallel time complexity on the polylogarithmic scale with respect to the number of nodes in a network. This makes it suitable to perform real-time analysis of very large-scale networks with billions of nodes, easily identifiable in the big data era, such as Facebook (in the range of 1.280.000.000 nodes) or the Internet of Things (expected to expand to an order of 1 trillion of nodes within the next few years).

To give an impression of the significance of the computational capability at hand, let us consider what GOT could achieve in a 1 trillion Internet of Things network of the near future. Assuming that each device would run GOT and would be able to transmit one message per millisecond. The scalability figures given above, would lead to a complete computation of all node and link ranks in a timespan comprised between 0.8 seconds (given by the lower bound of GOT with SC_2) and up to 22 seconds (given by the upper bound of GOT with SC_2). By comparison, if we were to use the state-of-the-art parallel processing algorithms of today on powerful computers, it would take at least several weeks of continuous computation to achieve comparable results. This places GOT in a much better position in terms of performing real-time centrality computations on massive-scale networks, being able to tackle not only scale but also network dynamics.

Table 2.2 – Experiments summary. Area under the curve (AUC), rounded to the nearest integer, computed for each metric from each subplot from Figures 2.6, 2.7, and 2.8. The bold values represent the best performer for that specific scenario, while n/a means that the metric is not suitable for that specific scenario.

				SOC	DACCER	BC	CFBC	GOT
Random generated unweighted networks (Figure 2.6)	Erdos	Nodes centrality	Giant size	4293	4625	4348	4237	**4086**
			Components number	486	267	453	530	**668**
		Links centrality	Giant size	n/a	n/a	8185	**6843**	6853
			Components number	n/a	n/a	1574	3129	**3141**
	Scale-free	Nodes centrality	Giant size	2960	4167	2987	2823	**2794**
			Components number	1386	571	1433	1569	**1620**
		Links centrality	Giant size	n/a	n/a	8308	6888	**6362**
			Components number	n/a	n/a	1277	3042	**3633**
	Small-World	Nodes centrality	Giant size	4447	4655	4472	4312	**4257**
			Components number	272	167	252	306	**365**
		Links centrality	Giant size	n/a	n/a	**6038**	7001	7645
			Components number	n/a	n/a	1484	1127	**2322**
Random generated weighted networks (Figure 2.7)	Erdos	Nodes centrality	Giant size	4501	n/a	4659	4360	**4219**
			Components number	354	n/a	258	453	**566**
		Links centrality	Giant size	n/a	n/a	8413	8358	**8287**
			Components number	n/a	n/a	1262	1257	**1340**
	Scale-free	Nodes centrality	Giant size	3204	n/a	3803	2974	**2917**
			Components number	1199	n/a	841	1392	**1479**
		Links centrality	Giant size	n/a	n/a	8301	8380	**8073**
			Components number	n/a	n/a	1547	1444	**1837**
	Small-World	Nodes centrality	Giant size	4636	n/a	4732	4511	**4456**
			Components number	191	n/a	162	244	**280**
		Links centrality	Giant size	n/a	n/a	**7907**	8231	8263
			Components number	n/a	n/a	**1208**	1177	1187
Real-World networks (Figure 2.8)	Dolphins	Nodes centrality	Giant size	3643	3527	2490	2400	**2272**
			Components number	1030	1228	1875	1971	**2344**
		Links centrality	Giant size	n/a	n/a	5094	**4172**	5625
			Components number	n/a	n/a	3691	3946	**4244**
	Internet	Nodes centrality	Giant size	179	1020	180	**163**	179
			Components number	4217	3034	4631	4577	**4641**
		Links centrality	Giant size	n/a	n/a	5164	4111	**4027**
			Components number	n/a	n/a	4289	3685	**5972**
	High Energy	Nodes centrality	Giant size	n/a	n/a	654	n/a	**649**
			Components number	n/a	n/a	2789	n/a	**3291**
		Links centrality	Giant size	n/a	n/a	3390	n/a	**3299**
			Components number	n/a	n/a	4645	n/a	**6458**

2.7. Conclusion

In this chapter we introduce a new viewpoint to understand and model complex networks, which overlays a homogeneous artificial system over a network to unveil its hidden properties. We propose a novel algorithm to compute centrality in networks, dubbed GOT. We show that GOT can compute all node and link centralities, treated together, in a polylogarithmic time with respect to the number of nodes in the network. GOT has the computational simplicity of nature-inspired swarm algorithms, while performing human-behaviour like computations [**179**] (namely, egoistic behaviour). We demonstrate on thousands of simulated networks with different types of topologies, and on real-world networks, that GOT can compute the whole range of link and node strengths of any complex

network, while being more accurate, much faster, scalable and technologically viable than the state-of-the-art centrality metrics. Moreover, we have also used it to confirm well-established findings about a non-obvious behaviour of natural networks [**175**]. Natively, GOT permits to investigate much larger networks, which are not tractable with current algorithms - for instance GOT would require less than 9 seconds to compute the centrality of the one-billion network formed by all Facebook user devices.

Hence, we anticipate that our approach may lead to advances in various research fields for which nodes and links centrality is of crucial importance [**8**,**46**,**60**, **63**, **85**, **87**, **90**, **97**, **134**, **168**, **191**, **224**]. Thus, we consider that our viewpoint will start a novel class of methods in network science which natively incorporate the primordial property of real-world networks, i.e. decentralization, and which may change our understanding about the natural and human-made complex systems modelled by networks.

CHAPTER 3

Generative replay: towards memory-free online learning with artificial neural networks

In the previous chapter we saw how artificial intelligence can be used to improve network science algorithms. In this chapter, we start studying the opposite link. We show how network science can be used to improve artificial intelligence, by taking inspiration from biological neural network functional behavior to improve the functionality of Artificial Neural Networks (ANNs). More exactly, we tackle the complex problem of online learning with ANNs, which is in many cases difficult due to the need of storing and relearning large amount of previous experiences. This limitation can be partially surpassed using a mechanism conceived in the early 1990s, named experience replay. Traditionally, experience replay can be applied to all machine learning paradigms (i.e. unsupervised, supervised, and reinforcement learning). Recently, it has contributed to improving the performance of deep reinforcement learning. Yet, its application to many practical settings is still limited by the excessive memory requirements, necessary to explicitly store previous observations. From a biological sense of memory, the human brain does not store all observations explicitly, but instead it dynamically generates approximate reconstructions of those experiences for recall. In this chapter, inspired by this biological fact, we remedy the experience replay downside, by proposing a novel approach, dubbed Generative Replay (GR). Generative replay uses the sampling capabilities of generative models to generate approximations of past experiences, instead of recording them, as experience replay does. We demonstrate our approach using Restricted Boltzmann Machines (RBMs). Herein, we propose a novel method to train RBMs, dubbed Online Contrastive Divergence with Generative Replay (OCD_{GR}). Thus, the RBM can be trained online and does not require the system to store any of the observed data points. We highlight that generative replay is a broad concept, which, further on, may be used in combination with other types of generative artificial neural network models.

3.1. Introduction

Experience Replay (ER) [**116**] (dubbed interleaved learning in [**129**]) has been shown to be a successful mechanism in helping online learning algorithms to reuse past experiences. In ER, the data acquired during the online learning process is stored explicitly and presented repeatedly to the online learning algorithm, such as Reinforcement Learning (RL) [**5**], Deep Reinforcement Learning (DRL) [**133**], or supervised learning [**129**]. The ER process enables the learner to achieve good

This chapter is integrally based on: D.C. Mocanu, M. Torres Vega, E. Eaton, P. Stone, A. Liotta: *Online contrastive divergence with generative replay: Experience replay without storing data*, CoRR, 2016 (to be submitted for journal publication).

performance from limited training data, and helps to break temporal correlations in the observations which go against the independent and identically distributed (i.i.d) assumptions of many stochastic gradient-based algorithms [**180**]. Since ER uses recorded data in chunks, it has sometimes been deemed a batch learning approach [**94**]. In general, ER focuses on the reuse of observed data in its raw form as stored in memory, replaying it to the online learner. However, this causes ER to scale poorly, since the memory requirements increase as the environment and system requirements increase. One common practice is to limit the available memory of the ER mechanism and to either 1) discard the oldest experiences as the memory buffer becomes full and/or 2) prioritize the experiences [**180**].

From a biological sense of memory (i.e. hippocampal replay in [**129**]), the human brain does *not* store all observations explicitly, but instead it *dynamically generates approximate reconstructions* of those experiences for recall. This idea has also been applied to online learning through model-based learning as an alternative to ER. Such approaches indirectly reuse experiences by first modeling the environment, and then using that model to generate new data. This procedure is used by Dyna and other model-based learning approaches [**194**]. Building a model will generally require less memory than storing the raw data, and can diminish the effects of noise in the observations. However, model learning incurs additional computational costs and, more importantly, will introduce modeling errors that can significantly decrease performance [**194**]. For this reason, it is necessary to look for alternatives that are able to scale effectively (which is one of the biggest issues in ER) and yield performance results that are comparable with those obtained under ER, without its increased computational complexity.

At the same time, Restricted Boltzmann Machines (RBMs) [**187**], the original building blocks in deep learning models [**20**], besides providing (in an unsupervised manner) accurate weights for the deep belief networks initialization [**110**], have been shown to be very good density estimators and to have powerful generative capabilities [**144**, **177**]. Due to these capabilities, RBMs and models derived from them have been successfully applied to various problems also as standalone models. Examples of these applications are: modeling human choice [**163**], collaborative filtering [**176**], information retrieval [**70**], transfer learning [**28**], or multi-class classification [**104**]. However, in all of the above settings RBMs have been used offline, i.e. using offline training algorithms. This reduces drastically their capabilities to tackle real-world problems which can not be handled on server clouds using Graphic Processing Unit (GPU) computing, and require fast training algorithms capable of continuous learning when the environment is changing. For example, in the world of wireless sensor networks, which is by definition an environment with low-resources (e.g. memory, computational power, low energy), it would be extremely useful if lightweight devices could perform tasks such as anomaly detection on the observed time series, as exemplified in [**25**]. Recently, RBMs have been used to estimate the similarity between data distributions in various domains (e.g. image quality assessment [**137**], Markov decision processes [**10**]). We then hypothesize that due to these density estimation capabilities, RBMs could be used to perform anomaly detection directly on any wireless node if they could rely on an online training algorithm having low memory requirements (the best case would be to store none of the historical data). Still, to our knowledge, there are no dedicated algorithms to train RBMs in a fully online manner. The only currently available

solution is to employ ER mechanisms with memory, by following the successful examples from other deep learning models (e.g. [**133**]).

In this chapter, we combine the generative capabilities of RBMs with the biological inspiration behind experience replay, yielding a novel algorithm to train RBMs in online settings, which we call *Online Contrastive Divergence with Generative Replay* (OCD_{GR}). In comparison with state-of-the-art ER techniques, OCD_{GR} acts more like the experience replay concept in a biological sense. Instead of explicitly storing past observations in memory, it generates new training data dynamically to represent historical observations, using the generative capabilities of the RBM itself. In contrast to model-based learning approaches, which learn models from the environment [**194**], OCD_{GR} relies on the underlying RBM, which models only the observed data distribution—a substantially easier problem. OCD_{GR} derived methods may have a wide applicability to a variety of tasks (e.g. regression, classification, reinforcement learning, anomaly detection), but in this chapter we focus on demonstrating the benefits of OCD_{GR} over current ER approaches on the RBMs main task (i.e. distribution estimation), this being a must-have for any further developments. Thus, using 9 real-world datasets we show how OCD_{GR} outperforms ER in training RBMs, while having reduced memory requirements and an equivalent time complexity.

The remainder of this chapter is organized as follows. Section 3.2 presents background knowledge about experience replay and restricted Boltzmann machines, for the benefit of the non-specialist reader. Section 3.3 introduces our proposed method, while Section 3.4 describes the experiments performed and assesses the results. Finally, Section 3.5 concludes the chapter and presents further research directions.

3.2. Background and related work

In this section, we first discuss related work on ER. Next, background informations on RBMs and their offline training methods are presented.

3.2.1. Experience replay

Experience replay was first introduced for reinforcement learning in [**116**] and for supervised learning in [**129**]. Schematically, its functionality is depicted in Figure 3.2a for the specific case of RBMs. In short, the basic idea behind ER for any type of online learning model is to update the model parameters at discrete time steps t. To perform this update we start from the latest values of the model parameters, and we use the batch of data points observed in the environment between $t-1$ and t, together with same randomly chosen data points from the data stored in the memory and observed before $t-1$. Then, this procedure is repeated continuously, the termination criteria being specific to the type of problem under scrutiny.

A number of methods have subsequently been proposed to improve ER, aiming to model the environment and to optimize the performance of online learning. A complete review of ER applicability does not constitute a goal of this chapter. It can be found in [**94**], were it was shown how standard RL and batch approaches (ER) lead to comparable performance. Recently, ER and its variants have contributed to improving DRL [**133**]. In [**155**] the authors propose a form

of re-sampling in the context of DRL, which separates the learner experience into two parts: one for the positive rewards and for the negative ones, respectively. Further on, an online semi-supervised learning algorithm using deep hybrid Boltzmann machines and denoising autoencoders is proposed in [**162**]. However, all of these approaches require memory to explicitly store past observations for recall, making them less suitable to the online learning setting.

3.2.2. Restricted Boltzmann Machines (RBMs)

In this chapter, we use the generative capabilities of restricted Boltzmann machines to dynamically generate new training data during the online learning process, instead of explicitly storing and recalling past observations. We next review the mathematical details of RBMs. They were introduced in [**187**] as a powerful model to learn a probability distribution over its inputs. Formally, RBMs are generative stochastic neural networks with two binary layers: the hidden layer $\mathbf{h} = [h_1, h_2, .., h_{n_h}] \in \{0,1\}^{n_h}$, and the visible layer $\mathbf{v} = [v_1, v_2, .., v_{n_v}] \in \{0,1\}^{n_v}$, where n_h and n_v are the numbers of hidden neurons and visible neurons, respectively. In comparison with the original Boltzmann machine [**4**], the RBM architecture (Figure 3.1) is restricted to be a complete bipartite graph between the hidden and visible layers, disallowing intra-layer connections between the units. The energy function of an RBM for any state $\{\mathbf{v}, \mathbf{h}\}$ is computed by summing over all possible interactions between neurons, weights, and biases as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^\mathsf{T}\mathbf{v} - \mathbf{b}^\mathsf{T}\mathbf{h} - \mathbf{h}^\mathsf{T}\mathbf{W}\mathbf{v} \ , \tag{3.1}$$

where $\mathbf{W} \in \mathbb{R}^{n_h \times n_v}$ is the weighted adjacency matrix for the bipartite connections between the visible and hidden layers, and $\mathbf{a} \in \mathbb{R}^{n_v}$ and $\mathbf{b} \in \mathbb{R}^{n_h}$ are vectors containing the biases for the visible and hidden neurons, respectively. For convenience, we can bundle the RBM's free parameters together into $\Theta = \{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$. Functionally, the visible layer encodes the data, while the hidden layer increases the learning capacity of the RBM model by enlarging the class of distributions that can be represented to an arbitrary complexity [**197**]. Due the binary state of the neurons, the free energy of the visible units may be computed as [**20**]:

$$\mathcal{F}(\mathbf{v}) = -\mathbf{a}^\mathsf{T}\mathbf{v} - \sum\nolimits_j \log(1 + \exp\left(b_j + \mathbf{W}_{j:}\mathbf{v}\right)) \ , \tag{3.2}$$

where $\mathbf{W}_{j:}$ represents the j^{th} row of the matrix $\mathbf{W}$. The activations of the hidden or visible layers are generated by sampling from a sigmoid $\mathcal{S}(\cdot)$ according to: $P(\mathbf{h} = 1|\mathbf{v}, \Theta) = \mathcal{S}(\mathbf{b} + \mathbf{W}\mathbf{v})$ and $P(\mathbf{v} = 1|\mathbf{h}, \Theta) = \mathcal{S}(\mathbf{a} + \mathbf{W}^\mathsf{T}\mathbf{h})$.

3.2.3. Offline RBM training via contrastive divergence

The RBM parameters can be learned effectively by following the log-likelihood gradient computed over a training set $\mathcal{D}$, with n_v-dimensional binary instances. The log-likelihood gradient is given by:

$$\mathbb{E}_{\hat{P}}\Big[\frac{\partial(logP(\mathbf{v}))}{\partial\theta}\Big] = -\mathbb{E}_{\hat{P}}\Big[\frac{\partial\mathcal{F}(\mathbf{v})}{\partial\theta}\Big] + \mathbb{E}_{P}\Big[\frac{\partial\mathcal{F}(\mathbf{v})}{\partial\theta}\Big] \ , \tag{3.3}$$

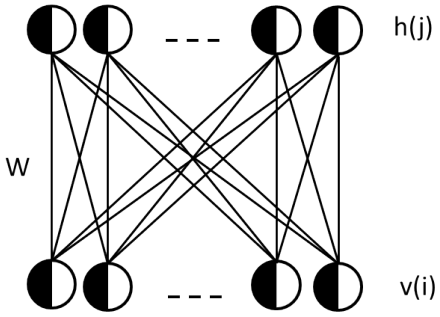


Figure 3.1 – Restricted Boltzmann Machine architecture.

where $\hat{P}$ represents the empirical distribution of $\mathcal{D}$ and $\mathbb{E}_P$ is the expectation computed under the model distribution [**20**]. However, sampling from P to compute the free energy and running long Monte-Carlo Markov Chains (MCMC) to obtain an estimator of the log-likelihood gradient is usually intractable. Due to this intractability, Hinton proposed an approximation method called Contrastive Divergence (CD) [**81**], which solves the above problem by making two approximations. The first approximation is to replace the average over all possible inputs from the second term of Equation 3.3 by a single sample. The second approximation is to run each MCMC chain for only a specific number of steps (n_{CD}), starting from a data point $\mathbf{v}^0 \in \mathcal{D}$, as follows:

$$\mathbf{v}^0 \overset{P(\mathbf{h}|\mathbf{v}^0)}{\longmapsto} \mathbf{h}^0 \overset{P(\mathbf{v}|\mathbf{h}^0)}{\longmapsto} \mathbf{v}^1 \overset{P(\mathbf{h}|\mathbf{v}^1)}{\longmapsto} \mathbf{h}^1 \dashmapsto \mathbf{v}^{n_{CD}} \longmapsto \mathbf{h}^{n_{CD}} \ .$$

The free parameters can then be updated afterwards via:

$$\Delta\Theta = \frac{\partial \mathcal{F}(\mathbf{v^0})}{\partial \theta} - \frac{\partial \mathcal{F}(\mathbf{v}^{n_{CD}})}{\partial \theta} \ , \tag{3.4}$$

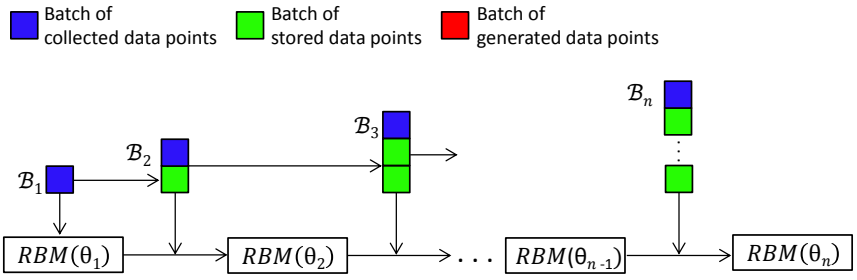
yielding the following update rules for the free parameters of binary RBMs:

$$\begin{aligned} \Delta W_{ji} &\propto v_i^0 h_j^0 - v_i^{n_{CD}} h_j^{n_{CD}} && \text{for } 1 \leq i \leq n_v, 1 \leq j \leq n_h \\ \Delta a_i &\propto v_i^0 - v_i^{n_{CD}} && \text{for } 1 \leq i \leq n_v \\ \Delta b_j &\propto h_j^0 - h_j^{n_{CD}} && \text{for } 1 \leq j \leq n_h \ . \end{aligned} \tag{3.5}$$

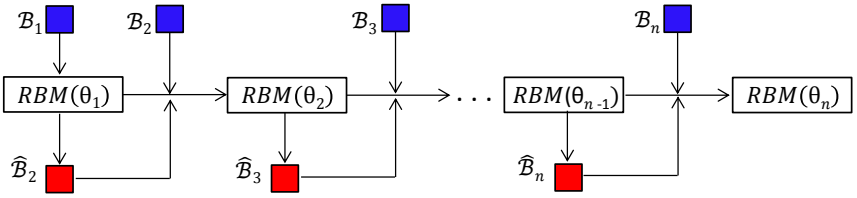
Several other variants of contrastive divergence have been proposed to train RBMs offline. Examples of these are: persistent contrastive divergence [**200**], fast persistent contrastive divergence [**201**], parallel tempering [**52**], and the replace of the Gibbs sampling with a transition operator to obtain a faster mixing rate and an improved learning accuracy without affecting the computational costs [**33**]. Yet, in this chapter we use the original CD [**81**], as it is easily adaptable to online settings, and at the same time it is widely used and allows for a direct comparison with other results reported in the literature.

3.3. Online contrastive divergence with generative replay

This section presents our novel algorithm to train RBMs online: *Online Contrastive Divergence with Generative Replay*. Our approach adapts the standard CD algorithm (see Section 3.2.3) to the online learning setting, and uses dynamic generation of data as a replay mechanism. We show how an RBM trained via



(a) RBM training via Experience Replay



(b) RBM training via OCD_{GR}

Figure 3.2 – A comparison of ER with memory (a) and OCD_{GR} (b) for training RBMs online. Each subscript $\cdot_t$ represents a discrete time step t. $\mathcal{B}_t$ represents a batch of observed data between $t-1$ and t, while $\hat{\mathcal{B}}_t$ represents samples generated by the RBM model using the free parameters Θ_{t-1} (i.e. the parameters values at time $t-1$).

OCD_{GR} can have the same functionality as training via ER. However, OCD_{GR} provides the significant advantage of not needing to explicitly store past observations in memory, substantially reducing its space complexity. To our knowledge, this capability is unique, since the state-of-the-art experience replay mechanisms require a memory dataset to store all historical data, or at least a part of them.

3.3.1. Intuition and formalism

Our algorithm is motivated by the fact that hippocampal replay [**129**] in the human brain does *not* recall previous observations explicitly, but instead it generates *approximate reconstructions* of those past experiences for recall. At the same time, RBMs can generate good samples of the incorporated data distribution via Gibbs sampling [**20**]. Intuitively, by using those generated samples (instead of previous observations from stored memory as in ER) during the online training process, any RBM model can retain knowledge of past observations while learning new ones.

Before entering into the technical details of our proposed method, we mention that further on we use the following notations: $\mathcal{B}_t$ represents a batch of observed data between time $t-1$ and t, while $\hat{\mathcal{B}}_t$ represents samples generated by the RBM model using the free parameters Θ_{t-1} (i.e. the parameters values at time $t-1$). Figure 3.2 summarizes the main differences between the OCD_{GR} (Figure 3.2b) and ER mechanisms (Figure 3.2a) for training RBMs online, showing how ER explicitly stores previous observations in memory for recall, while OCD_{GR} dynamically

generates samples from its current model of the input data distribution. Clearly, the memory used by ER increases linearly with the amount of data observed (up to a fixed limit for memory-bounded ER methods), while by contrast, OCD_{GR} maintains the same memory footprint throughout the training process. Also, note that the ER mechanism with memory does not have the Markov property, since for ER, $P(\Theta_t)$ is dependent upon $\{\Theta_{t-1}, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t\}$, while OCD_{GR} has the Markov property that $P(\Theta_t)$ depends only upon Θ_{t-1} and $\mathcal{B}_t$. This is an important aspect for an algorithm which runs for an indefinite amount of time, as may occur in many real-time systems. Formally, OCD_{GR} is a continuous-time Markov chain with finite (countable) state space $\mathcal{X}$, given by a family $\{RBM_t = RBM(t)\}_{t>0}$ of $\mathcal{X}$ such that:

(1) $t \mapsto RBM(t)$ are right-continuous step functions, and
(2) $\forall s, s_1, ..., s_k \in \mathcal{X}$, and every sequence of times $t_1 < t_2 < ... < t_k < t_{k+1}$, it holds that:

$$P\big(RBM(t_{k+1}) = s \mid RBM(t_k) = s_k, ..., RBM(t_1) = s_1\big) = P\big(RBM(t_{k+1}) = s \mid RBM(t_k) = s_k\big) \ .$$

The second condition is the natural continuous-time analogue of the Markov property, and it requires that the future is conditionally independent of the past given the present RBM. A continuous time Markov chain is a non-lattice semi-Markov model, so it has no concept of periodicity. Consequently, the long-runtime averages equals the limiting probabilities, and it has an equilibrium distribution.

3.3.2. Algorithm

OCD_{GR} is presented in Appendix B as Algorithm B.2. As input, the algorithm accepts various meta-parameters, two of them being specific for OCD_{GR}, while the others are common to all RBM models (Algorithm B.2, line 2). The two meta-parameters specific for OCD_{GR} are the number of Gibbs sampling steps for the generation of the new training data points (n_{Gs}), and the number of new data points generated by the RBM with Gibbs sampling ($n_{\hat{B}}$). The common RBMs meta-parameters include the number of hidden neurons (n_h), the number of visible neurons (n_v) (which is given by the dimensionality of the data), the number of CD steps (n_{CD}), the number of training epochs (n_E), the number of data points stored in a mini-batch before the RBM parameters are updated (n_B), the learning rate (α), the momentum (ρ), and the weight decay (ξ). Except for the two OCD_{GR} specific parameters, the settings for the others are discussed in [**83**].

The algorithm first initializes the RBM's free parameters Θ and the discrete time step t (lines 3–5). Each time step, the algorithm observes a new data instance, collecting n_B new data points into a mini-batch $\mathcal{B}_t$ (lines 8–10). After observing n_B new data points, OCD_{GR} updates the RBM's parameters (line 11–41). The update procedure proceeds in two phases:

Dynamic generation of historical data (lines 14–22) As it has been shown in [**52**] and in [**39**] that RBMs can sample uniformly from the state space, we generate $n_{\hat{B}}$ new training data points to represent past observations based on the data distribution modeled by the RBM at time $t-1$; these generated data points are collected into the set $\hat{\mathcal{B}}_t$. To obtain good data points with high representational

power, we perform Gibbs sampling starting from random values of the hidden neurons drawn from a uniform distribution $\mathcal{U}(0, 1)$.

CD update with generative replay (lines 26–41) In the second phase, we update the RBM's weights and biases (Θ) using standard CD for a number of epochs, computing the update only over the most recent mini-batch composed by the union of $\mathcal{B}_t$ and $\hat{\mathcal{B}}_t$. This most recent mini-batch consists of (1) the data points observed between time $t-1$ and time t and (2) the data points generated by the RBM at time t. Note that line 39 of Algorithm B.2 contains the general form of the update equation, in which Ψ^+ (statistics collected from the data) and Ψ^- (statistics collected from the model) can be computed for each free parameter type using Equation 3.5. Finally, the data points observed between $t-1$ and t, and the data points generated with the RBM at time t are deleted from memory, and OCD$_{GR}$ advances to the next discrete time step $t+1$ (lines 42–46).

It is easy to observe that an RBM trained with OCD$_{GR}$ acts at any time step t as a generative replay mechanism to provide repetition of approximated past experiences, providing a memory-free alternative to ER. In our experiments, we demonstrate empirically that OCD$_{GR}$ can be used successfully to train RBMs in an online setting.

3.3.3. Computational complexity

The primary difference between ER and OCD$_{GR}$ at each discrete time step t is that ER has to recall random data from memory, while OCD$_{GR}$ generates the data via Gibbs sampling. For ER, the memory recall time depends upon the hardware platform and the programming environment (e.g. Matlab, C++), and so is not easily quantifiable. For OCD$_{GR}$, the *dynamic generation of historical data* phase using Gibbs sampling requires, on one side, a small number of matrix multiplication (which may be parallelized) and are linearly dependent by n_{Gs} and, on the other side, the computation of the sigmoid functions for the visible and hidden neurons. This yields a per-update time of $\mathcal{O}(2n_{Gs}n_v n_h + n_{Gs}n_h + n_{Gs}n_v)$, which in the typical case of $n_{Gs} = 1$, reduces to $\mathcal{O}(n_v n_h)$.

3.4. Experiments and results

3.4.1. Evaluation method

Firstly, we considered a toy scenario (i.e. an artificially generated dataset) to illustrate the OCD$_{GR}$ behavior. Secondly, we evaluated OCD$_{GR}$ performance on the MNIST dataset[1] of handwritten digits, and on the UCI evaluation suite [**72**]. Thus, overall, the evaluation was performed on 9 datasets coming from different domains, which are detailed in Table 3.1.

To simulate the online learning setting, each training instance was fed to the RBM training algorithm only once in a sequential manner in one of two orders: 1) a worst-case scenario, in which the data instances are presented in order of the classes, and 2) a more realistic scenario, in which the instances are ordered randomly.

[1] http://yann.lecun.com/exdb/mnist/. Last visit on 26 September 2016.

Table 3.1 – Datasets characteristics.

	Dataset	Dataset Properties			
		Domain	Features [#]	Train samples [#]	Test samples[#]
	MNIST	digits	784	60000	10000
UCI evaluation suite	ADULT	households	123	5000	26147
	Connect4	games	126	16000	47557
	DNA	biology	180	1400	1186
	Mushrooms	biology	112	2000	5624
	NIPS-0-12	documents	500	400	1240
	OCR-letters	letters	128	32152	10000
	RCV1	documents	150	40000	150000
	Web	Internet	300	14000	32561

The update procedure of the RBM's free parameters was triggered each time after the system had observed and collected 100 data points (i.e. $n_B = 100$). To find the best meta-parameters specific for OCD_{GR} (i.e. $n_{\hat{B}}$, n_{Gs}) we conducted a random search. Based on this small experiment, before each update procedure took place, we generated another $n_{\hat{B}} = 300$ data points according with Algorithm B.2, lines 14–22, with n_{Gs} set to 1. Moreover, another reason to set a small number of steps for Gibbs sampling when new data points are generated (i.e. $n_{Gs} = 1$) is given by the fact that if we use samples from the model for both components of the gradient (i.e. Equation 3.3), these will cancel out in expectation. Except when specified otherwise, the other meta-parameters used usually in the RBM training process were set to typically values, such as $n_E = 10$, $n_{CD} = 1$, $\alpha = 0.05$, $\rho = 0.9$ (except the first 5 training epochs in which $\rho = 0.5$), and $\xi = 0.0002$, following [**83**]. Please note that even a higher number of contrastive divergence steps (i.e. n_{CD}) may lead to a better performance on some specific datasets, i.e. MNIST, it leads also to an increasing amount of computations. As our goal was to propose a fast algorithm to train RBMs in an online manner also on low-resources devices, we preferred to perform most of our experiments using just 1 step contrastive divergence.

3.4.2. Behavior illustration (toy scenario)

To easy visualize the quality of the samples generated by RBMs trained with OCD_{GR} (RBM_{OCD}) we have considered a toy scenario with artificially generated data and an RBM with 100 visible neurons and 50 hidden neurons. For training we have created 10000 data points (each data point being a binary vector of 100 elements) split in 10 classes of 1000 data points each, as following. For Class 1 $p(v_i = 1) = 0.3 \Leftrightarrow 1 \leq i \leq 10$ and $p(v_i = 1) = 0 \Leftrightarrow 11 \leq i \leq 100$, and so on up to Class 10 for which $p(v_i = 1) = 0.3 \Leftrightarrow 91 \leq i \leq 100$ and $p(v_i = 1) = 0 \Leftrightarrow 1 \leq i \leq 90$. During training, we have firstly observed all data instances belonging to Class 1, and after that we have generated 1000 samples with the trained RBM_{OCD}. Next, we have continued the training procedure using all data points belonging to Class 2, and then we have generated another 1000 samples, and further on we repeated this procedure until all 10 classes have been considered. To classify the samples generated by RBM_{OCD} we used k-nearest neighbors. Figure 3.3 shows

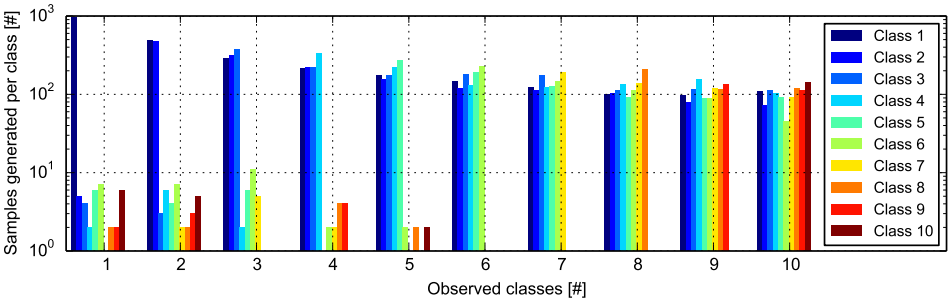


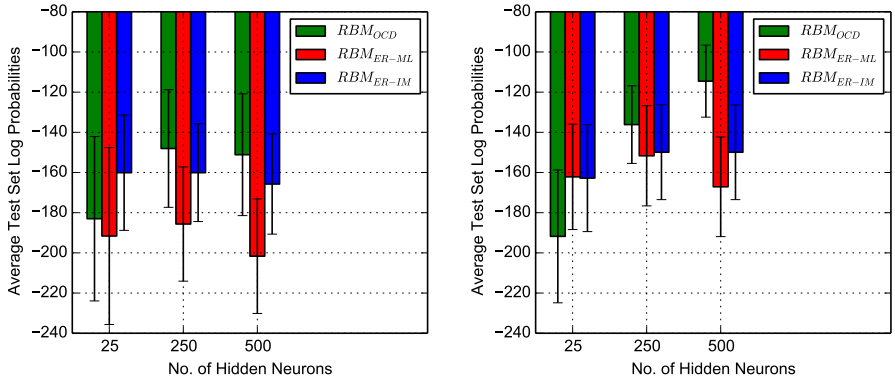
Figure 3.3 – Illustration of the OCD$_{GR}$'s behavior (toy scenario). At any time, the samples generated by RBM_{OCD} are distributed equally among all observed classes. The y-axis uses the log-scale.

that OCD$_{GR}$ behaves as expected: as new classes are observed the RBM_{OCD} enlarges its encoded distribution.

3.4.3. Comparative evaluation

We compared our proposed method, RBM_{OCD}, against 1) RBMs trained using Experience Replay with a Memory Limit ($RBM_{ER\text{-}ML}$) and 2) RBMs trained using Experience Replay with Infinite Memory ($RBM_{ER\text{-}IM}$). For a fair comparison, in the case of $RBM_{ER\text{-}ML}$ we limited the number of data points stored in memory to occupy approximately the same number of bytes as the parameters of RBM_{OCD}. We highlight that by allowing $RBM_{ER\text{-}ML}$ to have an experiences memory of the same size as the RBM_{OCD} parameters means that, in fact, $RBM_{ER\text{-}ML}$ needs twice the memory size of RBM_{OCD}, as it needs also some memory to store its own parameters. In contrast, we allow $RBM_{ER\text{-}IM}$ to store all observed data points in memory. To train both experience replay models, i.e. $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$, we use a similar algorithm to Algorithm B.2. The only main difference is that instead of generating new samples using the RBM models themselves (Algorithm B.2, lines 11-23), we retrieve those samples from the replay memory. For both models, $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$, we used 300 randomly chosen data points from the memory of past experiences and the same meta-parameters values as for RBM_{OCD}. To quantify the generative performance of the trained networks, we used Annealed Importance Sampling (AIS) with the same parameters as in the original study [**177**] to estimate the partition function of the RBMs and to calculate their log probabilities. On each dataset, after all training data points were given to the learner, we computed the average log probabilities on the entire test set. As the RBM's training objective is to maximize the expected log probabilities on the training data, we may expect that also on the testing data the average log probabilities of the scrutinized RBMs to be as high as possible.

3.4.3.1. *Worst case scenario: sorted order.* In the first scenario, we have used the binarized MNIST dataset. During training, the data instances were ordered sequentially in ascending order of the digits $(0, 1, \ldots, 9)$, making it a difficult scenario for online learning. For each algorithm, we considered various numbers of hidden neurons ($n_h \in \{25, 250, 500\}$), and 784 visible neurons (i.e. 28×28 binary image). Figure 3.4a shows that RBM_{OCD} outperforms $RBM_{ER\text{-}ML}$ in all



(a) Worst case scenario, with instances in ascending order by digit.

(b) Realistic scenario, where the instances are ordered randomly.

Figure 3.4 – Performance on the MNIST dataset. For each model, we plot the average log probabilities computed on the entire test set, with error bars representing the standard deviation of the average log probabilities computed on each digit class.

cases, regardless of the number of hidden neurons. Moreover, it outperforms even $RBM_{ER\text{-}IM}$ when it has enough representational power (i.e. 250 and 500 hidden neurons). It is interesting to see that while the generative power of RBM_{OCD} increases with the number of hidden neurons, $RBM_{ER\text{-}IM}$ is not significantly affected when given more hidden neurons. Further, the $RBM_{ER\text{-}ML}$ model *loses* its generative power when the number of hidden neurons is increased. These results may be explained by the fact that having more hidden neurons helps RBM_{OCD} to better model the data distribution. In contrast, in the case of experience replay mechanisms with memory, a larger RBM would need more past-experience training data to avoid forgetting the distribution of the first observed data points, especially in the case of $RBM_{ER\text{-}ML}$. This situation does not occur for RBM_{OCD}, due to the fact the data points generated randomly by the RBM itself using Gibbs sampling approximate well the distribution of the past-experience data. For the sake of clarification, we mention that even if at a first look an inter-class standard deviation of $30 - 40$ *nats* for all online trained models is striking, this is, in fact the same as the one obtained on offline trained RBMs.

3.4.3.2. *Realistic scenario: random order.* In the second more realistic scenario, the training instances were presented sequentially in random order. As this is an usually encountered situation, herein, besides the MNIST dataset, we have used also the UCI evaluation suite [**72**]. The latter one contains 8 real-world binary datasets from various domains, specially selected to evaluate the performance of density estimation models.

MNIST dataset. Figure 3.4b shows that RBM_{OCD} outperforms both models, $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$, when it has enough representational power (i.e. 250 and 500 hidden neurons). As in the previous experiment, the generative performance of RBM_{OCD} increases as the number of hidden neurons increases, but the gain in performance is even higher in this situation, culminating with -114.53 *nats* in the case of an RBM with 500 hidden neurons. Thus, RBM_{OCD} outperforms $RBM_{ER\text{-}IM}$ by 35.39 *nats* at the same number of hidden neurons. In fact,

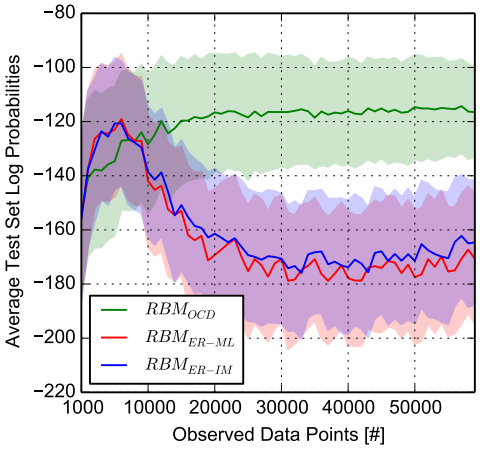


Figure 3.5 – Model performance over time on the realistic scenario using the MNIST dataset. The lines represent the average log probabilities computed on the entire test set, while the shadowed areas represent the standard deviation of the average log probabilities computed on each digit class.

RBM_{OCD} with 500 hidden neurons outperforms by 11.01 *nats* even the state-of-the-art results reported by Salakhutdinov and Murray [**177**] (see Table 3.2, third row) for an RBM with 500 hidden neurons trained completely offline with standard one-step contrastive divergence. Besides the improved average performance on the entire test set, also observe that as the number of hidden neurons increases in the case of RBM_{OCD}, the standard deviation (computed on the average log probabilities from each digit class) decreases. The smaller standard deviation implies that the model represents all classes well, without imbalance.

To better understand RBM_{OCD}'s behavior, we performed an additional experiment. We again trained RBM_{OCD}, $RBM_{ER\text{-}ML}$, and $RBM_{ER\text{-}IM}$ models, each with 500 hidden neurons. However, in this experiment, we measured the performance on the MNIST test set during the training phase at intervals of 1,000 observed data points. Figure 3.5 shows an interesting behavior for all three models. The RBM_{OCD} has a very stable learning curve which increases over time. In contrast, $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$ show unstable learning curves. This behavior can be explained by the fact that when the probability of selecting for replay any past observed data point decreases below a certain threshold, then the subset of the selected data points for replay no longer represents well the distribution of the past-experience data, and the models become over-fitted. To avoid this situation, the number of selected data points from the replay memory would need to increase linearly with the number of observations. However, this solution is infeasible as it would induce a linear increase in the computational complexity of $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$ over time, leading to unviable online learning algorithms. In contrast, RBM_{OCD} is a Markov chain and it is not affected by this situation, explaining why RBM_{OCD} outperforms $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$ after observing approximately 8,000 instances. Thus, the reduced memory requirements and stable learning behavior make RBM_{OCD} a viable model in online learning settings, where data may

Table 3.2 – Realistic scenario. The results are given for RBMs with $n_h = 500$ and $n_{CD} = 1$. On the MNIST dataset the offline RBM results are taken from [**177**], while on the UCI evaluation suite the offline RBMs results are taken from [**72**].

	Dataset	**Online models**			**Offline model**
		RBM_{OCD}	$RBM_{ER\text{-}IM}$	$RBM_{ER\text{-}ML}$	RBM
	MNIST	**-114.52**	-151.67	-167.11	-125.53
UCI evaluation suite	ADULT	-19.64	-18.08	**-17.28**	-16.26
	Connect4	-16.28	**-16.03**	-17.64	-22.66
	DNA	**-103.14**	-111.81	-114.84	-96.74
	Mushrooms	**-16.64**	-20.38	-17.58	-15.15
	NIPS-0-12	**-290.06**	-365.03	-339.82	-277.37
	OCR-letters	**-47.61**	-51.35	-53.85	-43.05
	RCV1	**-53.28**	-56.34	-79.06	-48.88
	Web	-33.47	**-32.58**	-35.07	-29.38

come continuously for an indefinite period of time, even if its initial learning curve grows slower than the ones of $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$.

In our final experiment on the MNIST dataset, we varied the number of contrastive divergence steps, training an RBM_{OCD} with 500 hidden neurons using 3 steps and 10 steps of contrastive divergence. Similarly to the RBM's behavior reported by Salakhutdinov and Murray [**177**], further CD steps improved the generative performance of these models. In the case when $n_{CD} = 3$, the average log probabilities on the MNIST test set was -108.96; for $n_{CD} = 10$, it was -104.31.

UCI evaluation suite. In this case, we have trained RBM_{OCD}, $RBM_{ER\text{-}ML}$, and $RBM_{ER\text{-}IM}$ with $n_h = 500$, and n_v set to the number of features of each dataset. The results reflected in Table 3.2 show that RBM_{OCD} outperforms $RBM_{ER\text{-}ML}$ and $RBM_{ER\text{-}IM}$ on 5 out of 8 datasets, while on the other 3 it has a very close generative performance to the top performer. Overall, we may observe that as the size of the dataset increases, or as the data distribution becomes more complex, RBM_{OCD} has a clear advantage over $RBM_{ER\text{-}ML}$ or $RBM_{ER\text{-}IM}$.

In all experiments performed, we observed that our MATLAB implementations of all algorithms completed in approximately the same time. Given the same RBM configuration, $RBM_{ER\text{-}IM}$ was slightly slower than RBM_{OCD}, which was slightly slower than $RBM_{ER\text{-}ML}$. However, the differences were on the order of few milliseconds. In $RBM_{ER\text{-}IM}$, the difference increased slightly with the number of data points saved in memory, but still it was on the milliseconds order.

3.5. Conclusion

We have proposed a novel method, OCD_{GR}, to train RBMs in online settings. Unlike current experience replay methods which directly recall recorded observations from memory, OCD_{GR} uses the generative capabilities of RBMs to dynamically simulate past experiences. As a consequence, it does not need to store past observations in memory, substantially reducing memory requirements. We demonstrated that RBMs trained online with OCD_{GR} outperform RBMs trained online using experience replay with memory. In few exceptions, the performance was comparable. We highlight that in some cases RBM_{OCD} even outperforms conventional RBMs trained *offline* with standard contrastive divergence.

In future work, we intend to better understand the effect that the various OCD_{GR} meta-parameters (especially the relation between the number of generated samples and the number of observed samples) have on the RBM's generative performance. Also, we intend to extend the generative replay concept to other suitable generative models, e.g. deep Boltzmann machines, autoencoders. Other interesting research directions, would be to use RBMs trained with OCD_{GR} as follows: (1) to perform anomaly detection in low-resources devices; (2) to control DRL algorithms by generating RL atomic operations instead of using experience replay mechanisms with memory to store them; and (3) to perform online supervised learning by generating input-output pairs for the online training of deep artificial neural networks.

CHAPTER 4

Scale-free restricted Boltzmann machines

While in the previous chapter we introduced the generative replay concept to decrease the memory requirements of online learning with Restricted Boltzmann Machines (RBMs), herein we tackle the computational time needed for training and exploiting RBMs. We employ techniques from network science aiming to reduce their number of parameters (i.e. weighted connections between neurons). In turn, this reduction leads to significant improvements in computational time. Thus, our main contribution in this chapter is to look at RBMs from a topological perspective, bringing insights from network science. Firstly, here we show that RBMs and Gaussian RBMs (GRBMs) are bipartite graphs which naturally have a small-world topology. Secondly, we demonstrate both on synthetic and real-world datasets that by constraining RBMs and GRBMs to a scale-free topology (while still considering local neighborhoods and data distribution), we reduce the number of weights that need to be computed by a few orders of magnitude, at virtually no loss in generative performance. Thirdly, we show that, for a fixed number of weights, our proposed sparse models (which by design have a higher number of hidden neurons) achieve better generative capabilities than standard fully connected RBMs and GRBMs (which by design have a smaller number of hidden neurons), at no additional computational costs.

4.1. Introduction

Since its conception, deep learning [**20**] has been a subject of intensive study due to its broad set of applications. It has been applied to different real-world machine learning problems such as audio recognition [**114**], activity recognition [**141**], image understanding [**110**], or to theoretical research areas such as reinforcement learning [**133**], transfer learning [**28**]. Deep learning models are artificial neural networks with multiple layers of hidden neurons, which have connections only among neurons belonging to consecutive layers, but have no connections within the same layers. In general, these models are composed by basic building blocks, such as Restricted Boltzmann Machines (RBMs) [**187**]. In turn, RBMs have proven to be successful not just in providing good initialization weights in deep architectures (in both supervised and unsupervised learning), but also as standalone models in other types of applications. Examples are density estimation to model human choice [**163**], collaborative filtering [**176**], information retrieval [**70**], or multi-class classification [**104**]. Thus, an important research direction is to improve the performance of RBMs on any component (e.g. computational time, generative and discriminative capabilities).

This chapter is integrally based on: D.C. Mocanu, E. Mocanu, P. Nguyen, M. Gibescu, A. Liotta: *A topological insight into restricted Boltzmann machines*, Machine Learning 104 (2016), no. 2, 243-270.

The main contribution of this chapter is to look at the deep learning basic building blocks, i.e. RBMs and Gaussian RBMs (GRBMs) [**80**], from a topological perspective, bringing insights from network science. This is an extension of graph theory which analyzes real world complex networks [**192**]. Firstly, we study the topological characteristics of typical fully connected RBMs and GRBMs, finding that these exhibit a small-world topology [**218**]. We then hypothesize that by constraining the topology to be also scale-free [**19**] it is possible to reduce the size of fully connected RBMs and GRBMs models, as it has been shown by [**51**] that the networks with scale-free topology are sparse. We introduce a method to make small-world, scale-free topologies while still considering local neighborhoods and data distribution. We dub the resulting models as compleX Boltzmann Machine (XBM) and Gaussian compleX Boltzmann Machine (GXBM), respectively.

An interesting finding is that constraining such XBM and GXBM topologies at their inception leads to intrinsically sparse networks. This brings considerable advantages compared to typical state-of-the-art methods in which sparsity is enforced as an aftermath, that is during testing (exploitation) phase [**113**, **124**, **172**, **195**, **216**]. In turn, XBM and GXBM have a considerably smaller number of weights, which further on contributes to considerably faster computational times (proportional to the number of weights in the model), both in the training and testing phases. What is more, we found that the proposed topology imposes an inductive bias on XBMs and GXBMs, which leads to better statistical performance than RBMs and GRBMs.

Our comparative study is based on both simulated and real-world data, including the Geographical origin of music dataset [**229**], the MNIST digits dataset, CalTech 101 Silhouettes dataset [**128**], and the 8 datasets from the UCI evaluation suite [**105**]. We show that, given the same number of hidden neurons, XBM and GXBM have similar or relatively close capabilities to RBM and GRBM, but are considerably faster, thanks to their reduced amount of weights. For instance, in a network of 100 visible and 100 hidden neurons, the reduction in weights was by one order of magnitude. A network with 1000 visible and 1000 hidden neurons led to a reduction in weights by two orders of magnitude. Additionally, given the same amount of weights to XBMs/GXBMs and to RBMs/GRBMs, XBMs/GXBMs have a higher amount of hidden neurons than RBMs/GRBMs, due to their designed sparse connectivity. In this context, we show that XBMs/GXBMs achieve better generative capabilities than the fully-connected RBMs/GRBMs.

The remaining of this chapter is organized as follows. Section 4.2 presents background knowledge about Boltzmann machines and complex networks for the benefit of the non-specialist reader and highlights the key motivations of our work. Section 4.3 discusses the relation between deep leaning and network science and details the mathematical models of our proposed methods. Section 4.4 describes the experiments performed and analyzes the results. Finally, Section 4.5 concludes the chapter and presents directions of future research.

4.2. Background and motivations

4.2.1. Boltzmann machines

Originally derived in [**4**], a Boltzmann machine is a network of symmetrically connected stochastic binary units (or neurons). To formalize a Boltzmann machine, and its variants, three main ingredients are required, namely an energy function providing scalar values for a given configuration of the network, the probabilistic inference and the learning rules required for fitting the free parameters. This bidirectional connected network with stochastic nodes has no unit connected with itself. However, Boltzmann machines with unconstrained connectivity are unfeasible in practical problems due to the intractable inference. A critical step in taming computational complexity is to add constraints to the connectivity network, which is what makes Boltzmann machines applicable to real world problems.

In 1987, Smolensky presented restricted Boltzmann machine that could learn a probability distribution over its set of inputs [**187**]. The model architecture was restricted by not allowing intra-layer connections between the units, as depicted in Figure 4.2 (left). Since their conception, different types of Boltzmann machines have been developed and successfully applied. Yet most of these variations preserve some fundamental characteristics. RBMs are generative stochastic neural networks consisting of two binary layers, the visible layer, $\mathbf{v} = [v_1, v_2, .., v_{n_v}]$, and the hidden layer, $\mathbf{h} = [h_1, h_2, .., h_{n_h}]$, with n_v being the number of visible neurons and n_h the number of the hidden ones. Formally, the energy function of RBMs for any state $\{\mathbf{v}, \mathbf{h}\}$ is computed by summing over all possible interactions between neurons, weights and biases, as follows:

$$E(v, h) = -\sum_{i,j} v_i h_j W_{ij} - \sum_i v_i a_i - \sum_j h_j b_j \tag{4.1}$$

where W_{ij} denotes the connection between the visible neuron i and the hidden neuron j, a_i is the bias for visible neuron i and b_j is the bias for hidden neuron j. The term $\sum_{i,j} v_i h_j W_{ij}$ represents the total energy between neurons from different layers, $\sum_i v_i a_i$ represents the energy of the visible layer and $\sum_j h_j b_j$ the energy of the hidden layer. The inference in RBMs is stochastic. For any hidden neuron j the conditional probability is given by $p(h_j|\mathbf{v}) = \mathcal{S}(b_j + \sum_i v_i W_{ij})$, and for any visible unit i it is given by $p(v_i|\mathbf{h}) = \mathcal{S}(a_i + \sum_j h_j W_{ij})$, where $\mathcal{S}(\cdot)$ is a sigmoid function.

Later on, Hinton and Salakhutdinov [**80**] have extended the RBMs models to make them suitable for a large number of applications with real-valued feature vectors. They used exponential family harmoniums results from [**220**] and developed the Gaussian Restricted Boltzmann Machine (GRBM) model that, like RBMs, forms a symmetrical bipartite graph. However, the binary units from the visible layer $\mathbf{v}$ are replaced by linear units with Gaussian noise. The hidden units $\mathbf{h}$ remain binary. Therein, the total energy function for a state $\{\mathbf{v}, \mathbf{h}\}$ of GRBMs is calculated in a similar manner to RBMs, but includes a slight change to take into consideration the Gaussian noise of the visible neurons, as defined in Equation 4.2.

$$E(v, h) = -\sum_{i,j} \frac{v_i}{\sigma_i} h_j W_{ij} - \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j h_j b_j \tag{4.2}$$

where, the term $\sum_{i,j} \frac{v_i}{\sigma_i} h_j W_{ij}$ gives the total energy between neurons from different layers, $\sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2}$ is the energy of the visible layer, and σ_i represents the standard deviation of the visible neuron i. The stochastic inference for any hidden neuron j can be done as for RBMs, while for any visible unit i is made by sampling from a Gaussian distribution, defined as $\mathcal{N}(a_i + \sum_j h_j W_{ij}, \sigma_i^2)$.

Parameters of RBM and GRBM models are fitted by maximizing the likelihood function. In order to maximize the likelihood of the model, the gradients of the energy function with respect to the weights have to be calculated. Because of the difficulty in computing the derivative of the log-likelihood gradients, Hinton [**81**] proposed an approximation method called Contrastive Divergence (CD). In maximum likelihood, the learning phase actually minimizes the Kullback-Leibler (KL) measure between the input data distribution and the model approximation. Thus, in CD, learning follows the gradient of:

$$CD_n \propto D_{KL}(p_0(\mathbf{x})||p_\infty(\mathbf{x})) - D_{KL}(p_n(\mathbf{x})||p_\infty(\mathbf{x})) \tag{4.3}$$

where, $p_n(.)$ is the resulting distribution of a Markov chain running for n steps. Besides that, other methods have been proposed to train RBMs (e.g. persistent contrastive divergence [**200**], fast persistent contrastive divergence [**201**], parallel tempering [**52**]), or to replace the Gibbs sampling with a transition operator for a faster mixing rate and to improve the learning accuracy without affecting computational costs [**33**].

4.2.2. Sparsity in restricted Boltzmann machines

In general and for the purposes of machine learning, obtaining a sparse version of a given model, leads to a reduction in parameters, which, in turns helps in addressing problems such as overfitting and excessive computational complexity. The sparsity issue in RBMs is so important that considerable attention is given to it in the literature. Hereafter we point to the most relevant works but will not attempt to provide a comprehensive overview. It is worth mentioning that in [**53**] and [**227**] it has been shown how the histogram of the RBM weights changes shape during the training process, going from a Gaussian shape (initially) to a shape that peaks around zero (which provides a further motivation towards sparsity enforcement). Also, in our own work [**145**], we have hinted a similar behavior, as it is depicted in Figure A.7 from Appendix A.

One of the most common methods to obtain sparse representations is by encouraging it during the training phase using different variants of a sparsity penalty function, as done for instance, in [**113**] and [**172**]. However, performing this process during the learning phase does not guarantee sparsity in the testing phase [**195**]. To overcome this limitation, Cardinality-RBM (Ca-RBM) was proposed in [**195**] to ensure sparsity in the hidden representation by introducing cardinality potentials into the RBMs energy function. Moreover, Wan et al. (2015) [**216**] have proposed Gaussian Ca-RBM, in which they replace the universal threshold for hidden units activation from Ca-RBM with adaptable thresholds. These thresholds are sampled from a certain distribution which takes into consideration the input data. Recently, Han et al. (2015) [**77**] introduced one of the most efficient methods to obtain weights sparsity in deep neural network. They successfully obtained up to 10 times less weights in deep neural networks with no

loss in accuracy. Their method assumes three simple steps: (1) the network is trained to learn the most important connections; (2) the unimportant connections are pruned; (3) the network is retrained to fine tune the weights of the remaining connections. To achieve the best performance the steps 2 and 3 have to be repeated iteratively which makes this a computationally expensive and unscalable method.

Thus, to the best of our knowledge, all of the state-of-the-art methods impose a sparsity regularization target during the learning process, which makes them impractical with large datasets having millions (or even billions) of input features. This is because the training process is excessively slow in such situations. Our solution to overcome this problem is to ensure weight sparsity from the initial design of an RBM using relevant findings from the field of network science (Section 4.3).

4.2.3. Complex networks

Complex networks (e.g. biological neural networks, actors and movies, power grids, transportation networks) are everywhere, in different forms and different fields, from neurobiology to statistical physics [**192**], and they are studied in network science. Formally, a complex network is a graph with non trivial topological features, human or nature made. The most two well-known and deeply studied types of topological features in complex networks are the scale-free and the small-world concepts, due to the fact that a wide range of real-world complex networks have these topologies. A network with a scale-free topology [**19**] is a sparse graph [**51**] that approximately has a power-law degree distribution $P(k) \sim k^{-\gamma}$, where the fraction $P(k)$ from the total nodes of the network has k connections to other nodes, and the parameter $\gamma \in (2, 3)$ usually.

At the same time, a network model with the small-world topological feature [**218**] is defined to be a graph in which the typical distance (L) between two randomly chosen nodes (the number of hops required to reach one node from the other) is very small, approximately on the logarithmic scale with respect to the total number of nodes (N) in the network, while at the same time being characterized by a clustering coefficient that is significantly higher than may appear by random chance. More formally, a graph sequence $(G_N)_{N \geq 1}$ has a small-world topology, if there is a constant $0 < K < \infty$ such that $lim_{N \to \infty} p(L_N \leq KlogN) = 1$, where L_N is the typical shortest path of G_N [**205**]. As an example, Figure 4.1 roughly illustrates a small-world topology, and a scale-free one, in two small randomly generated graphs. Both types of topologies are studied below in the context of restricted Boltzmann machines, leading to our proposal of sparse Boltzmann machine models.

4.3. Complex networks and Boltzmann machines

In this section, we firstly discuss the relation between complex networks on one side and restricted Boltzmann machines and Gaussian restricted Boltzmann machine on the other side. Secondly, we introduce an algorithm to generate sparse topologies for bipartite graphs which have both properties (i.e. scale-free and small-world) that also considers the distribution of the training data. Finally, we make

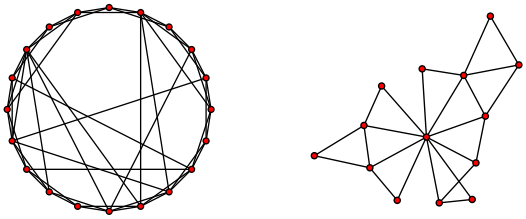


Figure 4.1 – Examples of complex networks topologies: (left) small-world; (right) scale-free.

use of the previous mentioned topology generator algorithm to present the mathematical details of two novel types of Boltzmann machines, dubbed compleX Boltzmann Machines (XBMs) and Gaussian compleX Boltzmann Machines (GXBMs).

4.3.1. Topological insight into RBMs and GRBMs

Lately, the neural networks of the human brain have started to be studied using tools from network science [**168**]. It has been found that these exhibit both a small-world topology (i.e. the shortest path between any two nodes or neurons is very small, approximately equal to the logarithm of the total number of nodes) and a scale-free topology (i.e their degree distribution follows a power law). At the same time, by making small steps towards mimicking the architecture and the functionality of the brain, deep learning methods have emerged as a promising solution in computer science to develop automated learning systems [**91**, **133**]. Here we argue that there is a clear relation between network science and deep learning. In the scope of these arguments, we introduce the following proposition:

Proposition 1. *Both restricted Boltzmann machines and Gaussian restricted Boltzmann machines are bipartite graphs which have a small-world topology.*

Proof. The diameter (i.e. the longest shortest path between any two neurons) of RBMs or GRBMs is 2, independently on the number of hidden or visible neurons, due to the fact that both models have all the possible interlayer connections, but no intralayer connections. This yields that L is bounded up by 2 for any RBM or GRBM. By replacing L in the small-world definition from Subsection 4.2.3, we obtain $lim_{N\to\infty} p(2 \leq KlogN) = 1$, which is true for any constant K, $0 < K < \infty$[1]. Similarly as RBMs and GRBMs are complete bipartite graphs, their clustering coefficient [**107**] is 1, being higher than any other possible cluster coefficient[2]. Thus, it is clear that any RBMs or GRBMs have a small-world topology. □

Following the same line, our intuition is that by preserving the small-world property of RBMs or GRBMs, while introducing also the scale-free property in their topology, we can obtain new sparse Boltzmann machines derived models which may have similar performance to RBMs or GRBMs, but with fewer free parameters (i.e. the weights between the visible and the hidden neurons). Thus,

[1]Please note that according to the definitions from [**205**], this reflects even a particular subset of small-worlds, namely ultra small-worlds.

[2]Please note that the clustering coefficient takes values between 0 and 1.

further on, we introduce the complex Boltzmann machine and the Gaussian complex Boltzmann machine (i.e. the derivatives of RBM and GRBM, respectively) which exhibit both scale-free and small-world topological properties.

4.3.2. Topology generation algorithm for XBM and GXBM

To generate a sparse topology in XBM and GXBM, we have devised a three stages heuristic method, detailed in Appendix B, Algorithm B.3. In the first stage, a scale-free bipartite graph is generated; in the second one, the graph is adjusted to be also small-world; and in the third stage, the graph topology is fitted to the data distribution. Below, this method is thoroughly discussed.

First we generate a power-law degree sequence with $n_v + n_h$ elements, using $P(k) = k^{-\gamma}$, $\forall k \in \mathbb{N}, 1 \leq k \leq n_v + n_h$, with minimum degree equal to four to favor the small-world topology, and we sort it in descending order (i.e. Algorithm B.3, lines 8-9). Each element from the sequence represents a node in the network and the actual value of that element (i.e. the degree) represents the number of connections for that specific node. After that, we split the degree sequence in two, by alternatively picking the nodes starting from those with the highest degree and proceeding until the smallest degree nodes are reached. In this way we populate two separate lists, for the hidden layer and for the visible layer, respectively (i.e. Algorithm B.3, lines 10-16). Once the list having the smallest number of elements is completed, we add all the remaining elements of the original sequence to the bigger list (i.e. Algorithm B.3, lines 17-22). In an undirected bipartite graph both layers need to have an equal number of connections. Due to the fact that the sum of the elements from one list might not be equal to the sum of the elements from the other list, we add some more degrees to the elements of the list with lower degrees (proportionally to its initial degree distribution) to equalize the two lists (i.e. Algorithm B.3, lines 23-28). Next, starting from these two lists, we create a bipartite graph G using a Havel-Hakimi procedure [**76**] (i.e. Algorithm B.3, line 29). Further on, we add few more connections to each node in G with the aim to achieve the optimal clustering coefficient as required in small-world topologies (i.e. Algorithm B.3, lines 30-49). This ensures also a dense local connectivity useful, by example, to take local neighborhoods of neurons into consideration. To clarify how this is done, note that in Algorithm B.3 (line 3[3]), when parameter σ^{neigh} is increased the nodes local neighborhoods gradually turn into larger neighborhoods (or even the graph as a whole). In turn, when parameter ϕ is increased, the neighborhoods tend to become denser. The whole algorithm proceeds iteratively until the bipartite graph meets the criteria of small-worldness (i.e. Algorithm B.3, line 51). We should mention, though, that during our whole study we observed that this property was usually achieved after just one iteration. To take the data distribution into consideration, as a final stage, the algorithm re-arranges the order of the visible nodes in G such that the nodes having more connections end up corresponding to the training data features with a higher standard deviation (i.e. Algorithm B.3, line 53). The resulting bipartite graph G can then be used

[3]In this chapter, we have varied σ^{neigh} and ϕ between 4 and 6 to favor the emergence of local medium connected neighborhoods.

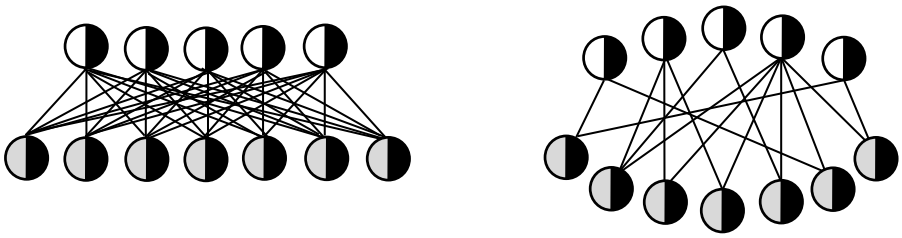


Figure 4.2 – Exemplification of RBM topology (left) and XBM topology (right). The layer with visible neurons (half gray, half black) is at the bottom of the plots, the layer with hidden neurons (half white, half black) is on the top of the plots, while in the middle we have the connections between the two layers.

as the topology of our XBM or GXBM models (i.e. Algorithm B.3, line 55-57), as detailed next.

4.3.3. CompleX Boltzmann Machines (XBMs)

Just like restricted Boltzmann machines, complex Boltzmann machines are made up by two layers of neurons, with connections only in between different layers and no connections within the same layer. The bottom layer (i.e. the visible one) is denoted further with a binary vector $\mathbf{v} = [v_1, v_2, .., v_{n_v}]$, in which each unit v_i is a binary unit, and where n_v is the size of $\mathbf{v}$ (i.e. the number of neurons of the visible layer). The top layer (i.e. the hidden one) is represented further by the binary vector $\mathbf{h} = [h_1, h_2, .., h_{n_h}]$, in which each element h_j is binary, and where n_h is the size of $\mathbf{h}$. Furthermore, each neuron from the visible layer has associated one bias. The biases of the visible neurons are collected in a vector $\mathbf{a} = [a_1, a_2, .., a_{n_v}]$. Similarly, hidden layer neurons have biases, collected in vector $\mathbf{b} = [b_1, b_2, .., b_{n_h}]$.

The difference between RBM and XBM consists in how the neurons from the different layers are connected between them. RBMs form a full undirected mesh between all the neurons on the hidden layer and all the neurons on the visible layer. By contrast, in XBMs the connections between the two layers are still undirected but sparse, as generated by Algorithm B.3. Thus, XBMs have both scale-free and small-world topological properties. These connections are defined in a sparse adjacency weights matrix $\mathbf{W} = [[w_{11}, w_{12}, ..., w_{1n_h}], .., [w_{n_v1}, w_{n_v2}, ..., w_{n_vn_h}]]$ in which the elements are either null ($w_{ij} = 0$) when there is no connection between the visible neuron i and the hidden neuron j or have a connection weight ($w_{ij} \neq 0$) when the connection between i and j exists. The high level architectures of RBMs and XBMs are depicted in Figure 4.2. The sparse topology of XBMs leads to a much smaller number of connections, which further on leads to faster computational times than RBMs.

The energy function of an XBM is defined as:

$$E(v, h) = -\sum_{i=1}^{n_v} \sum_{j \in \mathbf{\Gamma}_i^h} v_i h_j w_{ij} - \sum_{i=1}^{n_v} v_i a_i - \sum_{j=1}^{n_h} h_j b_j \tag{4.4}$$

where, $\mathbf{\Gamma}_i^h$ is the set of all hidden neurons connected to the visible neuron i (i.e. $\mathbf{\Gamma}_i^h = \{j | 1 \leq j \leq n_h, \forall j \in \mathbb{N} \wedge w_{ij} \neq 0\}$).

Due to the fact that there are no links between the neurons on the same layer, inference in XBM can be performed in parallel for any visible neuron i or any hidden neuron j, as below:

$$p(h_j = 1|\mathbf{v}, \mathbf{\Theta}) = \mathcal{S}\bigg(b_j + \sum_{i \in \mathbf{\Gamma}_j^v} v_i w_{ij}\bigg) \tag{4.5}$$

$$p(v_i = 1|\mathbf{h}, \mathbf{\Theta}) = \mathcal{S}\bigg(a_i + \sum_{j \in \mathbf{\Gamma}_i^h} h_j w_{ij}\bigg) \tag{4.6}$$

where, $\mathbf{\Gamma}_j^v$ is the set of all visible neurons connected to the hidden neuron j (i.e. $\mathbf{\Gamma}_j^v = \{i|1 \leq i \leq n_v, \forall i \in \mathbb{N} \wedge w_{ij} \neq 0\}$), $\mathcal{S}(\cdot)$ is the sigmoid function, and $\mathbf{\Theta}$ represents the free parameters of the model (i.e. $\mathbf{W}$, $\mathbf{a}$, $\mathbf{b}$).

The general update rule for the free parameters $\mathbf{\Theta}$ of the XBM model is given by:

$$\Delta\mathbf{\Theta}_{\tau+1} = \rho\Delta\mathbf{\Theta}_{\tau} + \alpha(\nabla\mathbf{\Theta}_{\tau+1} - \xi\mathbf{\Theta}_{\tau}) \tag{4.7}$$

where τ, ρ, α, and ξ represent the update number, momentum, learning rate, and weights decay, respectively. For a thorough discussion on the optimal choice of these parameters the interested reader is referred to [**83**]. Furthermore, $\nabla\mathbf{\Theta}_{\tau+1}$ for each of the free parameters can be computed by using contrastive divergence [**81**] and deriving the energy function from Equation 4.4 with respect to that parameter, yielding:

$$\nabla w_{ij} \propto \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_n; \forall i \in \mathbb{N}, 1 \leq i \leq n_v, \forall j \in \mathbf{\Gamma}_i^h; \tag{4.8}$$

$$\nabla a_i \propto \langle v_i \rangle_0 - \langle v_i \rangle_n; \forall i \in \mathbb{N}, 1 \leq i \leq n_v; \tag{4.9}$$

$$\nabla b_j \propto \langle h_j \rangle_0 - \langle h_j \rangle_n; \forall j \in \mathbb{N}, 1 \leq j \leq n_h; \tag{4.10}$$

with $\langle \cdot \rangle_n$ being the distribution of the model obtained after n steps of Gibbs sampling in a Markov Chain which starts from the original data distribution $\langle \cdot \rangle_0$. We must note that in this chapter we have chosen to train our proposed models using the original contrastive divergence method [**81**], which is widely used and allows for a direct comparison to the results reported in the literature. It may well be that other training methods would offer better performance; yet, overall, we do not expect particular training methods to significantly affect our findings.

4.3.4. Gaussian compleX Boltzmann Machines (GXBMs)

Just like in GRBMs and RBMs, the only differences between GXBMs and XBMs is that in the case of GXBMs the visible layer $\mathbf{v} = [v_1, v_2, .., v_{n_v}]$ has real values and each v_i is a linear unit with Gaussian noise [**80**]. Thus, the total energy equation of GXBMs is slightly changed to reflect the real visible layer, as follows:

$$E(v, h) = -\sum_{i=1}^{n_v} \sum_{j \in \mathbf{\Gamma}_i^h} \frac{v_i}{\sigma_i} h_j w_{ij} - \sum_{i=1}^{n_v} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n_h} h_j b_j \tag{4.11}$$

where, σ_i represents the standard deviation of the visible neuron i. We note that in the remainder we adopt the same notations used in XBM modeling, unless specified otherwise.

Furthermore, the inference in GXBMs can be performed in parallel for any visible neuron i or any hidden neuron j, as below:

$$p(h_j = 1|\mathbf{v}, \mathbf{\Theta}) = \mathcal{S}\left(b_j + \sum_{i \in \mathbf{\Gamma}_j^v} \frac{v_i}{\sigma_i} w_{ij}\right) \tag{4.12}$$

$$p(v_i = x|\mathbf{h}, \mathbf{\Theta}) = \mathcal{N}\left(a_i + \sum_{j \in \mathbf{\Gamma}_i^h} h_j w_{ij}, \sigma_i^2\right) \tag{4.13}$$

where, $\mathcal{N}(\cdot, \cdot)$ represents a Gaussian distribution. Finally, the learning in GXBM can be done using the same procedure as for XBM (Section 4.3.3).

4.4. Experimental results

4.4.1. Evaluation method

To assess the performance of XBM and GXBM we have conducted three sets of experiments in a step-wise fashion. In the first one, we study the behavior of XBM, GXBM and their topology generation algorithm. In the second one, we analyze the reconstruction error obtained by GXBM on random generated data and on a real world dataset, more exactly on the Geographical Origin of Music dataset [**229**]. Thirdly, we assess the statistical performance of XBM on the MNIST digits dataset, CalTech 101 Silhouettes dataset [**128**], and the 8 datasets from UCI evaluation suite [**105**] using Annealed Importance Sampling (AIS) [**177**].

Furthermore, in the last two sets of experiments, we compare GXBM/XBM against three other methods, as follows:

(1) the standard fully connected GRBM/RBM;
(2) sparse GRBM/RBM models, denoted with $\text{GRBM}_{\text{FixProb}}$ (Fixed Probability) / $\text{RBM}_{\text{FixProb}}$, in which the probability for any possible connection to exist is set to the number of weights of the counterpart GXBM/XBM model divided by the total number of possible connection for that specific configuration of hidden and visible neurons[4];
(3) sparse GRBM/RBM models, denoted with $\text{GRBM}_{\text{TrPrTr}}$ (Train Prune Train) / $\text{RBM}_{\text{TrPrTr}}$, in which the sparsity is obtained using the algorithm introduced in [**77**] with L2 regularization, and in which the weights sparsity target is set to the number of weights of the counterpart GXBM/XBM model. Please note that in all experiments if the weights sparsity target was not reached after 50 pruning iterations, we stopped the training algorithm and we used the obtained $\text{GRBM}_{\text{TrPrTr}}$/$\text{RBM}_{\text{TrPrTr}}$ model.

For each evaluated case, we analyze two scenarios when: (1) the number of connections is the same for the sparse and the full connected models, while the number of hidden neurons is different; (2) the number of hidden neurons is the same for the sparse and the full connected models, while the number of connections is different.

[4]Please note that this procedure yields approximately the same number of connections in $\text{GRBM}_{\text{FixProb}}$/$\text{RBM}_{\text{FixProb}}$ as in GXBM/XBM to ensure a fair comparison.

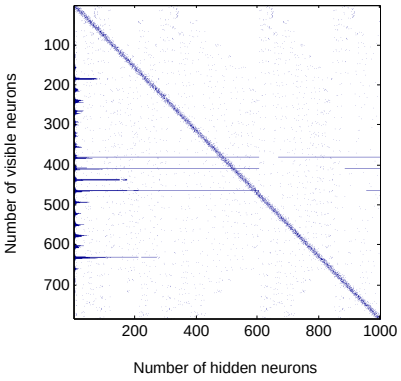
4.4.2. Scrutinizing XBM and GXBM topologies

In this set of experiments, we analyze the characteristics of the XBM sparse topology. For the sake of brevity, we refer just to the XBM - RBM relation, since the GXBM - GRBM is identical from the topological point of view. To perform the various operations on the bipartite graph we have used the NetworkX library [**75**], setting the power law coefficient γ to 2, a typical value in various real-world networks [**41**]. Firstly, we verified the output of the topology-generation algorithm (i.e. Algorithm B.3). Being one of the algorithm constraints, the small-world property is preserved in XBM. At the same time, due to the fact that the neurons from different layers are connected starting from a power law degree sequence, the scale-free property is also preserved. As example, Figure 4.3a depicts the weights distribution for an XBM having 784 visible and 1000 hidden neurons, while Figure 4.3b shows the degree distribution of an XBM with 100 visible and 1000 hidden neurons on the loglog scale. Figure 4.3b exhibits evidently a scale-free degree distribution. All other experiments exhibited the required scale-free distribution. Furthermore, we analyzed the number of connections in XBM in comparison with the number of connections in RBMs, given the same number of hidden (n_h) and visible (n_v) neurons. Figure 4.3c depicts how many times the number of connections in RBM is bigger than the number of connections in XBM for various configurations (the number of hidden and visible neurons varies from 10 to 1000). The actual values in the heat map are computed using the following formula n_w^{RBM}/n_w^{XBM}, where n_w^{XBM} is obtained after counting the links given by the topology generation algorithm for XBM, and $n_w^{RBM} = n_v n_h$. It can be observed that as the number of hidden and visible neurons increases, the number of weights in XBM becomes smaller and smaller than the one in RBM. For instance, we achieve around 14 times less weights in XBM for 100 visible and 100 hidden neurons, and approximatively 95 times less weights in XBM for 1000 visible and 1000 hidden neurons.

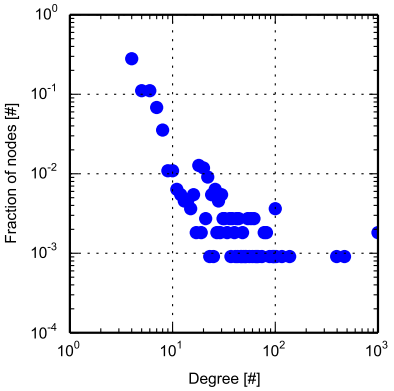
4.4.3. GXBM evaluation

In the second set of experiments, we assess the performance of GXBM against GRBM, $\text{GRBM}_{\text{FixProb}}$, and $\text{GRBM}_{\text{TrPrTr}}$ on randomly generated as well as on the real-world dataset Geographical Origin of Music [**229**].

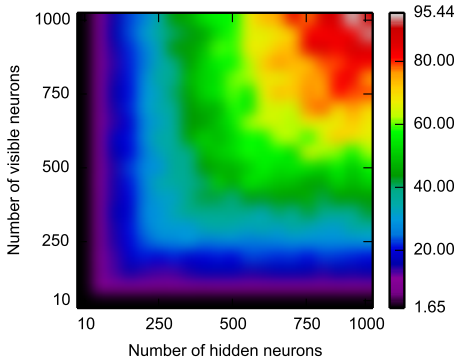
Settings and implementations. For all experiments performed in this set we have used Python implementations of the four models under scrutiny. In all models, the momentum was set to 0.5, the learning rate to 0.001, and the number of Gibbs sampling in contrastive divergence to 1, as discussed in [**83**]. The weights decay was 0.0002 for GXBM, GRBM, and $\text{GRBM}_{\text{FixProb}}$, while for $\text{GRBM}_{\text{TrPrTr}}$ we used the L2 regularization. The number of neurons in the visible layer was set to the dimension of the input data, while the number of hidden neurons was varied for a better comparison. In the learning phase, we stopped the models after 100 training epochs to ensure full convergence. In fact, convergence was much faster, as exemplified in Figure 4.3d which shows the case of GXBM with 100 visible and 1000 hidden neurons trained on random generated data. In the case of $\text{GRBM}_{\text{TrPrTr}}$, we have repeated the training procedure for a maximum of 50



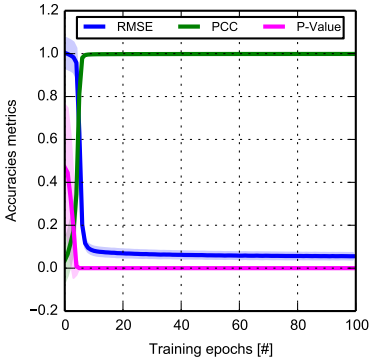
(a) Example of weight distribution for an XBM with 784 visible and 1000 hidden neurons.



(b) The degree distribution of an XBM with 1000 hidden neurons and 100 visible ones (loglog scale).



(c) Studying the relation between the number of weights in RBM and XBM (the heatmap values are given by n_w^{RBM}/n_w^{XBM}).



(d) Training behavior on random data for an GXBM with 1000 hidden neurons and 100 visible ones.

Figure 4.3 – Scrutinizing XBM and GXBM topologies and learning behavior.

pruning iterations trying to reach the same amount of weights as in GXBM, but this target was impossible to reach in all situations.

Performance metrics. To quantify the performance of the models, we used a variety of standard metrics. We have used: the Root Mean Square Error (RMSE) to estimate the distance between the reconstructed inputs and the ground truth; the Pearson Correlation Coefficient (PCC) to reflect the correlations between the estimated inputs and the ground truth; and the P-value to arrive at a statistically significant reconstruction level during the learning phase.

4.4.3.1. *GXBM performance on random generated data.* Firstly, we analyze how well GXBM is capable to reconstruct random generated data. To this end, we have generated 1000 data points, each one having 100 dimensions, and each dimension being sampled from a Gaussian distribution with 0 mean and standard deviation equal to 1, i.e. $\mathcal{N}(0, 1)$. Due to the fact that these are random generated data, there was no reason to use cross validation. Thus, we have used 70% of

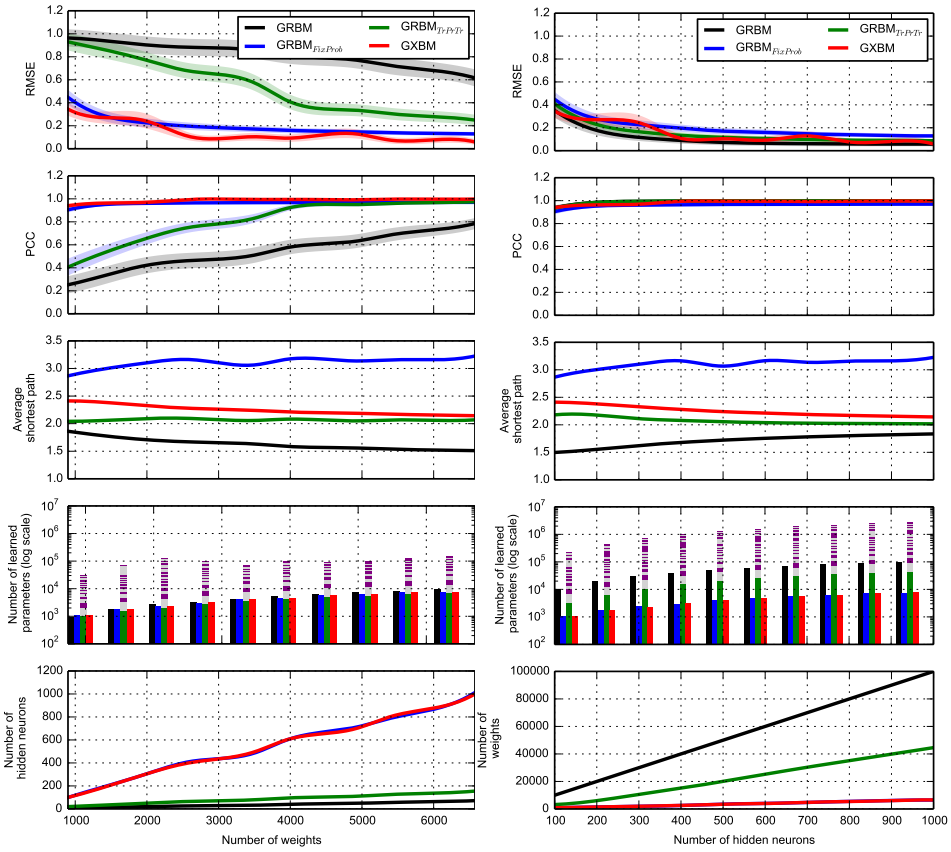


Figure 4.4 – Reconstruction capability on random generated data of GRBM, $\text{GRBM}_{\text{FixProb}}$, $\text{GRBM}_{\text{TrPrTr}}$ and GXBM with: (left) the same number of weights; (right) the same number of hidden neurons. The straight line represents the mean; the shadowed area shows the standard deviation. The number of learned parameters for $\text{GRBM}_{\text{TrPrTr}}$ is given in green after the last pruning iteration, while above the green color the alternating gray and purple colors represent the number of learned parameters at each pruning iteration starting with the first one from the top.

data to train the models, and the remaining 30% to test the models. Firstly, we analyzed the reconstruction capabilities of GXBM, $\text{GRBM}_{\text{FixProb}}$, $\text{GRBM}_{\text{TrPrTr}}$, and GRBM, given the the same number of weights. Figure 4.4 (left) depicts this situation, while the number of weights were varied from 700 up to approximately 7000. Clearly, GXBM outperforms GRBM in both, RMSE and PCC, while its internal topology permits it to have a higher number of hidden neurons. Remarkably, with approximately 1000 weights, the mean RMSE for GXBM is already very low, around 0.3, while the mean PCC is almost perfect, over 0.95. By contrast, GRBM with 1000 weights performed poorly. Furthermore, it is clear that the GRBM performance increases with the number of weights, yet it can be observed that even at approximately 7000 weights GRBM is not capable to reach the same level of performance of the GXBM with 1000 weights. Besides that, GXBM outperforms also the other sparse models, $\text{GRBM}_{\text{FixProb}}$ and $\text{GRBM}_{\text{TrPrTr}}$, but

not so drastically. In fact, $\text{GRBM}_{\text{FixProb}}$ has very close performance to GXBM. $\text{GRBM}_{\text{TrPrTr}}$ is not so close, while having also a very high computational cost as depicted in the fourth row of Figure 4.4.

To better understand these differences, we proceed to the next scenario, analyzing GRBM, $\text{GRBM}_{\text{FixProb}}$, $\text{GRBM}_{\text{TrPrTr}}$, and GXBM having the same number of hidden neurons. This is reflected in Figure 4.4 (right) in which the number of hidden neurons is varied from 100 to 1000. Surprising, even though the number of free parameters (i.e. weights) was smaller by at least one order of magnitude in GXBM (as it can be seen in the bottom-right plot of Figure 4.4), GXBM performs similarly to GRBM in terms of PCC and RMSE, while $\text{GRBM}_{\text{FixProb}}$ and $\text{GRBM}_{\text{TrPrTr}}$ reach almost a similar performance. Still, $\text{GRBM}_{\text{TrPrTr}}$ has the downside of not being capable to reach the same number of weights as GXBM or $\text{GRBM}_{\text{FiXProb}}$ even after 50 prunning iterations, as reflected on the fourth and fifth rows of Figure 4.4. Interestingly, for this specific dataset, all models seem to reach their maximum learning capacity when they have approximately 400 hidden neurons, showing no further improvement after this point.

4.4.3.2. *GXBM performance on geographical ethnomusicology data.* We have then assessed the reconstruction capabilities of GXBM on a real world dataset. We have used the Geographical Origin of Music dataset [**229**]. This contains 1059 tracks from different countries, each track having 70 dimensions (i.e. the first 68 represent audio features, while the last ones are latitude and longitude of each specific song), already normalized to have mean 0 and standard deviation equal to 1. We performed a 10-fold cross validation. The averaged results depicted in Figure 4.5 confirm the same findings obtained on the random generated dataset (Section 4.4.3.1). Given the same number of weights, GXBM outperforms clearly GRBM, and outperforms slightly $\text{GRBM}_{\text{FixProb}}$ and $\text{GRBM}_{\text{TrPrTr}}$, while the four perform similarly when the number of hidden neurons is comparable. By analyzing the reconstruction performance metrics (i.e. RMSE and PCC) reported in Figure 4.5 and Figure 4.4 it is interesting to observe that GXBM and $\text{GRBM}_{\text{FixProb}}$ are more stable, independently if the data are random or non-random. By contrast, GRBM and $\text{GRBM}_{\text{TrPrTr}}$ performance depend more on the data type.

To assess GXBM from a different perspective, we performed a small regression experiment on the geographical ethnomusicology dataset, even though the regression task does not constitute one of the goals of this chapter. Thus, we compared the ability of GRBM and GXBM in predicting the latitude and the longitude corresponding to the 68 audio features across all the tracks. We can see from Figure 4.5 that the dataset is well represented with just about 400 hidden neurons; thus we have used this value in both models. In addition, in GXBM we have set the visible neurons corresponding to latitude and longitude to the first two visible neurons having the largest number of connections. We then performed a 10-fold cross validation to obtain the average distance error between the predicted latitude and longitude and their true value counterparts. The resulting predictions were 3258 ± 175 kilometers (GRBM) and 3252 ± 176 kilometers (GXBM) which are comparable to the top performers found in [**229**], even if it is well known that GRBMs and RBMs are not best performers on classification and regression tasks, when used as standalone models - best performance would pursued by stacking these models in deep architectures [**20**].

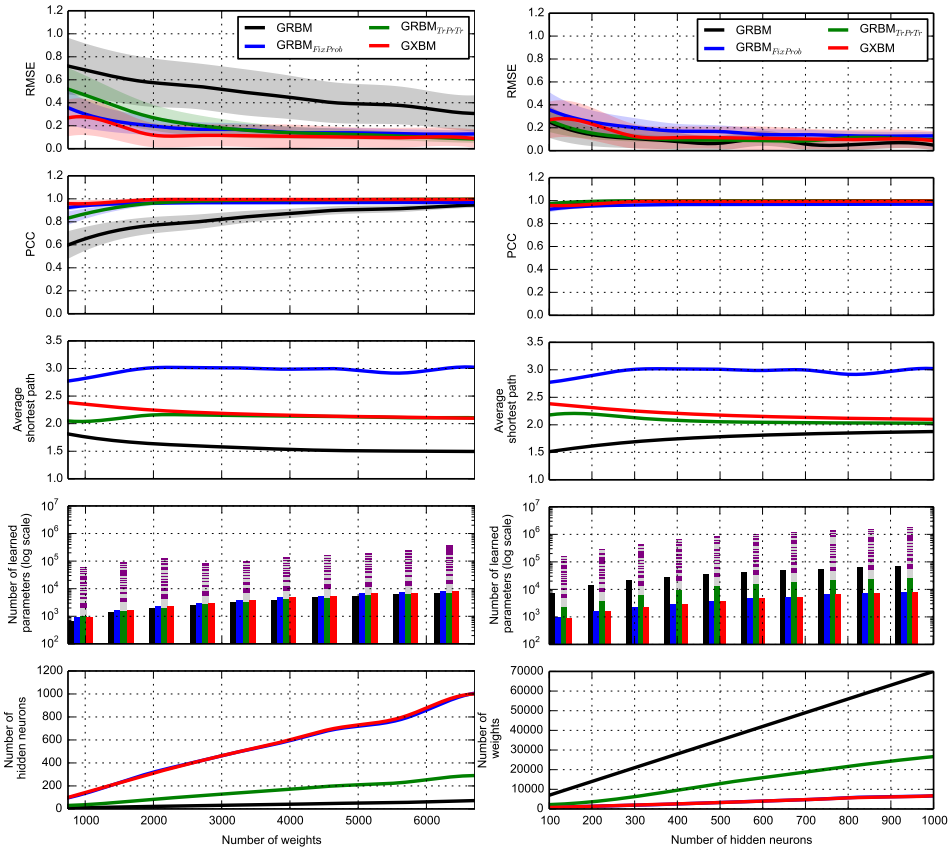


Figure 4.5 – Reconstruction capabilities on the geographical ethnomusicology dataset of GRBM, $\text{GRBM}_{\text{FixProb}}$, $\text{GRBM}_{\text{TrPrTr}}$, and GXBM with: (left) the same number of weights; (right) the same number of hidden neurons. The straight line represents the mean; the shadowed area shows the standard deviation. The number of learned parameters for $\text{GRBM}_{\text{TrPrTr}}$ is given in green after the last pruning iteration, while above the green color the alternating gray and purple colors represent the number of learned parameters at each pruning iteration starting with the first one from the top.

From the third row of Figures 4.4 and 4.5, it is very interesting to observe that, even if it was not a target, the random generated connections of $\text{GRBM}_{\text{FixProb}}$ exhibit a very small average shortest path. This observation may explain the good performance of $\text{GRBM}_{\text{FixProb}}$ close to the one of GXBM in this set of experiments, while having a similar computational time (i.e. the same number of weights to be computed). Even more, the $\text{GRBM}_{\text{TrPrTr}}$ models end up still having a small-world topology after the iterative pruning process. Still, the better performance obtained by GXBM is given by, all-together, its scale-free topology, the local neighborhoods connections, and the consideration of data distribution in the topology.

4.4.4. XBM evaluation

In the third set of experiments, we have assessed the performance of XBM on the MNIST digits dataset[5], CalTech 101 Silhouettes dataset [**128**], and the 8 datasets from the UCI evaluation suite [**105**].

Settings, implementations, and performance metrics. To allow for a direct comparison, we have adopted the same settings for all datasets. This time, all the models were implemented and the experiments were performed using the MATLAB® environment, partially to facilitate the comparisons with the results reported on the MNIST dataset in [**177**]. For each RBM model, we have trained two XBM, two $\text{RBM}_{\text{FixProb}}$, and two $\text{RBM}_{\text{TrPrTr}}$ models, as follows: (1) one having the same number of hidden neurons, but with much fewer weights; (2) the other with approximatively the same number of weights but with a higher number of hidden neurons. Just as in [**177**], we have used a fixed learning rate (i.e. 0.05) for all datasets, with the exception of the MNIST dataset when a decreasing learning rate was used for the situation in which the number of contrastive divergence steps was gradually increased from 1 up to 25, as suggested in [**35**]. In all cases, the number of training epochs was set to 259 and the training and testing data were split in mini-batches of 100 samples. Finally, to assess the performance of XBM, $\text{RBM}_{\text{FixProb}}$, and $\text{RBM}_{\text{TrPrTr}}$, we have used AIS (adopting the very same settings as in [**177**]) to estimate the average log-probabilities on the training and testing data of the datasets.

4.4.4.1. *XBM performance on the MNIST digits dataset.* The MNIST digits dataset is widely used to assess the performance of novel machine learning algorithms. The dataset contains 60000 images for training and 10000 images for testing, each image being a variant of a digit represented by a 28x28 binary matrix.

The results depicted in Table 4.1[6] confirm the findings from the previous set of experiments (i.e. GXBM - GRBM case). We see that: (1) given the same number of hidden neurons an XBM model has a generative performance close to an RBM, yet having the significant benefit of much fewer weights (i.e. this offers much smaller computational time); (2) given approximatively the same number of weights (i.e. at comparable computational time) an XBM model has a better generative performance than an RBM model. The results suggest that XBM models can achieve good generative performance already with one step contrastive divergence and a fix learning rate. This is because the performance gain obtained by increasing the number of contrastive divergence steps is smaller than in the case of RBMs. This may be considered also an advantage for XBMs as it is a common practice to use just one step contrastive divergence to decrease the learning time. It is worth highlighting, that an XBM with 15702 weights reaches 36.02 *nats* better than an RBM with 15680 weights and 20 hidden neurons. Similarly to the GXBM behaviour, as the models increase in size, we observe that the difference between XBMs and RBMs gets smaller. For instance, an XBM with 387955 weights is 10.89 *nats* better than an RBM with 392000 weights, which, further on, is just 6.99 *nats* better than an XBM having the same number of hidden neurons (i.e. 500) but with approximately 40 times fewer weights (i.e. 10790). Also worth noting that

[5] http://yann.lecun.com/exdb/mnist/, Last visit on October 18^{th} 2015.

[6] The average cluster coefficient was computed using the method proposed for bipartite graphs in [**107**].

Table 4.1 – Estimation of the average log-probabilities (log-prob) on the training and testing data obtained from the MNIST digits dataset using AIS. The results for RBMs are taken from [**177**].

No. of CD steps during learning	No. of weights	Model	No. of hidden units	Average shortest path	Average cluster coefficient	No. of pruning iterations	Average train log-prob	Average test log-prob
1 (fixed)	15680	RBM	20	1.94	1	0	-164.87	-164.50
	15702	XBM	1000	2.70	0.089	0	-130.26	**-128.48**
	15730	$RBM_{FixProb}$	1000	3.19	0.034	0	-143.32	-142.34
	65583	RBM_{TrPrTr}	1000	2.45	0.060	50	-146.51	-147.47
	4954	XBM	20	2.01	0.396	0	-167.36	-166.60
	4896	$RBM_{FixProb}$	20	2.25	0.209	0	-169.55	-169.10
	6119	RBM_{TrPrTr}	20	2.22	0.276	50	-176.50	-176.20
	19600	RBM	25	1.93	1		-153.46	-152.68
	19527	XBM	1500	2.70	0.071	0	-126.74	**-126.07**
	19875	$RBM_{FixProb}$	1500	3.28	0.037	0	-143.83	-142.91
	103395	RBM_{TrPrTr}	1500	2.38	0.059	50	-148.49	-151.08
	6358	XBM	25	2.01	0.350	0	-163.39	-161.09
	6389	$RBM_{FixProb}$	25	2.12	0.205	0	-162.66	-162.02
	6593	RBM_{TrPrTr}	25	2.25	0.234	23	-170.29	-169.68
	392000	RBM	500	1.52	1	0	-122.86	-125.53
	387955	XBM	27000	2.05	0.156	0	-115.28	**-114.64**
	391170	$RBM_{FixProb}$	27000	2.87	0.053	0	-140.97	-140.30
	2204393	RBM_{TrPrTr}	27000	2.10	0.071	50	-602.94	-652.21
	10790	XBM	500	2.44	0.082	0	-134.06	-132.52
	10846	$RBM_{FixProb}$	500	3.12	0.039	0	-149.27	-148.42
	29616	RBM_{TrPrTr}	500	2.62	0.064	50	-131.48	-131.34
3 (fixed)	19600	RBM	25	1.93	1	0	-144.11	-143.20
	19527	XBM	1500	2.70	0.071	0	-122.19	-121.59
	19875	$RBM_{FixProb}$	1500	3.28	0.037	0	-139.89	-138.86
	142237	RBM_{TrPrTr}	1500	2.34	0.074	50	-116.11	**-120.50**
	6358	XBM	25	2.01	0.350	0	-158.67	-157.69
	6389	$RBM_{FixProb}$	25	2.12	0.205	0	-161.35	-160.69
	7715	RBM_{TrPrTr}	25	2.14	0.259	50	-159.05	-158.39
	392000	RBM	500	1.52	1	0	-102.81	-105.50
	387955	XBM	27000	2.05	0.156	0	-103.66	**-101.93**
	391170	$RBM_{FixProb}$	27000	2.87	0.053	0	-125.51	-125.03
	2827787	RBM_{TrPrTr}	27000	2.05	0.087	5	-488.07	-512.56
	10790	XBM	500	2.44	0.082	0	-128.07	-127.41
	10846	$RBM_{FixProb}$	500	3.12	0.039	0	-145.11	-144.08
	38991	RBM_{TrPrTr}	500	2.46	0.069	50	-112.27	-112.45
from 1 to 25 (variable)	392000	RBM	500	1.52	1	0	-83.10	-86.34
	387955	XBM	27000	2.05	0.156	0	-86.12	**-85.21**
	391170	$RBM_{FixProb}$	27000	2.87	0.053	0	-107.23	-106.78
	3262957	RBM_{TrPrTr}	27000	2.18	0.076	50	-349.87	-376.92
	10790	XBM	500	2.44	0.082	0	-121.26	-120.43
	10846	$RBM_{FixProb}$	500	3.12	0.039	0	-136.27	-135.89
	36674	RBM_{TrPrTr}	500	2.35	0.071	50	-134.25	-135.76

when the CD learning steps were gradually increased from 1 to 25 the XBM model with 387955 weights slightly outperformed (i.e. 1.13 *nats*) the RBM model having 500 hidden neurons and 392000 weights trained in [**177**]. We should note that the latter is considered to be one of the best generative RBM models reported in the literature, to the best of our knowledge.

Finally, we would like to highlight that in 5 out of the 6 cases considered, XBMs outperform all the other models including the sparse ones, while in the remaining case, the best performer is not the fully connected RBM as expected, but still a sparse model,i.e. RBM_{TrPrTr}. Yet, the latter one, same as in Subsection 4.4.3, shows some robustness issues as it performs very badly for a large number of hidden neurons (i.e. 27000), while its computational time is much higher than for all the other models considered. Remarkably, $RBM_{FixProb}$ obtains very good results in

Table 4.2 – Estimation of the average log-probabilities (log-prob) on the training and testing data obtained from the CalTech 101 Silhouettes dataset using 1 step CD and AIS.

Dataset	No. of weights	Model	No. of hidden units	Average shortest path	Average cluster coefficient	No. of pruning iterations	Average train log-prob	Average test log-prob
	19600	RBM	25	1.93	1	0	-329.74	-315.22
	19201	XBM	1500	2.91	0.099	0	-146.97	**-142.96**
	19979	$RBM_{FixProb}$	1500	3.28	0.037	0	-164.84	-162.41
	414478	RBM_{TrPrTr}	1500	2.13	0.223	50	-141.77	-360.24
	6423	XBM	25	2.02	0.47	0	-330.37	-323.01
	6341	$RBM_{FixProb}$	25	2.14	0.20	0	-356.68	-353.25
28x28	6531	RBM_{TrPrTr}	25	2.04	0.316	13	-350.66	-339.94
image size	392000	RBM	500	1.52	1	0	-161.05	-261.44
	392464	XBM	27000	2.05	0.136	0	-178.16	-187.49
	381228	$RBM_{FixProb}$	27000	3.51	0.040	0	-277.06	-283.15
	390410	RBM_{TrPrTr}	27000	3.02	0.017	13	-145.38	-307.14
	11077	XBM	500	2.46	0.094	0	-169.56	**-164.07**
	11350	$RBM_{FixProb}$	500	3.09	0.038	0	-196.24	-191.71
	34479	RBM_{TrPrTr}	500	2.15	0.212	50	-240.09	-403.63
	6400	RBM	25	1.83	1	0	-102.87	-94.88
	6359	XBM	500	2.40	0.095	0	-74.60	**-69.95**
	6364	$RBM_{FixProb}$	500	2.93	0.048	0	-94.68	-89.93
	44573	RBM_{TrPrTr}	500	2.13	0.211	50	-121.19	-166.37
	2296	XBM	25	2.04	0.400	0	-99.78	-93.64
	2321	$RBM_{FixProb}$	25	2.09	0.226	0	-100.18	-92.89
16x16	2084	RBM_{TrPrTr}	25	2.03	0.350	3	-114.54	-106.69
image size	128000	RBM	500	1.54	1	0	-70.89	-98.64
	123580	XBM	10000	2.04	0.191	0	-77.96	-78.43
	122841	$RBM_{FixProb}$	10000	3.09	0.055	0	-104.06	-102.90
	609307	RBM_{TrPrTr}	10000	2.09	0.114	50	-102.48	-101.16
	6721	XBM	500	2.70	0.147	0	-73.83	**-69.29**
	6407	$RBM_{FixProb}$	500	2.92	0.048	0	-83.37	-78.48
	44573	RBM_{TrPrTr}	500	2.13	0.211	50	-121.19	-166.37

all cases, being behind XBM just from few up to a maximum of approximately 25 *nats* in the worst case.

4.4.4.2. *XBM performance on the CalTech 101 Silhouettes dataset.* To confirm the previous results on a different (more complicated) dataset, further on we assess XBMs generative performance on the CalTech 101 Silhouettes dataset [**128**]. This dataset contains silhouettes of objects extracted from the CalTech 101 image dataset. In total it has 101 classes and two datasets. One with binary images of 28x28 pixels split in a training set of 4100 samples and a testing set of 2307 samples, and one with binary images of 16x16 pixels split in a training set of 4082 samples and a testing set of 2302 samples. As our goal was not to fine tune the four models, but to have a clear direct comparison between them, for each dataset we have used 1 CD step and we considered two evaluation cases, a small RBM (i.e. 25 hidden neurons) and a large one (i.e. 500 hidden neurons). Table 4.2 confirms our previous findings and shows that XBMs are still the best performers outperforming clearly all the other models, with a striking difference of 118.48 *nats* against fully connected RBMs (which are subject to over-fitting) on the dataset of 28x28 image size. On both datasets, it is interesting to see that the best XBMs performers are not the largest models (i.e. 27000 and 10000 hidden neurons, respectively), but the average size ones (i.e. 1500 and 500 hidden neurons, respectively).

4.4.4.3. *XBM performance on the UCI evaluation suite.* Up to now all the datasets used to evaluate the XBMs were binarized images. In this last subset of experiments, we use the UCI evaluation suite, which contains 8 binary datasets

Table 4.3 – The characteristics of the UCI evaluation suite datasets.

Dataset	No. of inputs	Training set size	Testing set size
Adult	123	5000	26147
Connect4	126	16000	47557
DNA	180	1400	1186
Mushrooms	112	2000	5624
NIPS-0-12	500	400	1240
OCR-letters	128	32152	10000
RCV1	150	40000	150000
Web	300	14000	32561

coming from various domains. These datasets are carefully selected by [**105**] to assess the performance of generative and density estimation models. Their characteristics are well described in [**72**] and are summarized in Table 4.3. As before, to have a clear direct comparisons of all sparse models we have compared them with baseline fully connected RBMs with 100 hidden neurons using standard 1 CD step. Table 4.4 summarizes the results. XBMs outperform all the other models, including the fully connected RBMs, on 7 out of the 8 datasets. As usual, $\text{RBM}_{\text{FixProb}}$ shows a good performance overall, being even the best performer on one dataset, i.e. Connect4. By contrast, $\text{RBM}_{\text{TrPrTr}}$ has robustness issues, sometimes showing good generative capabilities and sometimes not. Even if it was not in our goal to outperform the best results from the literature and, as a consequence, we did not fine tune any parameter and we did not try other training algorithms, XBMs reach on all datasets very good performances close to the ones of the best generative models carefully optimized in [**72**].

To summarize this set of experiments performed on 10 binary datasets (i.e. MNIST digits, CalTech 101 Silhouettes, and UCI evaluation suite), we report that XBMs outperform all the other models in 16 out of 18 cases considered. In the other two cases, the winner is once $\text{RBM}_{\text{FixProb}}$, and once $\text{RBM}_{\text{TrPrTr}}$. Besides that, a very interesting finding is that in all cases $\text{RBM}_{\text{TrPrTr}}$ models end up having a small-world topology, as reflected by their average shortest path and cluster coefficient. Similarly, $\text{RBM}_{\text{FixProb}}$ models reach a very small average shortest path (suitable to be qualified as small-worlds), but we can not consider them to be purely small-worlds as their average cluster coefficient represents the one obtained by random chance. Still, the better overall performance obtained by XBMs may be explain by the fact that its small-world topology is supplemented by its other designed topological features, i.e. scale-free property, the consideration of local neighborhoods and data distribution. As reflected by experiments (including the ones with real-valued data for GXBMs) these features complements each other, while helping XBMs to model well very different data types.

4.5. Conclusion

In this chapter we look at the deep learning basic building blocks from a topological perspective, bringing insights from network science. Firstly, we point out that RBMs and GRBMs are small-world bipartite networks. Secondly, by introducing scale-free constraints in RBMs and GRBMs, while still considering some local neighborhoods of visible neurons, and fitting the most connected visible neurons

Table 4.4 – Estimation of the average log-probabilities (log-prob) on the training and testing data obtained from the UCI evaluation suite datasets using using 1 step CD and AIS.

Dataset	No. of weights	Model	No. of hidden units	Average shortest path	Average cluster coefficient	No. of pruning iterations	Average train log-prob	Average test log-prob
Adult	12300	RBM	100	1.49	1	0	-17.56	-17.86
	12911	XBM	1200	2.35	0.154	0	-15.51	**-15.89**
	12692	$RBM_{FixProb}$	1200	2.80	0.072	0	-17.31	-17.56
	11211	RBM_{TrPrTr}	1200	2.56	0.071	4	-135.29	-135.98
	1617	XBM	100	2.36	0.129	0	-17.92	-17.97
	1641	$RBM_{FixProb}$	100	2.50	0.082	0	-18.95	-19.04
	2089	RBM_{TrPrTr}	100	2.28	0.348	2	-100.20	-100.40
Connect4	12600	RBM	100	1.49	1	0	-16.80	-17.00
	12481	XBM	1200	2.14	0.142	0	-17.27	-17.37
	12498	$RBM_{FixProb}$	1200	2.83	0.072	0	-15.13	**-15.23**
	51412	RBM_{TrPrTr}	1200	2.10	0.211	50	-17.63	-18.11
	1692	XBM	100	2.36	0.164	0	-25.63	-25.68
	1722	$RBM_{FixProb}$	100	2.48	0.083	0	-32.01	-32.03
	1922	RBM_{TrPrTr}	100	2.43	0.188	6	-41.43	-41.52
DNA	18000	RBM	100	1.53	1	0	-94.75	-99.52
	17801	XBM	1600	2.71	0.157	0	-79.05	**-83.17**
	18314	$RBM_{FixProb}$	1600	2.93	0.060	0	-78.57	-85.53
	17597	RBM_{TrPrTr}	1600	3.26	0.087	13	-143.77	-155.75
	2267	XBM	100	2.41	0.133	0	-89.31	-90.31
	2291	$RBM_{FixProb}$	100	2.51	0.079	0	-91.53	-92.98
	2231	RBM_{TrPrTr}	100	2.44	0.177	4	-111.45	-114.17
Mushrooms	11200	RBM	100	1.49	1	0	-24.77	-25.60
	10830	XBM	1000	2.14	0.156	0	-14.21	**-14.71**
	10639	$RBM_{FixProb}$	1000	2.73	0.075	0	-15.29	-15.82
	22376	RBM_{TrPrTr}	1000	2.26	2.26	50	-21.76	-23.09
	1515	XBM	100	2.39	0.11	0	-17.14	-17.54
	1451	$RBM_{FixProb}$	100	2.54	0.083	0	-19.97	-20.21
	2017	RBM_{TrPrTr}	100	2.35	0.155	31	-21.52	-22.05
NIPS-0-12	50000	RBM	100	1.71	1	0	-251.44	-300.89
	50977	XBM	4500	2.17	0.127	0	-284.59	-289.47
	50609	$RBM_{FixProb}$	4500	3.43	0.048	0	-226.90	-293.74
	43569	RBM_{TrPrTr}	4500	3.00	0.040	7	-309.68	-525.63
	5144	XBM	100	2.22	0.113	0	-274.07	-287.43
	4966	$RBM_{FixProb}$	100	2.74	0.078	0	-272.95	**-286.77**
	5220	RBM_{TrPrTr}	100	2.29	0.164	41	-253.09	-289.62
OCR-letters	12800	RBM	100	1.49	1	0	-39.40	-39.58
	13053	XBM	1200	2.14	0.190	0	-33.07	**-33.08**
	12957	$RBM_{FixProb}$	1200	2.80	0.070	0	-40.03	-40.16
	14139	RBM_{TrPrTr}	1200	2.83	0.075	12	-44.17	-45.15
	1710	XBM	100	2.36	0.154	0	-45.70	-45.68
	1743	$RBM_{FixProb}$	100	2.49	0.083	0	-49.20	-49.10
	1960	RBM_{TrPrTr}	100	2.58	0.127	4	-63.72	-63.69
RCV1	15000	RBM	100	1.51	1	0	-52.04	-52.50
	14797	XBM	1400	2.15	0.162	0	-49.22	**-49.68**
	15003	$RBM_{FixProb}$	1400	2.90	0.066	0	-50.06	-50.59
	27555	RBM_{TrPrTr}	1400	2.58	0.081	50	-57.05	-59.47
	1992	XBM	100	2.35	0.151	0	-52.15	-52.30
	1994	$RBM_{FixProb}$	100	2.49	0.081	0	-52.01	-52.17
	2999	RBM_{TrPrTr}	100	2.33	0.147	15	-54.68	-54.94
Web	30000	RBM	100	1.62	1	0	-35.46	-35.43
	29893	XBM	2600	2.17	0.123	0	-30.00	**-30.62**
	29780	$RBM_{FixProb}$	2600	3.20	0.052	0	-45.52	-46.09
	34041	RBM_{TrPrTr}	2600	3.33	0.070	50	-1114.93	-1118.98
	3433	XBM	100	2.28	0.149	0	-33.99	-33.97
	3333	$RBM_{FixProb}$	100	2.62	0.076	0	-38.40	-38.30
	1302	RBM_{TrPrTr}	100	2.42	0.360	2	-252.90	-252.88

to the most important data features, we propose two novel types of Boltzmann machine models, dubbed complex Boltzmann machine and Gaussian complex Boltzmann machine. Looking at both artificial and real-world datasets (i.e. Geographical Origin of Music, MNIST digits, CalTech 101 Silhouettes, and UCI evaluation suite) we show that XBM and GXBM obtain better performance than other two sparse models (i.e. $\text{RBM}_{\text{FixProb}}$/$\text{GRBM}_{\text{FixProb}}$ and $\text{RBM}_{\text{TrPrTr}}$/$\text{GRBM}_{\text{TrPrTr}}$) and we illustrate how they outperform even the fully connected RBM and GRBM, respectively:

(1) Given the same number of hidden neurons, our proposed models exhibit much faster computational time thanks to a smaller number of parameters which have to be computed (up to a few orders of magnitude smaller than in RBM and GRBM) and comparable reconstruction capabilities.
(2) Given the same number of weights, or implicitly a much higher number of hidden neurons for XBM and GXBM, they significantly outperform RBM and GRBM, respectively.

It is worth noting that as the number of neurons increases the order of magnitude between the number of weights in XBM/GXBM and in RBM/GRBM also increases (e.g. one order of magnitude fewer weights in a XBM/GXBM with 100 visible and 100 hidden neurons, and two orders reduction in a XBM/GXBM with 1000 visible and 1000 hidden neurons). This relation will help increasing the typical number of neurons in deep artificial neural networks from the few hundred thousands of today [**13**, **101**] to even billions in the near-future. In turn this will lead to the ability to tackle problems having much higher dimensional data - something that is today unfeasible without performing dimensionality reduction. For instance, when working on ordinary images[7] today is still a common practice to first extract features using standard image processing techniques, and just those features can be served as inputs to deep models - an example can be found in [**189**]. We speculate that another significant benefit of using sparse topologies, as in XBM/GXBM, would be the ability to better disentangle the features extracted automatically by the hidden layer.

To conclude, in this chapter, we have shown empirically on 12 datasets that our proposed models, i.e. XBMs and GXBMs, achieve a very good performance as generative models. We mention that more research has to be done in order to understand why their proposed topology (e.g. the scale-free constraints) makes them to perform so well. Further on, we intend to investigate all these directions and to study analytically how the various parameters of the topology-generation algorithm (implicitly the bipartite graph properties) may increase or decrease the generative and discriminative capabilities of XBMs and GXBMs, respectively.

[7]Please note that a normal RGB image of 1000 by 1000 pixels has 3000000 dimensions.

CHAPTER 5

Quadratic parameter reduction in artificial neural networks

In the previous chapter we pioneered the use of formalized sparse topologies encountered in real-world complex networks, i.e. scale-free and small-world, to improve the performance of restricted Boltzmann machines and reduce their parameters. Here, we take that work to the next level. Taking inspiration from the networks properties of biological neural networks (e.g. sparsity, scale-freeness), we argue that (contrary to general practice) Artificial Neural Networks (ANN), too, should not have fully-connected layers. We show how ANNs perform perfectly well on sparsely-connected layers. Following a Darwinian evolutionary approach, we propose a novel algorithm which evolves an initial random sparse topology (i.e. an Erdős-Rényi random graph) of two consecutive layers of neurons into a scale-free topology, during the ANN training process. The resulted sparse layers can safely replace the corresponding fully-connected layers. Our method allows to quadratically reduce the number of parameters in ANNs, yielding to quadratically faster computational times in both phases (i.e. training and exploitation), at no decrease in accuracy. We demonstrate our claims on two popular ANN types (i.e. restricted Boltzmann machine and multi-layer perceptron), on two types of tasks (i.e. supervised and unsupervised learning), and on 14 benchmark datasets. We anticipate that our approach will enable ANNs having billions of neurons and evolutionary topologies capable of handling complex real-world tasks that are intractable by state-of-the-art methods.

5.1. Introduction

Through the success of deep learning [**110**], Artificial Neural Networks (ANNs) are among the most used artificial intelligence methods nowadays. ANNs have led to major breakthroughs in various domains, such as particle physics [**15**], reinforcement learning [**133**], speech recognition, and so on [**110**]. Typically, ANNs have layers of fully-connected neurons [**110**], which contain most of the network parameters (i.e. the weighted connections), leading to a quadratic number of connections with respect to their number of neurons. In turn, the network size is severely limited, due to obvious computational limitations.

By the very contrast to ANNs, biological neural networks have been demonstrated to have a sparse (rather than dense) topology [**168**, **192**], and also hold other important properties that are instrumental to learning efficiency. These

This chapter is integrally based on: D.C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, A. Liotta: *Evolutionary Training of Sparse Artificial Neural Networks: A Network Science Perspective*, 2017 (submitted for journal publication).

have been extensively studied in [**34**] and include scale-freeness [**19**] and small-worldness [**218**]. Nevertheless, ANNs have not evolved to mimic these topological features [**140**, **144**], which is why in practice they lead to extremely large models. Previous studies have demonstrated that, following the training phase, ANN models end up with weights histograms that peak around zero [**53**,**77**,**227**]. Moreover, in our previous work [**145**], we have hinted a similar fact, as it is depicted in Figure A.7 from Appendix A. Yet, in the machine learning state-of-the-art, sparse topological connectivity is pursued only as an aftermath of the training phase [**77**], which bears benefits only to the exploitation phase.

We claim that topological sparsity must be pursued since the ANN design phase, which leads to a substantial reduction in connections and, in turn, to memory and computational efficiency. At the same time, to be able to make use of standard training algorithms, e.g. Stochastic Gradient Descent (SGD), the structured multi-layer architecture of ANNs has to be preserved. Otherwise we would not be able to train large ANNs with a complete random sparse topology, due to the difficulty of finding suitable optimization algorithms.

In a recent paper, we introduced compleX Boltzmann Machines (XBMs), a sparse variant of Restricted Boltzmann Machines (RBMs), conceived with a sparse scale-free topology [**144**]. XBMs outperform their fully-connected RBMs counterparts and are much faster, both in the training and the exploitation phases. Yet, being based on a fixed sparsity pattern, XBMs may fail to properly model the data distribution. To overcome this limitation, in this chapter we introduce a Sparse Evolutionary Training (SET) procedure, which takes into consideration data distributions and creates sparse bipartite layers suitable to replace the fully-connected bipartite layers in any type of ANNs.

SET follows the natural simplicity of the Darwinian evolutionary approach, which was explored successfully in our previous work on evolutionary function approximation [**221**]. The bipartite ANN layers start from a random sparse topology (i.e. Erdős-Rényi random graph [**57**]), evolving through a random process during the training phase into a scale-free topology. Remarkably, this process does not have to incorporate any constraints to force the scale-free topology. But our evolutionary algorithm is not arbitrary: it follows a phenomenon that takes place in real-world complex networks (such as biological neural networks, and protein interaction networks). Starting from an Erdős-Rényi random graph topology and throughout millenniums of natural evolution, networks end-up with a more structured connectivity, i.e. scale-free [**19**] or small-world [**218**] topologies.

The remainder of this chapter is organized as follows. Section 5.2 presents background knowledge mainly for the benefit of the less specialist reader. Section 5.3 introduces the proposed method, SET. Section 5.4 describes the experiments performed and discusses the results. Finally, Section 5.5 concludes the chapter and proposes future research directions.

5.2. Background

5.2.1. Artificial neural networks

Artificial Neural Networks [**21**] are mathematical models, inspired by biological neural networks, which can be used in all three machine learning paradigms

(i.e. supervised learning [**78**], unsupervised learning [**78**], and reinforcement learning [**193**]). These make them very versatile and powerful, as quantifiable by the remarkable success registered recently by the last generation of ANNs (also known as deep artificial neural networks or deep learning [**110**]) in many fields from computer vision [**110**] to gaming [**133**, **183**]. Just like their biological counterparts, ANNs are composed by neurons and weighted connections between these neurons. Based on their purposes and architectures, there are many models of ANNs, such as restricted Boltzmann machines [**187**], multi layer perceptron [**173**], convolutional neural networks [**111**], recurrent neural networks [**74**], and so on. Many of these ANN models contain fully-connected layers. A fully-connected layer of neurons means that all its neurons are connected to all the neurons belonging to its adjacent layer in the ANN architecture. For the purpose of this chapter, in this section we briefly describe two models that contain fully-connected layers, i.e. Restricted Boltzmann Machines [**187**] and multi layer perceptron [**173**].

A restricted Boltzmann machine is a two-layer, generative, stochastic neural network that is capable to learn a probability distribution over a set of inputs [**187**] in an unsupervised manner. From a topological perspective, it allows only inter-layer connections. Its two layers are: the visible layer, in which the neurons represent the input data; and the hidden layer, in which the neurons represent the features automatically extracted by the RBM model from the input data. Each visible neuron is connected to all hidden neurons through a weighted undirected connection, leading to a fully-connected topology between the two layers. Thus, the flow of information is bidirectional in RBMs, from the visible layer to the hidden layer, and from the hidden layer to the visible layer, respectively. RBMs, beside being very successful in providing very good initialization weights to the supervised training of deep artificial neural network architectures [**82**], are also very successful as stand alone models in a variety of tasks, such as density estimation to model human choice [**163**], collaborative filtering [**176**], information retrieval [**70**], multi-class classification [**104**], and so on.

Multi Layer Perceptron [**173**] (MLP) is a classical feed-forward ANN model that maps a set of input data to the corresponding set of output data. Thus, it is used for supervised learning. It is composed by an input layer in which the neurons represent the input data, an output layer in which the neurons represent the output data, and an arbitrary number of hidden layers in between, with neurons representing the hidden features of the input data (to be automatically discovered). The flow of information in MLPs is unidirectional, starting from the input layer towards the output layer. Thus, the connections are unidirectional and exist just between consecutive layers. Any two consecutive layers in MLPs are fully-connected. There are no connections between the neurons belonging to the same layer, or between the neurons belonging to layers which are not consecutive. In [**47**], it has been demonstrated that MLPs are universal function approximators, so they can be used to model any type of regression or classification problems.

In general, working with ANN models involves two phases: (1) training (or learning), in which the weighted connections between neurons are optimized using various algorithms (e.g. backpropagation procedure combined with stochastic gradient descent [**27**, **174**] used in MLPs, contrastive divergence [**81**] used in RBMs) to minimize a loss function defined by their purpose; and (2) exploitation, in which the optimized ANN model is used to fulfill its purpose.

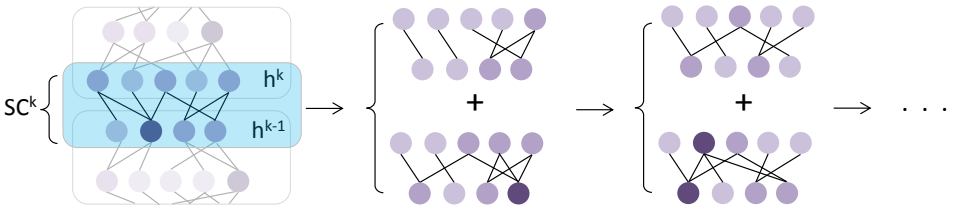


Figure 5.1 – An illustration of the SET procedure. For each Sparse Connected layer, SC^k (left plot), of an ANN at the end of a training epoch a fraction of the weights, the ones closest to zero, are removed (top middle plot). Then, new weighs are added randomly in the same amount as the ones previously removed (bottom middle plot). Further on, a new training epoch is performed (right plot), and the procedure to remove and add weights is repeated. The process continues for a finite number of training epochs, as usual in the ANNs training.

5.2.2. Scale-free complex networks

Complex networks (e.g. biological neural networks, actors and movies, power grids, transportation networks) are everywhere, in different forms, and different fields (from neurobiology to statistical physics [**192**]). Formally, a complex network is a graph with non-trivial topological features, human- or nature-made. One of the most well-known and deeply studied type of topological features in complex networks is scale-freeness, due to the fact that a wide range of real-world complex networks have this topology. A network with a scale-free topology [**19**] is a sparse graph [**51**] that approximately has a power-law degree distribution $P(d) \sim d^{-\gamma}$, where the fraction $P(d)$ from the total nodes of the network has d connections to other nodes, and the parameter γ usually stays in the range $\gamma \in (2, 3)$.

5.3. Sparse Evolutionary Training (SET)

SET is detailed in Appendix B, Algorithm B.4, and exemplified in Figure 5.1. Formally, let us defined a Sparse Connected (SC^k) layer in an ANN. This layer has n^k neurons, collected in a vector $\mathbf{h}^k = [h_1^k, h_2^k, ..., h_{n^k}^k]$. Any neuron from $\mathbf{h}^k$ is connected to an arbitrary number of neurons belonging to layer below $\mathbf{h}^{k-1}$. The connections between the two layers are collected in a sparse weight matrix $\mathbf{W}^k \in \mathbb{R}^{n^{k-1} \times n^k}$. Initially, $\mathbf{W}^k$ is a Erdős-Rényi random graph, in which the probability of the connection between the neuron h_i^k and h_j^{k-1} to exist is given by:

$$p(W_{ij}^k) = \frac{\epsilon(n^k + n^{k-1})}{n^k n^{k-1}} \tag{5.1}$$

whereby $\epsilon \in \mathbb{R}^+$ is a parameter of SET controlling the sparsity level. If $\epsilon \ll n^k$ and $\epsilon \ll n^{k+1}$ then there is a linear number of connections (i.e. non-zero elements), $n^W = |\mathbf{W}^k| = \epsilon(n^k + n^{k-1})$, with respect to the number of neurons in the sparse layers. In the case of fully-connected layers the number of connections is quadratic, i.e. $n^k n^{k-1}$.

However, it may be that this random generated topology is not suited to the particularities of the data that the ANN model tries to learn. To overcome this situation, during the training process, after each training epoch, a fraction ζ of the smallest positive weights and of the highest negative weights of SC^k is removed.

These removed weights are the ones closest to zero, thus we do not expect that their removal will notably change the model performance [**77**]. Next, to let the topology of SC^k to evolve so as to fit the data, an amount of new random connections, equal to the amount of weights removed previously, is added to SC^k. In this way, the number of connections in SC^k remains constant during the training process. After the training ends, we keep the topology of SC^k as the one obtained after the last weight removal step, without adding new random connections.

It is worth highlighting that in the initial phase of conceiving the SET procedure, the weight-removal and weight-addition steps after each training epoch were introduced intuitively. Still, in the last phases of preparing this chapter we have found that there is a similarity between SET and a phenomenon which takes place in biological brains, named *synaptic shrinking during sleep*. This phenomenon has been demonstrated recently in two papers published in the Science journal in February 2017 [**50**, **54**]. In short, it was found that during sleep the weakest synapses in the brain shrink, while the strongest synapses remain unaltered, supporting the hypothesis that one of the core functions of sleeping is to renormalize the overall synaptic strength increased while awake [**50**]. By keeping the analogy, this is - in a way - what happens also with the ANNs during the SET procedure.

5.4. Experiments and results

5.4.1. Evaluation method

We evaluate SET in two types of ANNs, i.e. restricted Boltzmann machine [**187**], and Multi Layer Perceptron (MLP) [**110**], to experiment with both unsupervised and supervised learning. In total, we evaluate SET on 14 benchmark datasets, as detailed in Table 5.1, covering a wide range of fields in which ANNs are employed, such as biology, physics, computer vision, data mining, and economics. We also assess SET in combination with two different training methods, i.e. contrastive divergence [**81**] and stochastic gradient descent [**110**].

5.4.2. SET performance on restricted Boltzmann machines

First, we have analyzed the performance of SET on a bipartite undirected stochastic ANN model, i.e. restricted Boltzmann machine [**187**], which is popular for its unsupervised learning capability [**20**] and high performance as a feature extractor and density estimator [**163**]. The new model derived from the SET procedure was dubbed SET-RBM. In all experiments, we set $\epsilon = 11$, and $\zeta = 0.3$, performing a small random search.

There are few studies on RBM connectivity sparsity [**144**]. Still, to get a good estimation of SET-RBM capabilities we compared it against RBM$_{FixProb}$ [**144**] (a sparse RBM model with a fixed Erdős-Rényi topology), fully-connected RBMs, and with the state-of-the-art results of XBMs from [**144**]. We performed experiments on 11 benchmark datasets coming from various domains, as depicted in Table 5.1, using the same splitting for training and testing data as in [**144**]. All models were trained for 5000 epochs using Contrastive Divergence [**81**] (CD) with 1, 3, and 10 CD steps, a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0002, as discussed in [**83**]. We evaluated the generative performance of the

Table 5.1 – Datasets characteristics. The data used in this chapter have been chosen to cover a wide range of fields where ANNs have the potential to advance state-of-the-art, including biology, physics, computer vision, data mining, and economics.

Experiments type	**Dataset**		**Dataset Properties**				
			Domain	**Data type**	**Features [#]**	**Train samples [#]**	**Test samples[#]**
Assessment of RBMs variants	UCI evaluation suite [**105**]	ADULT	households	binary	123	5000	26147
		Connect4	games	binary	126	16000	47557
		DNA	genetics	binary	180	1400	1186
		Mushrooms	biology	binary	112	2000	5624
		NIPS-0-12	documents	binary	500	400	1240
		OCR-letters	letters	binary	128	32152	10000
		RCV1	documents	binary	150	40000	150000
		Web	Internet	binary	300	14000	32561
	CalTech 101 Silhouettes [**128**]	16x16	images	binary	256	4082	2302
		28x28	images	binary	784	4100	2307
	MNIST		digits	binary	784	60000	10000
Assessment of MLPs variants	MNIST		digits	grayscale	784	60000	10000
	CIFAR10		images	RGB colors	3072	50000	10000
	HIGGS [**15**]		particle physics	real values	28	10500000	500000

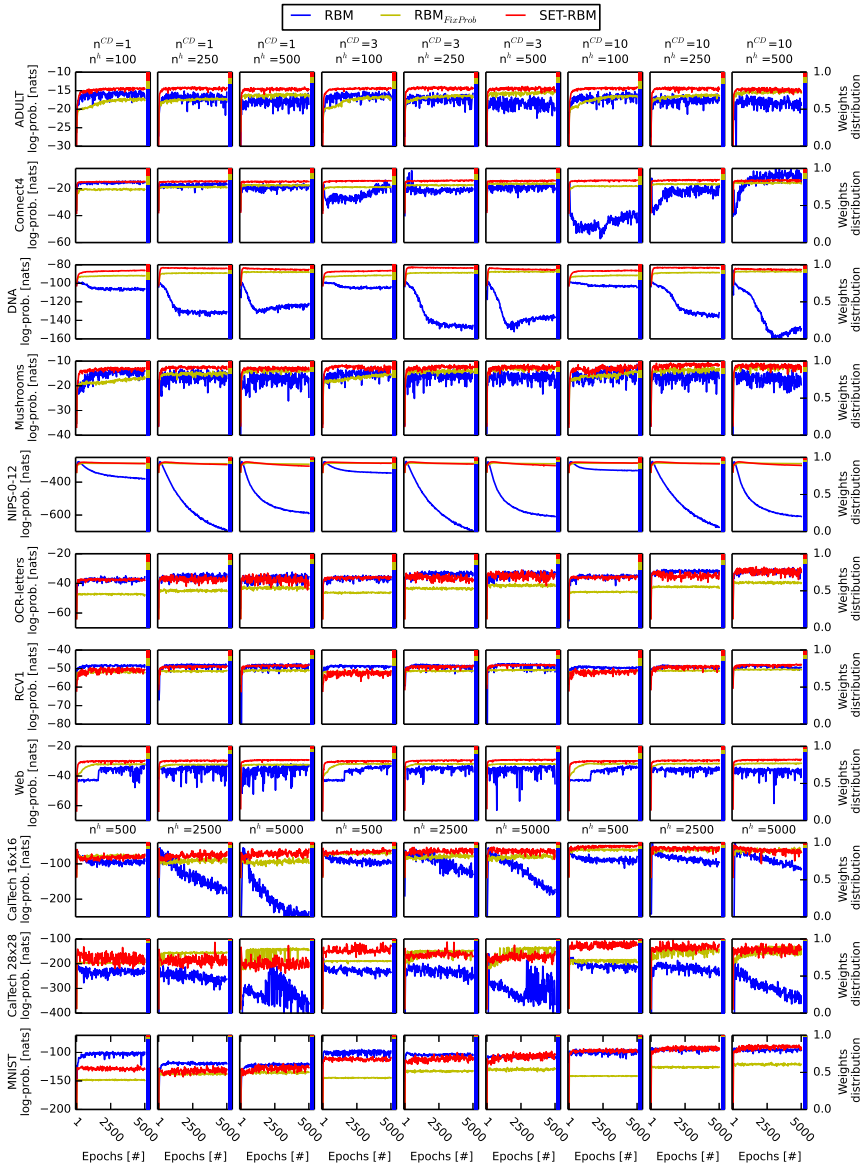


Figure 5.2 – Experiments with RBM variants using 11 benchmark datasets. For each model studied we have considered three cases for the number of Contrastive Divergence steps $n^{CD} = \{1, 3, 10\}$, and three cases for the number of hidden neurons (n^h). For the first 8 datasets (from top to bottom) we have used $n^h = \{100, 250, 500\}$, and for the last three datasets we have used $n^h = \{500, 2500, 5000\}$. The x-axes show the training epochs; the left y-axes show the average log-probabilities computed on the test data with AIS [**177**]; and the right y-axes (the stacked bar on the right part of the plots) reflect the fraction given by the n^W of each model over the sum of the n^W of all three models. Overall, SET-RBM outperforms the other two models in most of the cases. Also, it is interesting to see that SET-RBM and RBM$_{FixProb}$ are much more stable and do not present the over-fitting problems of RBM.

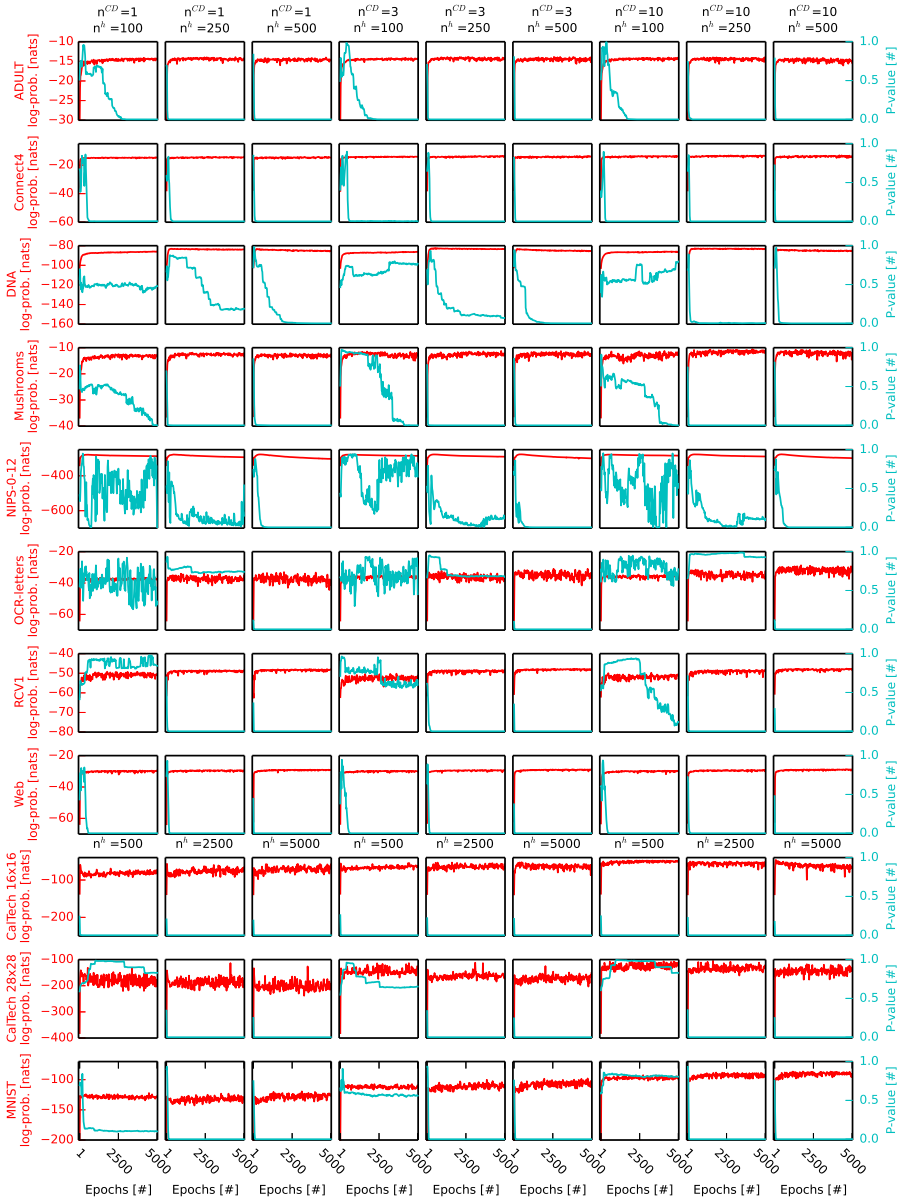


Figure 5.3 – SET-RBM evolution towards a scale-free topology. We have considered three cases for the number of Contrastive Divergence steps $n^{CD} = \{1, 3, 10\}$, and three cases for the number of hidden neurons (n^h). For the first 8 datasets (from top to bottom) we have used $n^h = \{100, 250, 500\}$, and for the last three datasets we have used $n^h = \{500, 2500, 5000\}$. The x-axes show the training epochs; the left y-axes (red color) show the average log-probabilities computed for SET-RBMs on the test data with AIS [**177**]; and the right y-axes (cyan color) show the p-values computed between the degree distribution of the hidden neurons in SET-RBM and a power-law distribution. We may observe that for models with a high enough number of hidden neurons, the SET-RBM topology always tends to become scale-free.

scrutinized models by computing the log-probabilities on the test data using Annealed Importance Sampling (AIS) [**177**], setting all parameters as in [**144**, **177**]. We have used MATLAB for this set of experiments. We implemented SET-RBM and RBM$_{FixProb}$ ourselves; while for RBM and AIS we have adapted the code provided by [**177**].

Figure 5.2 depicts the models performance on all datasets, using various amounts of hidden neurons; while Table 5.2 summarizes the results, presenting the best performer for each type of model for each dataset. In 7 out of 11 datasets, SET-RBM outperforms the fully-connected RBMs, while reducing the parameters by a few orders of magnitude. For instance, on the MNIST dataset, SET-RBM reaches -86.41 *nats*, with a 5.29-fold improvement over the fully-connected RBM, and a parameters reduction down to 2%. In 10 out of 11 datasets, SET-RBM outperforms XBM, which represents the state-of-the-art results on these datasets for sparse variants of RBM [**144**].

Figure 5.2 shows striking results on stability. While fully-connected RBMs show instability and over-fitting issues, the SET procedure stabilizes SET-RBMs and avoids over-fitting. This situation can be observed more often when a high number of hidden neurons is chosen (columns 2, 3, 5, 6, 8, and 9 of Figure 5.2). For instance, if we look at the DNA dataset, independently on the values of n^h and n^{CD} (Figure 5.2, third row), we may observe that SET-RBMs are very stable after they reach around -85 *nats*, having almost a flat learning behavior after that point. Contrary, on the same dataset, the fully-connected RBMs have a very short initial good learning behavior (for few epochs) and, after that, they go up and down during the 5000 epochs analyzed, reaching the minimum performance of -160 *nats* (Figure 5.2, third row, last column). We have to mention that these good stability and over-fitting avoidance capacity, are induced not just by the SET procedure, but also by the sparsity itself, as RBM$_{FixProb}$, too, has a stable behavior in almost all the cases.

We finally verified our initial hypothesis about sparse connectivity in SET-RBM. Figure 5.3 shows how the connectivity naturally evolves towards a scale-free topology. To assess this fact, we have used the null hypothesis from statistics [**58**], which assumes that there is no relation between two measured phenomena. To see if the null hypothesis between the degree distribution of the hidden neurons and a power-law distribution can be rejected, we have computed the p-value [**160**] between them. To reject the null hypothesis the p-value has to be lower than a statistically significant threshold of 0.05. In all cases (all plots of Figure 5.3), looking at the p-values (y-axes to the right of the plots), we can see that at the beginning of the learning phase the null hypothesis is not rejected. This was to be expected, as the initial degree distribution of the hidden neurons is binomial due to the randomness of the Erdős-Rényi random graphs [**158**] used to initialize the SET-RBMs topology.

Subsequently, during the learning phase, we can see that, in many cases, the p-values decrease considerably under the 0.05 threshold during training. When these situations occur, it means that the degree distribution of the hidden neurons in SET-RBM starts to approximate a power-law distribution. As to be expected, the cases with fewer neurons (e.g. Figure 5.3, fifth row, first column) fail to evolve to scale-free topologies, while the cases with more neurons always evolve towards a scale-free topology (Figure 5.3, columns 3, 6, and 9). To summarize, in 70 out

of 99 cases studied, the SET-RBMs topology evolves clearly during the learning phase from an Erdős-Rényi topology towards a scale-free one.

We can conclude that the SET procedure is coherent with real-world complex networks, whereby nodes connections tend to evolve into scale-free topologies [**18**]. This feature has important implications in ANNs: we could envision a computational time reduction by reducing the number of training epochs, if we would use for instance preferential attachment algorithms [**9**] to evolve faster the topology of the bipartite ANN layers towards a scale-free one. Of course, this possible improvement has to be treated carefully, as forcing the model topology to evolve unnaturally faster into a scale-free topology may be prone to errors - for instance, the data distribution may not be perfectly matched.

5.4.3. SET performance on multi layer perceptron

To better explore the capabilities of SET, we have also assessed its performance on classifications tasks based on supervised learning. We developed a variant of Multi Layer Perceptron (MLP) [**110**], dubbed SET-MLP, in which the fully-connected layers have been replaced with sparse layers obtained through the SET procedure, with $\epsilon = 20$, and $\zeta = 0.3$. We kept the ζ parameter as in the previous case of SET-RBMs, while for the ϵ parameter we performed a small random search. We compared SET-MLP to a standard fully-connected MLP, and to a sparse variant of MLP having a fixed Erdős-Rényi topology, dubbed $\text{MLP}_{FixProb}$. For the assessment, we have used three benchmark datasets (Table 5.1), two coming from the computer vision domain (MNIST and CIFAR10), and one from particle physics (the HIGGS dataset [**15**]). In all cases, we have used the same data processing techniques, network architecture, training method (i.e. Stochastic Gradient Descent [**110**] with fixed learning rate of 0.01, momentum of 0.9, and weight decay of 0.0002), and a dropout rate of 0.3 (Table 5.3). The only difference between MLP, $\text{MLP}_{FixProb}$, and SET-MLP, consisted in their topological connectivity. We have used Python and the Keras library [**40**] with Theano back-end [**7**] for this set of experiments. For MLP we have used the standard Keras implementation, while we implemented ourselves SET-MLP and $\text{MLP}_{FixProb}$ on top of the standard Keras libraries.

The results depicted in Figure 5.4 show how SET-MLP outperforms $\text{MLP}_{FixProb}$. Moreover, SET-MLP always outperforms MLP, while having two orders of magnitude fewer parameters. Looking at the CIFAR10 dataset, we can see that with only just 1% of the weights of MLP, SET-MLP leads to significant gains. At the same time, SET-MLP has comparable results with state-of-the-art MLP models after these have been carefully fine-tuned. To quantify, the second best MLP model in the literature on CIFAR10 reaches about 74.1% classification accuracy [**204**] and has 31 million parameters: while SET-MLP reaches a better accuracy (74.84%) having just about 0.3 million parameters. Moreover, the best MLP model in the literature on CIFAR10 has 78.62% accuracy [**117**], with about 12 million parameters, while also benefiting from a pre-training phase [**80**, **82**]. Although we have not pre-trained the MLP models studied here, we should mention that SET-RBM can be easily used to pre-train a SET-MLP model to further improve performance.

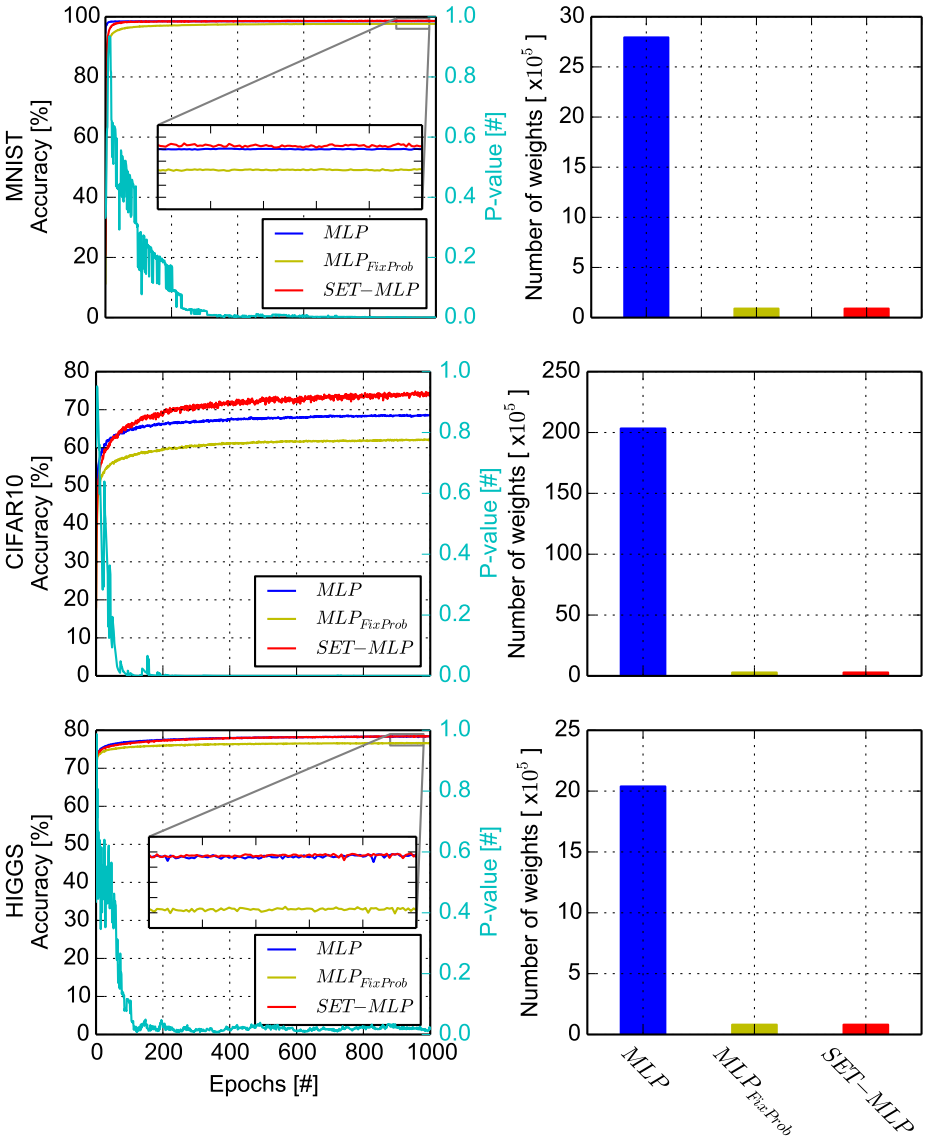


Figure 5.4 – Experiments with MLP variants using 3 benchmark datasets. The plots on the left side reflect models performance in terms of classification accuracy (left y-axes) over training epochs (x-axes); the right y-axes of the left plots give the p-values computed between the degree distribution of the hidden neurons of the SET-MLP models and a power-law distribution, showing how the SET-MLP topology becomes scale-free over training epochs. The bar plots on the right side depict the number of weights of the three models on each dataset. The most striking situation happens for the CIFAR10 dataset (second row) where the SET-MLP model outperforms drastically the MLP model, while having approximately 100 times fewer parameters.

Table 5.2 – Summarization of the experiments with RBM variants. On each dataset, we report the best average log-probabilities obtained with AIS on the test data for each model. n^h represents the number of hidden neurons, n^{CD} the number of CD steps, and n^W the number of weights in the model.

Dataset		**RBM**				**RBM**$_{FixProb}$				**SET-RBM**				**XBM**			
		log-prob.	n^h	n^W	n^{CD}	**log-prob.**	n^h	n^W	n^{CD}	**log-prob.**	n^h	n^W	n^{CD}	**log-prob.**	n^h	n^W	n^{CD}
UCI evaluation suite	ADULT	-14.91	100	12300	10	-14.79	500	4984	10	-13.85	500	4797	3	-15.89	1200	12911	1
	Connect4	-5.01	500	63000	10	-15.01	500	5008	10	-13.12	500	4820	10	-17.37	1200	12481	1
	DNA	-85.97	500	90000	10	-86.90	500	5440	10	-82.51	250	3311	3	-83.17	1600	17801	1
	Mushrooms	-11.35	100	11200	10	-11.36	500	4896	10	-10.63	250	2787	10	-14.71	1000	10830	1
	NIPS-0-12	-274.60	250	125000	3	-282.67	500	8000	10	-276.62	500	7700	3	-287.43	100	5144	1
	OCR-letters	-29.33	500	64000	10	-38.58	500	5024	10	-28.69	500	4835	10	-33.08	1200	13053	1
	RCV1	-47.24	500	75000	3	-50.34	500	5200	10	-47.60	500	5005	10	-49.68	1400	14797	1
	Web	-31.74	500	150000	1	-31.32	500	6400	10	-28.74	500	6160	10	-30.62	2600	29893	1
CalTech 101 Silhouettes	16x16	-28.41	2500	640000	10	-53.25	5000	42048	10	-46.08	5000	40741	10	-69.29	500	6721	1
	28x28	-159.51	5000	3920000	3	-126.69	5000	46272	10	-104.89	2500	25286	10	-142.96	1500	19201	1
MNIST		-91.70	2500	1960000	10	-117.55	5000	46272	10	-86.41	5000	44536	10	-85.21	27000	387955	1:25

Table 5.3 – Summarization of the experiments with MLP variants. On each dataset, we report the best classification accuracy obtained by each model on the test data. n^W represents the number of weights in the model.

Dataset	**Data augmentation**	**Architecture**	**Activation**	**MLP**		**MLP**$_{FixProb}$		**SET-MLP**	
				Accuracy [%]	n^W	**Accuracy** [%]	n^W	**Accuracy** [%]	n^W
MNIST	no	784-1000-1000-1000-10	SReLu	98.55	2794000	97.68	89797	98.74	89797
CIFAR10	yes	3072-4000-1000-4000-10	SReLu	68.70	20328000	62.19	278630	74.84	278630
HIGGS	no	28-1000-1000-1000-2	SReLu	78.44	2038000	76.69	80614	78.47	80614

Regarding the topological features, we can see from Figure 5.4 that, similarly to what was found in the SET-RBM experiments (Figure 5.3), the hidden neuron connections in SET-MLP rapidly evolve towards a power-law distribution.

Considering the different datasets under scrutiny, we should stress that we have assessed both image-intensive and non-image sets. On image datasets, Convolutional Neural Networks (CNNs) [**110**] typically outperform MLPs. These, in fact, matches perfectly with the SET procedure. For instance, SET may be used to replace all CNNs fully connected layers with sparse evolutionary counterparts. The benefit would be two-fold: to reduce the total number of parameters in CNNs, and to permit the use of larger CNN models. However, CNNs are not viable on other types of high-dimensional data, such as biological data (e.g. [**48**]), or theoretical physics data (e.g. [**15**]). In those cases, MLPs will be a better choice. This is in fact the case of the HIGGS dataset (Figure 5.4, last row), where SET-MLP achieves 78.47% classification accuracy and has about 90000 parameters. Whereas, one of the best MLP models in the literature achieved a 78.54% accuracy with three many times as many parameters [**117**].

5.5. Conclusion

In this chapter we have introduced SET, a simple procedure to replace ANNs fully-connected bipartite layers with sparse layers. We have validated our approach on 14 datasets (from different domains) and on two widely used ANN models, i.e. RBMs and MLPs. We have evaluated SET in combination with two different training methods, i.e. contrastive divergence and stochastic gradient descent. We showed that SET is capable to *quadratically* reduce the number of parameters of bipartite neural networks layers, at no decrease in accuracy. In most of the cases, SET-RBMs and SET-MLPs outperform their fully-connected counterparts.

SET can be widely adopted to reduce the fully-connected layers into sparse topologies in other types of ANNs, e.g. convolutional neural networks [**110**], recurrent neural networks [**110**], deep reinforcement learning networks [**133**, **183**], and so on. SET may prove to be the basis of much larger ANNs, possibly on a billion-node scale. This will be enabled by the linear relation between the number of neurons and the amount of connections between them. These networks will have much more representational power, and better adaptive capabilities than the current state-of-the-art ANNs, and will push artificial intelligence well beyond its current boundaries.

CHAPTER 6

Conclusions and discussions

This chapter, summarizes the thesis contributions and discusses possible future research directions.

6.1. Conclusions

In this thesis, we followed a multidisciplinary approach inspired by the complex systems paradigm. First, as briefly discussed in Section 1.4, we addressed real-world challenges by proposing novel methods that were either purely based on network science or purely based on artificial intelligence algorithms or new applications of existing algorithms. Although we were successful in addressing specific challenges, our solutions were still confined by scalability bounds. Further on, these made us to define some fundamental research questions (Section 1.5) to help increasing the various scalability bounds of the used algorithms, i.e. centrality metrics in complex networks and artificial neural networks, as follows.

6.1.1. Thesis contributions

In Chapter 2, we introduced a new viewpoint to understand and model complex networks, whereby we overlay an artificial homogeneous system over a network to unveil the network's hidden properties. We could then propose a novel algorithm to compute centrality in complex networks, dubbed GOT. GOT can compute all nodes and links centralities, treated together, in a polylogarithmic time with respect to the number of nodes in the network. It has the computational simplicity of nature-inspired swarm algorithms, while performing human-behaviour like computations [**179**] (namely, an egoistic behaviour). We demonstrated on thousands of simulated networks with different types of topologies, as well as on real-world networks, that GOT can compute the whole range of link and node strengths of any complex network, while being more accurate, much faster, scalable and technologically viable than the state-of-the-art centrality metrics. Moreover, we have also used it to confirm well-established findings about a non-obvious behaviour of natural networks [**175**]. Natively, GOT permits to investigate much larger networks, which are not tractable with current algorithms - for instance GOT would require less than 9 seconds to compute the centrality of the one-billion network formed by Facebook user devices.

In Chapter 3, we have proposed generative replay, a memory-less, brain-inspired approach capable to replace experience replay, a standard procedure used

This chapter is partly based on: D.C. Mocanu: *On the synergy of network science and artificial intelligence*, International Joint Conference on Artificial Intelligence (IJCAI), 2016, New York, USA.

to train ANNs on-line. Generative replay uses the generative capabilities of restricted Boltzmann machines to generate approximations of past experiences, instead of recording them, as experience replay does. Thus, the RBM can be trained online, and does not require the system to store any of the observed data points. In the experiments performed, we showed that generative replay outperforms experience replay on RBMs trained on-line, reaching a performance which is even comparable to RBMs trained off-line. Furthermore, the generative replay concept may be used in combination with generative artificial neural network models to serve dynamic approximations of past experiences to any ANN model that performs on-line learning.

In Chapter 4, we looked at the building block of deep learning, i.e. restricted Boltzmann machines, from a topological perspective, bringing insights from network science. Firstly, we proved that RBMs and GRBMs are small-world bipartite networks. Secondly, we proposed a novel algorithm to generate scale-free and small-world topologies in sparse variants of RBMs and GRBMs. We dubbed the novel obtained models, complex Boltzmann machine and Gaussian complex Boltzmann machine, respectively. To achieve a good performance, in the topology of these models we also considered some local neighborhoods of visible neurons, and we fit the most connected visible neurons to the most important data features.

Looking at both artificial and real-world datasets (i.e. Geographical Origin of Music, MNIST digits, CalTech 101 Silhouettes, and UCI evaluation suite) we showed that XBM and GXBM obtain better performance than other two sparse models (i.e. $\text{RBM}_{\text{FixProb}}$/$\text{GRBM}_{\text{FixProb}}$ and $\text{RBM}_{\text{TrPrTr}}$/$\text{GRBM}_{\text{TrPrTr}}$) and we illustrated how they outperform even the fully connected RBM and GRBM, respectively. We found that: (1) given the same number of hidden neurons, our models exhibit much faster computational time, thanks to a smaller number of parameters which have to be computed (up to a few orders of magnitude smaller than in RBM and GRBM) and a comparable reconstruction capabilities; (2) given the same number of weights, or implicitly a much higher number of hidden neurons for XBM and GXBM, they significantly outperform RBM and GRBM, respectively.

In Chapter 5, taking the work from Chapter 4 further, we have introduced the Sparse Evolutionary Training (SET) procedure. SET is a simple, yet effective procedure that reduces an ANN fully-connected bipartite topology to sparse connected topology, before the training phase. We have validated our approach on data coming from various domains, on 14 datasets and on two widely used ANN models, i.e. RBMs and MLPs, using two different optimization methods, i.e. contrastive divergence and stochastic gradient descent. We demonstrated that SET is capable to *quadratically* reduce the number of parameters of bipartite neural networks layers, at no decrease in accuracy. We highlight that, in most of the cases, SET-RBMs and SET-MLPs outperform their fully connected counterparts. We advocate that SET will be widely adopted to replace the fully connected layers with sparse ones in other types of ANNs, e.g. convolutional neural networks [**110**], recurrent neural networks [**110**], deep reinforcement learning networks [**133**,**183**], and so on. Further on, SET may prove to be the base of a new generation of ANNs (much larger than the state-of-the-art ANN models) with billions of neurons and sparse connectivity, thanks their linear relation between neurons and connections.

Such networks would have much more representational power, and better adaptive capabilities than the current state-of-the-art ANNs, and will push artificial intelligence well beyond its current boundaries.

To conclude, this thesis addresses a range of topics around the common theme of *network efficiency*. We explore the fascinating opportunities that arise when AI is employed to master network complexity, and *vice versa*. The applicability of such fundamental concepts is vast, with the possibility to make impact on virtually any domain whereby problems can be modeled as networks. We have explored examples in communication networks [**23**], wireless sensor networks [**38**, **100**, **134**, **148**], smart grids [**135**, **150**], computer vision [**141**, **142**, **149**], computer security [**146**], transfer learning [**10**, **28**], and multimedia quality of experience [**136**, **137**, **145**, **207**–**212**].

There is also enormous potential in employing scalable artificial neural networks onto other problems that cannot yet be tackled due to the scalability boundaries of current methods, e.g. understanding of the brain, understanding of very high resolution images and videos, online learning in low-resources devices, etc.

Looking at the synergy between network science, artificial intelligence, and biological neural networks, we have been able to push the scalability bounds of various networks algorithms much beyond their state-of-the-art. Our combined approach to complexity and AI goes beyond the current methods, which tend to focus on either of the two, independently.

6.1.2. Limitations

Although the proposed approach, to study the synergy between network science and artificial intelligence, has proven to be very successful, its broadness makes it very difficult to be tackled in just one PhD research. To avoid having the contributions lost among lofty goals of connecting two fields, we had to narrow the focus down and to focus just on the scalability issues of various networks algorithms. Even here, we had to narrow even more and to address just few intriguing aspects, as described in Chapters 2-5. Thus, we had to let aside very challenging research directions such as communities and robustness in complex networks [**18**], deep reinforcement learning [**133**], and so on. Moreover, each of the contributions proposed in Chapter 2-5 suffer from their own limitations, as follows.

In Chapter 2, the analysis of the parallel capabilities of the GOT algorithm assumes ideal conditions. In reality, its parallel implementation on a single (super) computer would be prone to the parallel programming limitations (e.g. synchronization problems, memory utilization, expensive communication between processes). Alternatively, its parallel implementation in an Internet of Things like scenario may be affected by the communication network problems (e.g. network load, network impairments, packet losses).

Further on, the main flaw of the RBM_{OCD} model presented in Chapter 3 is the sensitivity to the meta-parameters choice, as in the case of many other ANN models. Specifically, this sensitivity is given by the meta-parameters common to any RBM model, but also by the two meta-parameters specific to the RBM_{OCD} model, i.e. the number of Gibbs sampling steps for the generation of the new training data points, and the number of new data points generated in each epoch with the RBM_{OCD} model using Gibbs sampling.

The main limitations of the XBM and GXBM models, proposed in Chapter 4, have been successfully addressed by the SET procedure, proposed in Chapter 5. Even so, the fact that most of the deep learning optimized hardware (e.g. GPUs) has implemented just optimized multiplications for dense matrices may influence the time until the XBM, GXBM, and SET based ANN models may start to be used in the production environment. However, we hope that our research will trigger the developing of new hardware and software for deep learning which have incorporated optimizations also for operations with sparse matrices. Moreover, all of the above limitations represent virtually possibilities for future research.

6.2. Future research directions

This research may be expanded in many directions. Let us group them into the two main categories of 'applied research' and 'fundamental research'. The applied research direction is straightforward and assumes applying the novel algorithms proposed in the research chapters of this thesis to real-world challenges, as we described in Section 1.4 where we tackled domains such as, wireless sensor networks, computer security, transfer learning, computer vision, quality of experience, and smart grid. At the end of each research chapters, we exemplified how the specific algorithm proposed there could be used in the real-world. The fundamental research direction could be furthered by continuing to explore the synergy between network science, artificial intelligence, and biological principles of nature. We can foresee two time horizons, as follows.

Short term horizon. Key open research directions are:

(1) An important goal of SET is to train ANNs (Chapter 5) by identifying the most important connections of the neural network to remove topological redundancy. In Chapter 5, we have used just a very simple technique to determine the connections importance, by considering that the connections which have the weights closest to zero are not important. Further improvements may be achieved using centrality metrics, for instance using GOT to establish the relative importance of nodes and links (Chapter 2) prior to training the ANN.

(2) SET tends to end-up with scale-free ANN topologies, but these are not enforced (they are achieved as part of the natural evolution of the procedure). It would be interesting to see if training time can be further reduced, using specialized algorithms, such as preferential attachment [**9**], to enforce scale-freeness during training.

(3) Combining generative replay, reinforcement learning, and the sparse artificial neural networks created with SET may permit to create scalable deep reinforcement learning models. On one side, these may be small enough to run properly in low-resources devices such as the one belonging to the Internet of Things. On the other hand, it will be possible to master much larger learning systems.

Longer term horizon. An interesting research direction would be to try to combine traditional AI techniques (i.e. knowledge representation, logic reasoning) with deep learning. This represents an important problem and an active research area, as highlighted by a recent paper [**103**]. One possibility to tackle it would be to follow our incipient approach, as proposed in [**146**], using restricted Boltzmann machines as density estimators for the inductive logic rules.

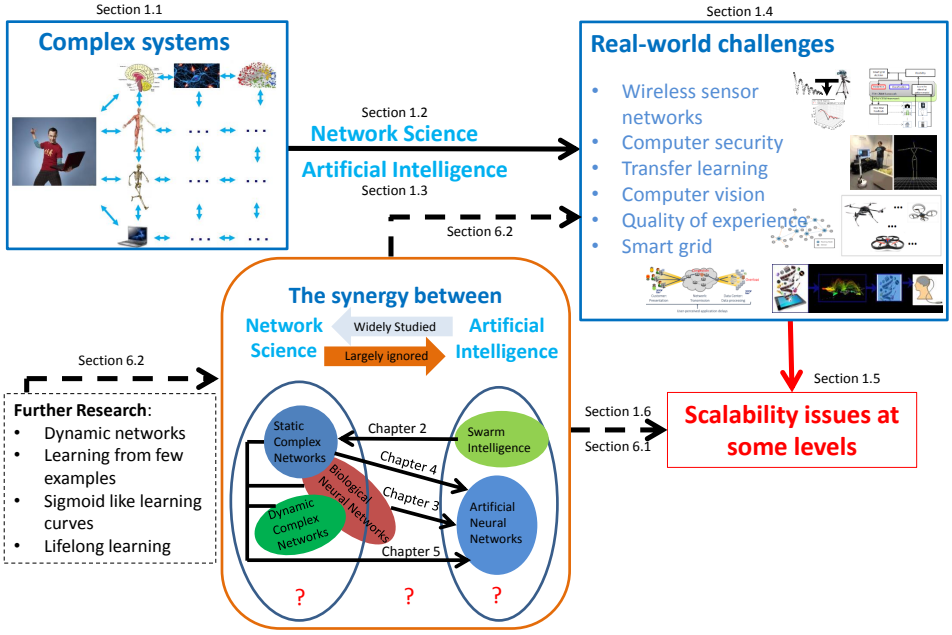


Figure 6.1 – Thesis storyline. The newly added red question marks in comparison with Figure 1.3 point out where to look for future research directions.

Another possibility, would be to study two of the most important challenges in artificial neural networks. The first one relates to the high number of examples that ANNs need to rely upon for learning. The second one is the slow learning curve of gradient based optimization methods. These do not follow the strategy of human learning, which have a much higher generalization power and can learn new concepts using just few labeled examples or even purely unsupervised. Furthermore, the learning curve in humans is sigmodal, which is not the case of gradient based learning. Intuitively, one would hypothesize that if we could more accurately follow the laws of nature we would make new breakthroughs in machine learning, particularly in generalization capability, evolutionary and continuously learning. That would require achieving further insights into the dynamics of biological neural networks, looking from a network science perspective, while trying to find out how to answer to the red question marks sketched in Figure 6.1.

Bibliography

1. *Methodology for the subjective assessment of the quality of television pictures*, Recommendation ITU-R BT.500-13, Jan 2012.
2. *Qualinet white paper on definitions of quality of experience*, European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), March 2013.
3. BP Abbott, R Abbott, TD Abbott, MR Abernathy, F Acernese, K Ackley, C Adams, T Adams, P Addesso, RX Adhikari, et al., *Observation of gravitational waves from a binary black hole merger*, Physical Review Letters **116** (2016), no. 6, 061102.
4. H. Ackley, E. Hinton, and J. Sejnowski, *A learning algorithm for boltzmann machines*, Cognitive Science (1985), 147–169.
5. Sander Adam, Lucian Busoniu, and Robert Babuska, *Experience replay for real-time reinforcement learning control.*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **42** (2012), no. 2, 201–212.
6. E. Aguiar, A. Riker, A. Abelem, E. Cerqueira, and Mu Mu, *Video quality estimator for wireless mesh networks*, Quality of Service (IWQoS), 2012 IEEE 20th International Workshop on, June 2012, pp. 1–9.
7. Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, and Christof Angermueller et al., *Theano: A Python framework for fast computation of mathematical expressions*, arXiv e-prints **abs/1605.02688** (2016).
8. Ráka Albert, Hawoong Jeong, and Albert-László Barabási, *Error and attack tolerance of complex networks*, Nature **406** (2000), 378–382.
9. Réka Albert and Albert-László Barabási, *Statistical mechanics of complex networks*, Rev. Mod. Phys. **74** (2002), 47–97.
10. Haitham Bou Ammar, Eric Eaton, Matthew E Taylor, Decebal Constantin Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls, *An automated measure of mdp similarity for transfer in reinforcement learning*, Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
11. P. W. Anderson, *More is different*, Science **177** (1972), no. 4047, 393–396.
12. Sinan Aral and Dylan Walker, *Identifying influential and susceptible members of social networks*, Science **337** (2012), no. 6092, 337–341.
13. Jimmy Ba and Rich Caruana, *Do deep nets really need to be deep?*, Advances in Neural Information Processing Systems 27, 2014, pp. 2654–2662.
14. Hakan Bakrcolu and Takn Koak, *Survey of random neural network applications*, European Journal of Operational Research **126** (2000), no. 2, 319 – 330.
15. P. Baldi, P. Sadowski, and D. Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, Nature Communications **5** (2014), 4308.
16. Dana Harry Ballard and Christopher M. Brown, *Computer vision*, 1st ed., Prentice Hall Professional Technical Reference, 1982.
17. Albert-Laszlo Barabasi, *The network takeover*, Nature Physics **8** (2012), 14–16.
18. Albert-László Barabási, *Network science*, 2016.
19. Albert-László Barabási and Réka Albert, *Emergence of scaling in random networks*, Science **286** (1999), no. 5439, 509–512.
20. Yoshua Bengio, *Learning deep architectures for ai*, Found. Trends Mach. Learn. **2** (2009), no. 1, 1–127.
21. Christopher M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
22. E. Bonabeau, M. Dorigo, and G. Theraulaz, *Inspiration for optimization from social insect behaviour*, Nature **406** (2000), 39–42.

23. K. Borchert, M. Hirth, T. Zinner, and D. C. Mocanu, *Correlating qoe and technical parameters of an sap system in an enterprise environment*, 2016 28th International Teletraffic Congress (ITC 28), vol. 03, Sept 2016, pp. 34–36.
24. Stephen P. Borgatti, *Centrality and network flow*, Social Networks **27** (2005), no. 1, 55 – 71.
25. Hedde HWJ Bosman, Giovanni Iacca, Arturo Tejada, Heinrich J. Wrtche, and Antonio Liotta, *Spatial anomaly detection in sensor networks using neighborhood information*, Information Fusion **33** (2017), 41 – 56.
26. H.H.W.J. Bosman, G. Iacca, H.J. Wortche, and A. Liotta, *Online fusion of incremental learning for wireless sensor networks*, Data Mining Workshop (ICDMW), 2014 IEEE International Conference on, Dec 2014, pp. 525–532.
27. Léon Bottou and Olivier Bousquet, *The tradeoffs of large scale learning*, Advances in Neural Information Processing Systems (J.C. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), vol. 20, NIPS Foundation (http://books.nips.cc), 2008, pp. 161–168.
28. Haitham Bou-Ammar, Decebal Constantin Mocanu, Matthew E. Taylor, Kurt Driessens, Karl Tuyls, and Gerhard Weiss, *Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines*, Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 8189, Springer Berlin Heidelberg, 2013, pp. 449–464.
29. Ulrik Brandes, *A faster algorithm for betweenness centrality*, Journal of Mathematical Sociology **25** (2001), 163–177.
30. Ulrik Brandes, *On variants of shortest-path betweenness centrality and their generic computation.*, Social Networks **30** (2008), no. 2, 136–145.
31. Ulrik Brandes and Daniel Fleischer, *Centrality measures based on current flow*, Proceedings of the 22Nd Annual Conference on Theoretical Aspects of Computer Science, STACS'05, Springer-Verlag, 2005, pp. 533–544.
32. Dirk Brockmann and Dirk Helbing, *The hidden geometry of complex, network-driven contagion phenomena*, Science **342** (2013), no. 6164, 1337–1342.
33. Kai Brgge, Asja Fischer, and Christian Igel, *The flip-the-state transition operator for restricted boltzmann machines*, Machine Learning **93** (2013), no. 1, 53–69 (English).
34. Ed Bullmore and Olaf Sporns, *Complex brain networks: graph theoretical analysis of structural and functional systems*, Nature Reviews Neuroscience **10** (2009), no. 3, 186–198.
35. Miguel A. Carreira-Perpinan and Geoffrey E. Hinton, *On contrastive divergence learning*, 10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS), 2005.
36. Liming Chen, J. Hoey, C.D. Nugent, D.J. Cook, and Zhiwen Yu, *Sensor-based activity recognition*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **42** (2012), no. 6, 790–808.
37. S. Chikkerur, V. Sundaram, M. Reisslein, and L.J. Karam, *Objective video quality assessment methods: A classification, review, and performance comparison*, Broadcasting, IEEE Transactions on **57** (2011), no. 2, 165–182.
38. M. Chincoli, A. A. Syed, D. C. Mocanu, and A. Liotta, *Predictive power control in wireless sensor networks*, 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), April 2016, pp. 309–312.
39. K. Cho, T. Raiko, and A. Ilin, *Parallel tempering is efficient for learning restricted boltzmann machines*, The 2010 International Joint Conference on Neural Networks (IJCNN), July 2010, pp. 1–8.
40. Franois Chollet, *keras*, 2015.
41. Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman, *Power-law distributions in empirical data*, SIAM Review **51** (2009), no. 4, 661–703.
42. Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo, *Distributed Optimization by Ant Colonies*, European Conference on Artificial Life, 1991, pp. 134–142.
43. OASIS XACML Technical Committee, 2013, eXtensible Access Control Markup Language (XACML).
44. Sean P. Cornelius, William L. Kath, and Adilson E. Motter, *Realistic control of network dynamics*, Nature Communications **4** (2013).
45. C. Cramer, E. Gelenbe, and P. Gelenbe, *Image and video compression*, Potentials, IEEE **17** (1998), no. 1, 29–33.

46. Paolo Crucitti, Vito Latora, and Sergio Porta, *Centrality measures in spatial networks of urban streets*, Phys. Rev. E **73** (2006), 036125.
47. G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals, and Systems (MCSS) **2** (1989), no. 4, 303–314.
48. Samuel A Danziger, S Joshua Swamidass, Jue Zeng, Lawrence R Dearth, Qiang Lu, Jonathan H Chen, Jianlin Cheng, Vinh P Hoang, Hiroto Saigo, Ray Luo, et al., *Functional census of mutation sequence spaces: the example of p53 cancer rescue mutants*, IEEE/ACM Transactions on Computational Biology and Bioinformatics **3** (2006), no. 2, 114–125.
49. Himansu Das, Bighnaraj Naik, Bibudendu Pati, and Chhabi Rani Panigrahi, *A survey on virtual sensor networks framework*, International Journal of Grid Distribution Computing **7** (2014), no. 5, 121–130.
50. Luisa de Vivo, Michele Bellesi, William Marshall, Eric A. Bushong, Mark H. Ellisman, Giulio Tononi, and Chiara Cirelli, *Ultrastructural evidence for synaptic scaling across the wake/sleep cycle*, Science **355** (2017), no. 6324, 507–510.
51. Charo I. Del Genio, Thilo Gross, and Kevin E. Bassler, *All scale-free networks are sparse*, Phys. Rev. Lett. **107** (2011), 178701.
52. Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau, *Parallel tempering for training of restricted boltzmann machines*, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, May 13-15, 2010, Chia Laguna Resort, Sardinia, Italy (Yee Whye Teh and Mike Titterington, eds.), 2010, pp. 145–152.
53. Sander Dieleman and Benjamin Schrauwen, *Accelerating sparse restricted boltzmann machine training using non-gaussianity measures*, Deep Learning and Unsupervised Feature Learning, Proceedings (Yoshua Bengio, James Bergstra, and Quoc Le, eds.), 2012, p. 9 (eng).
54. Graham H. Diering, Raja S. Nirujogi, Richard H. Roth, Paul F. Worley, Akhilesh Pandey, and Richard L. Huganir, *Homer1a drives homeostatic scaling-down of excitatory synapses during sleep*, Science **355** (2017), no. 6324, 511–515.
55. Agoston E. Eiben and J. E. Smith, *Introduction to evolutionary computing*, SpringerVerlag, 2003.
56. Albert Einstein, *Näherungsweise integration der feldgleichungen der gravitation*, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin), Seite 688-696. (1916).
57. Paul Erdős and Alfréd Rényi, *On random graphs i.*, Publicationes Mathematicae (Debrecen) **6** (1959), 290–297.
58. Brian Everitt, *The cambridge dictionary of statistics*, Cambridge University Press, Cambridge, UK; New York, 2002.
59. G. Exarchakos and N. Antonopoulos, *Cooperative stalking of transient nomadic resources on overlay networks*, Future Generation Computer Systems **29** (2013), no. 6, 1473–1484.
60. Alex Fornito, Andrew Zalesky, and Michael Breakspear, *The connectomics of brain disorders*, Nature Reviews Neuroscience **16** (2015), 159–172.
61. Santo Fortunato, *Community detection in graphs*, Physics Reports **486** (2010), no. 35, 75 – 174.
62. James A Foster, *Evolutionary computation*, Nature Reviews Genetics **2** (2001), no. 6, 428–436.
63. Linton C. Freeman, *Centrality in social networks conceptual clarification*, Social Networks **1** (1978), no. 3, 215 – 239.
64. S. Galzarano, R. Giannantonio, A. Liotta, and G. Fortino, *A task-oriented framework for networked wearable computing*, Automation Science and Engineering, IEEE Transactions on **PP** (2014), no. 99, 1–18.
65. Stefano Galzarano, Giancarlo Fortino, and Antonio Liotta, *A learning-based mac for energy efficient wireless sensor networks*, Internet and Distributed Computing Systems (Giancarlo Fortino, Giuseppe Di Fatta, Wenfeng Li, Sergio Ochoa, Alfredo Cuzzocrea, and Mukaddim Pathan, eds.), Lecture Notes in Computer Science, vol. 8729, Springer International Publishing, 2014, pp. 396–406 (English).

66. Juan Pablo Garella, José Joskowicz, Rafael Sotelo, Marcos Juayek, and Diego Durán, *Subjective video quality test: Methodology, database and experience*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2015.
67. Morrie Gasser, *Building a secure computer system*, Van Nostrand Reinhold Co., New York, NY, USA, 1988.
68. P. Gastaldo, S. Rovetta, and R. Zunino, *Objective quality assessment of mpeg-2 video streams by using cbp neural networks*, Neural Networks, IEEE Transactions on **13** (2002), no. 4, 939–947.
69. D. Geelen, G. van Kempen, F. van Hoogstraten, and A. Liotta, *A wireless mesh communication protocol for smart-metering*, Computing, Networking and Communications (ICNC), 2012 International Conference on, Jan 2012, pp. 343–349.
70. Peter V. Gehler, Alex D. Holub, and Max Welling, *The rate adapting poisson model for information retrieval and object recognition*, Proceedings of the 23rd International Conference on Machine Learning (New York, NY, USA), ICML '06, ACM, 2006, pp. 337–344.
71. E Gelenbe, *Stability of the random neural network model*, Neural Computation **2** (1990), no. 2, 239–247.
72. Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle, *MADE: masked autoencoder for distribution estimation*, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, 2015, pp. 881–889.
73. D. Ghadiyaram and A.C. Bovik, *Blind image quality assessment on real distorted images using deep belief nets*, Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on, Dec 2014, pp. 946–950.
74. Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, *A novel connectionist system for unconstrained handwriting recognition*, IEEE Trans. Pattern Anal. Mach. Intell. **31** (2009), no. 5, 855–868.
75. Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart, *Exploring network structure, dynamics, and function using networkx*, Proceedings of the 7th Python in Science Conference (Pasadena, CA USA) (Gaël Varoquaux, Travis Vaught, and Jarrod Millman, eds.), 2008, pp. 11 – 15.
76. S. L. Hakimi, *On realizability of a set of integers as degrees of the vertices of a linear graph. I*, J. Soc. Indust. Appl. Math. **10** (1962), 496–506.
77. Song Han, Jeff Pool, John Tran, and William Dally, *Learning both weights and connections for efficient neural network*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 1135–1143.
78. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.
79. G. E. Hinton, *A practical guide to training restricted boltzmann machines*, Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 7700, Springer Berlin Heidelberg, 2012, pp. 599–619.
80. G. E. Hinton and R. R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, Science **313** (2006), no. 5786, 504–507.
81. Geoffrey E. Hinton, *Training Products of Experts by Minimizing Contrastive Divergence*, Neural Computation **14** (2002), no. 8, 1771–1800.
82. Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, Neural Comput. **18** (2006), no. 7, 1527–1554.
83. GeoffreyE. Hinton, *A practical guide to training restricted boltzmann machines*, Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 7700, Springer, 2012, pp. 599–619 (English).
84. T. Hobfeld, R. Schatz, and S. Egger, *Sos: The mos is not enough!*, Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on, Sept 2011, pp. 131–136.
85. S. Hong and A. Dey, *Network analysis of cosmic structures: Network centrality and topological environment*, Monthly Notices of the Royal Astronomical Society **450** (2015), no. 2, 1999–2015.
86. S Huang and X. Zhao, *Redundant Nodes Elimination in Wireless Sensor Networks*, 4th International Conference on Information Computing and Applications (ICICA), 2013.
87. Jaime Iranzo, Javier M. Buldú, and Jacobo Aguirre, *Competition among networks highlights the power of the weak*, Nature Communications **7** (2016).

88. Herbert Jaeger, *Artificial intelligence: Deep neural reasoning*, Nature **538** (2016), no. 7626, 467–468.
89. Mathilde Jauzac, Eric Jullo, Jean-Paul Kneib, Harald Ebeling, Alexie Leauthaud, Cheng-Jiun Ma, Marceau Limousin, Richard Massey, and Johan Richard, *A weak lensing mass reconstruction of the large-scale filament feeding the massive galaxy cluster macsj0717.5+3745*, Monthly Notices of the Royal Astronomical Society **426** (2012), no. 4, 3369–3384.
90. H. Jeong, S. P. Mason, Albert-László Barabási, and Z. N. Oltvai, *Lethality and centrality in protein networks*, Nature **411** (2001), 41–42.
91. Nicola Jones, *Computer science: The learning machines*, Nature **505** (2014), no. 7482, 146–148.
92. J. Joskowicz and R. Sotelo, *A Model for Video Quality Assessment Considering Packet Loss for Broadcast Digital Television Coded in H.264*, International Journal of Digital Multimedia Broadcasting **2014** (2014), no. 5786, 11.
93. José Joskowicz, Rafael Sotelo, Marcos Juayek, Diego Durán, and Juan Pablo Garella, *Automation of subjective video quality measurements*, Proceedings of the Latin America Networking Conference on LANC 2014, ACM, 2014, pp. 7:1–7:5.
94. Shivaram Kalyanakrishnan and Peter Stone, *Batch reinforcement learning in a complex domain*, The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (New York, NY, USA), ACM, May 2007, pp. 650–657.
95. Evangelos Karapanos, Jean-Bernard Martens, and Marc Hassenzahl, *Accounting for diversity in subjective judgments*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI '09, ACM, 2009, pp. 639–648.
96. Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Sericola, and Gilles Trádan, *Second order centrality: Distributed assessment of nodes criticity in complex networks*, Computer Communications **34** (2011), no. 5, 619–628, Special Issue: Complex Networks.
97. M. Kitsak, L. K. L. K. Gallos, S. Havlin, F. Liljeros, L. H. Muchnik, E. Stanley, and H. A. Makse, *Identification of influential spreaders in complex networks*, Nature Physics **6** (2010), 888–893.
98. J. Zico Kolter and Matthew J. Johnson, *Redd: A public data set for energy disaggregation research*, in SustKDD, 2011.
99. B. Konuk, E. Zerman, G. Nur, and G.B. Akar, *A spatiotemporal no-reference video quality assessment model*, Image Processing (ICIP), 2013 20th IEEE International Conference on, Sept 2013, pp. 54–58.
100. Roshan Kotian, Georgios Exarchakos, Decebal Constantin Mocanu, and Antonio Liotta, *Predicting battery depletion of neighboring wireless sensor nodes*, Algorithms and Architectures for Parallel Processing, Lecture Notes in Computer Science, vol. 8286, Springer, 2013, pp. 276–284 (English).
101. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems 25, 2012, pp. 1097–1105.
102. S. Kumar Singh, M. P. Singh, and D. K. Singh, *Routing Protocols in Wireless Sensor Networks - A Survey*, International Journal of Computer Science & Engineering Survey (IJCSES) **1** (2010), no. 2.
103. Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman, *Building machines that learn and think like people*, Behavioral and Brain Sciences (2016), 1–101.
104. Hugo Larochelle and Yoshua Bengio, *Classification using discriminative restricted boltzmann machines*, Proceedings of the 25th International Conference on Machine Learning (New York, NY, USA), ICML '08, ACM, 2008, pp. 536–543.
105. Hugo Larochelle and Iain Murray, *The neural autoregressive distribution estimator.*, AISTATS, JMLR Proceedings, vol. 15, JMLR.org, 2011, pp. 29–37.
106. Jonathan Laserson, *From neural networks to deep learning: Zeroing in on the human brain*, XRDS **18** (2011), no. 1, 29–34.
107. Matthieu Latapy, Clmence Magnien, and Nathalie Del Vecchio, *Basic notions for the analysis of large two-mode networks*, Social Networks **30** (2008), no. 1, 31 – 48.
108. Glenn Lawyer, *Understanding the influence of all nodes in a network*, Scientific Reports **5** (2015).

109. P. Le Callet, C. Viard-Gaudin, and D. Barba, *A convolutional neural network approach for objective video quality assessment*, Neural Networks, IEEE Transactions on **17** (2006), no. 5, 1316–1327.
110. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.
111. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.
112. Heidi Ledford, *How to solve the world's biggest problems*, Nature **525** (2015), no. 7569, 308–311.
113. Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng, *Sparse deep belief net model for visual area v2*, Advances in Neural Information Processing Systems 20 (J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, eds.), Curran Associates, Inc., 2008, pp. 873–880.
114. Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng, *Unsupervised feature learning for audio classification using convolutional deep belief networks*, Advances in Neural Information Processing Systems 22, 2009, pp. 1096–1104.
115. Ji Li, Lachlan L.H. Andrew, Chuan Heng Foh, Moshe Zukerman, and Hsiao-Hwa Chen, *Connectivity, coverage and placement in wireless sensor networks*, Sensors **9** (2009), no. 10, 7664.
116. Long-Ji Lin, *Self-improving reactive agents based on reinforcement learning, planning and teaching*, Machine Learning **8** (1992), no. 3, 293–321.
117. Zhouhan Lin, Roland Memisevic, and Kishore Konda, *How far can we go without convolution: Improving fully-connected networks*, arXiv preprint arXiv:1511.02580 (2015).
118. A. Liotta, *The cognitive NET is coming*, IEEE Spectrum **50** (2013), no. 8, 26–31.
119. Antonio Liotta, Danil Geelen, Gert van Kempen, and Frans van Hoogstraten, *A survey on networks for smartmetering systems*, International Journal of Pervasive Computing and Communications **8** (2012), no. 1, 23–52.
120. Chuang Liu and Zi-Ke Zhang, *Information spreading on dynamic social networks*, Communications in Nonlinear Science and Numerical Simulation **19** (2014), no. 4, 896 – 904.
121. Xuxun Liu, *A survey on clustering routing protocols in wireless sensor networks.*, Sensors (Basel, Switzerland) **12** (2012), no. 8, 11113–53 (en).
122. Dietwig Lowet, Melvin Isken, Wei Pien Lee, Frank van Heesch, and Henk Eertink, *Robotic telepresence for 24/07 remote assistance to elderly at home, workshop on social robotic telepresence*, Ro-Man 2012, 21st IEEE International Symposium on Robot and Human Interactive Communication, 2012.
123. Dietwig Lowet and Frank van Heesch, *Florence - a multipurpose robotic platform to support elderly at home*, Workshop on Ambient Intelligence Infrastructures (WAmIi), Pisa, Italy., 2012.
124. Heng Luo, Ruimin Shen, Changyong Niu, and Carsten Ullrich, *Sparse group restricted boltzmann machines.*, AAAI (Wolfram Burgard and Dan Roth, eds.), AAAI Press, 2011.
125. David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson, *The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations*, Behavioral Ecology and Sociobiology **54** (2003), no. 4, 396–405.
126. S. Maleschlijski, G.H. Sendra, A. Di Fino, L. Leal-Taix, I. Thome, A. Terfort, N. Aldred, M. Grunze, A.S. Clare, B. Rosenhahn, and A. Rosenhahn, *Three dimensional tracking of exploratory behavior of barnacle cyprids using stereoscopy*, Biointerphases **7** (2012), no. 1-4 (English).
127. A. Manzanares, L. Martinez, M. Isken, D. Lowet, and A. Remazeilles, *User studies of a mobile assistance robot for supporting elderly: methodology and results*, Workshop at IROS 2012 on Assistance and Service robotics in a human environment, At Vila Moura, Portugal, 2012.
128. Benjamin M. Marlin, Kevin Swersky, Bo Chen, and Nando de Freitas, *Inductive principles for restricted boltzmann machine learning.*, AISTATS, JMLR Proceedings, vol. 9, JMLR.org, 2010, pp. 509–516.
129. J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, *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** (1995), 419–457.

130. Roland Memisevic and Geoffrey E. Hinton, *Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines*, Neural Computation **22** (2010), no. 6, 1473–1492.
131. Vlado Menkovski, Georgios Exarchakos, and Antonio Liotta, *The value of relative quality in video delivery*, J. Mob. Multimed. **7** (2011), no. 3, 151–162.
132. Melanie Mitchell, *An introduction to genetic algorithms*, MIT Press, Cambridge, MA, USA, 1998.
133. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, *Human-level control through deep reinforcement learning*, Nature **518** (2015), no. 7540, 529–533.
134. D. C. Mocanu, G. Exarchakos, and A. Liotta, *Node centrality awareness via swarming effects*, 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2014, pp. 19–24.
135. D. C. Mocanu, E. Mocanu, P. H. Nguyen, M. Gibescu, and A. Liotta, *Big iot data mining for real-time energy disaggregation in buildings*, 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct 2016, pp. 003765–003769.
136. D.C. Mocanu, G. Exarchakos, H.B. Ammar, and A. Liotta, *Reduced reference image quality assessment via boltzmann machines*, Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, May 2015, pp. 1278–1281.
137. D.C. Mocanu, G. Exarchakos, and A. Liotta, *Deep learning for objective quality assessment of 3d images*, Image Processing (ICIP), 2014 IEEE International Conference on, Oct 2014, pp. 758–762.
138. D.C. Mocanu, A. Liotta, A. Ricci, M.T. Vega, and G. Exarchakos, *When does lower bitrate give higher quality in modern video services?*, Network Operations and Management Symposium (NOMS), 2014 IEEE, May 2014, pp. 1–5.
139. D.C. Mocanu, E. Mocanu, P. Stone, P.H. Nguyen, M. Gibescu, and A. Liotta, *Evolutionary training of sparse artificial neural networks: A network science perspective*, Science (under review) (2017).
140. Decebal Constantin Mocanu, *On the synergy of network science and artificial intelligence*, Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, USA, July 9-15, 2016.
141. Decebal Constantin Mocanu, Haitham Bou Ammar, Dietwig Lowet, Kurt Driessens, Antonio Liotta, Gerhard Weiss, and Karl Tuyls, *Factored four way conditional restricted boltzmann machines for activity recognition*, Pattern Recognition Letters **66** (2015), 100 – 108, Pattern Recognition in Human Computer Interaction.
142. Decebal Constantin Mocanu, Haitham Bou-Ammar, Luis Puig, Eric Eaton, and Antonio Liotta, *Estimating 3d trajectories from 2d projections via disjunctive factored four-way conditional restricted boltzmann machines*, Pattern Recognition (2017).
143. Decebal Constantin Mocanu, Georgios Exarchakos, and Antonio Liotta, *Decentralized dynamic understanding of hidden relations in complex networks*, Nature Communications (under review) (2017).
144. Decebal Constantin Mocanu, Elena Mocanu, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta, *A topological insight into restricted boltzmann machines*, Machine Learning **104** (2016), no. 2, 243–270.
145. Decebal Constantin Mocanu, Jeevan Pokhrel, Juan Pablo Garella, Janne Seppnen, Eirini Liotou, and Manish Narwaria, *No-reference video quality measurement: added value of machine learning*, Journal of Electronic Imaging **24** (2015), no. 6, 061208.
146. Decebal Constantin Mocanu, Fatih Turkmen, and Antonio Liotta, *Towards abac policy mining from logs with deep learning*, 18th International Multiconference, IS 2015, Intelligent Systems, Ljubljana, Slovenia, 2015.
147. Decebal Constantin Mocanu, Maria Torres Vega, Eric Eaton, Peter Stone, and Antonio Liotta, *Online contrastive divergence with generative replay: Experience replay without storing data*, CoRR **abs/1610.05555** (2016).
148. Decebal Constantin Mocanu, Maria Torres Vega, and Antonio Liotta, *Redundancy reduction in wireless sensor networks via centrality metrics*, 2015 IEEE International Conference on Data Mining Workshop (ICDMW) **00** (2015), 501–507.

149. E. Mocanu, D.C. Mocanu, H.B. Ammar, Z. Zivkovic, A. Liotta, and E. Smirnov, *Inexpensive user tracking using boltzmann machines*, Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on, Oct 2014, pp. 1–6.
150. E. Mocanu, D.C. Mocanu, P.H. Nguyen, A. Liotta, M. Webber, M. Gibescu, and J.G. Slootweg, *On-line building energy optimization using deep reinforcement learning*, IEEE Transactions on Smart Grid (under review) (2017).
151. E. Mocanu, P. H. Nguyen, and M. Gibescu, *Energy disaggregation for real-time building flexibility detection*, 2016 IEEE Power and Energy Society General Meeting (PESGM), July 2016, pp. 1–5.
152. Elena Mocanu, Phuong H. Nguyen, Madeleine Gibescu, and Wil L. Kling, *Deep learning for estimating building energy consumption*, Sustainable Energy, Grids and Networks **6** (2016), 91 – 99.
153. Elena Mocanu, Phuong H. Nguyen, Wil L. Kling, and Madeleine Gibescu, *Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning*, Energy and Buildings **116** (2016), 646 – 655.
154. S. Mohamed and G. Rubino, *A study of real-time packet video quality using random neural networks*, Circuits and Systems for Video Technology, IEEE Transactions on **12** (2002), no. 12, 1071–1083.
155. Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay, *Language understanding for text-based games using deep reinforcement learning.*, CoRR **abs/1506.08941** (2015).
156. M. E. J. Newman, *The structure of scientific collaboration networks*, Proceedings of the National Academy of Sciences of the United States of America **98** (2001), no. 2, 404–409.
157. Mark E. J. Newman, *Networks: An introduction*, Oxford University Press, 2010.
158. Mark EJ Newman, Steven H Strogatz, and Duncan J Watts, *Random graphs with arbitrary degree distributions and their applications*, Physical review E **64** (2001), no. 2, 026118.
159. M.E. J. Newman, *A measure of betweenness centrality based on random walks*, Social Networks **27** (2005), no. 1, 39–54.
160. Regina Nuzzo, *Scientific method: Statistical errors*, Nature **506** (2014), no. 7487, 150–152.
161. F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy., *Berkeley mhad: A comprehensive multimodal human action database*, Proceedings of the IEEE Workshop on Applications on Computer Vision (WACV), 2013.
162. Alexander G. Ororbia, C. Lee Giles, and David Reitter, *Online semi-supervised learning with deep hybrid boltzmann machines and denoising autoencoders*, CoRR **abs/1511.06964** (2015).
163. Takayuki Osogami and Makoto Otsuka, *Restricted boltzmann machines modeling human choice*, Advances in Neural Information Processing Systems 27, 2014, pp. 73–81.
164. Shumao Ou, Kun Yang, A. Liotta, and Liang Hu, *Performance analysis of offloading systems in mobile wireless environments*, Communications, 2007. ICC '07. IEEE International Conference on, June 2007, pp. 1821–1826.
165. Stéphane Péchard, Romuald Pépion, and Patrick Le Callet, *Suitable methodology in subjective video quality assessment: a resolution dependent paradigm*, International Workshop on Image Media Quality and its Applications, IMQA2008, 2008, p. 6.
166. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.
167. Sen Pei and Hernán A Makse, *Spreading dynamics in complex networks*, Journal of Statistical Mechanics: Theory and Experiment **2013** (2013), no. 12, P12002.
168. Luiz Pessoa, *Understanding brain networks and brain organization*, Physics of Life Reviews **11** (2014), no. 3, 400 – 435.
169. Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon, *Overlapping community detection using bayesian non-negative matrix factorization*, Phys. Rev. E **83** (2011), 066114.
170. NadiaN. Qadri and Antonio Liotta, *Analysis of pervasive mobile ad hoc routing protocols*, Pervasive Computing (Aboul-Ella Hassanien, Jemal H. Abawajy, Ajith Abraham, and Hani Hagras, eds.), Computer Communications and Networks, Springer London, 2010, pp. 433–453 (English).
171. Sen Qiu, Yu Yang, Jijian Hou, Ran Ji, Huosheng Hu, and Zhelong Wang, *Ambulatory estimation of 3d walking trajectory and knee joint angle using marg sensors*, Innovative

Computing Technology (INTECH), 2014 Fourth International Conference on, Aug 2014, pp. 191–196.
172. Marc Aurelio Ranzato, Y lan Boureau, and Yann L. Cun, *Sparse feature learning for deep belief networks*, Advances in Neural Information Processing Systems 20 (J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, eds.), Curran Associates, Inc., 2008, pp. 1185–1192.
173. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, 1962.
174. D.E. Rumelhart, G.E. Hintont, and R.J. Williams, *Learning representations by back-propagating errors*, Nature **323** (1986), no. 6088, 533–536.
175. S. Saavedra, Daniel B. Stouffer, B. Uzzi, and Jordi Bascompte, *Strong contributors to network persistence are the most vulnerable to extinction*, Nature **478** (2011), 233–235.
176. Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton, *Restricted boltzmann machines for collaborative filtering*, Proceedings of the 24th International Conference on Machine Learning, ICML '07, ACM, 2007, pp. 791–798.
177. Ruslan Salakhutdinov and Iain Murray, *On the quantitative analysis of deep belief networks*, In Proceedings of the International Conference on Machine Learning, 2008, pp. 872–879.
178. A. L. Samuel, *Some studies in machine learning using the game of checkers*, IBM J. Res. Dev. **3** (1959), no. 3, 210–229.
179. Alan G. Sanfey, *Social decision-making: Insights from game theory and neuroscience*, Science **318** (2007), no. 5850, 598–602.
180. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, *Prioritized experience replay*, CoRR **abs/1511.05952** (2015).
181. Muhammad Shahid, Andreas Rossholm, Benny Lovstrom, and Hans-Jurgen Zepernick, *No-reference image and video quality assessment: a classification and review of recent approaches*, EURASIP Journal on Image and Video Processing **2014** (2014), no. 1, 40.
182. Hugo Silva, Andr Dias, Jos Almeida, Alfredo Martins, and Eduardo Silva, *Real-time 3d ball trajectory estimation for robocup middle size league using a single camera*, RoboCup 2011: Robot Soccer World Cup XV (Thomas Rfer, N.Michael Mayer, Jesus Savage, and Uluc Saranl, eds.), Lecture Notes in Computer Science, vol. 7416, Springer Berlin Heidelberg, 2012, pp. 586–597 (English).
183. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., *Mastering the game of go with deep neural networks and tree search*, Nature **529** (2016), no. 7587, 484–489.
184. K.D. Singh, Y. Hadjadj-Aoul, and G. Rubino, *Quality of experience estimation for adaptive http/tcp video streaming using h.264/avc*, Consumer Communications and Networking Conference (CCNC), 2012 IEEE, Jan 2012, pp. 127–131.
185. K.D. Singh and G. Rubino, *Quality of experience estimation using frame loss pattern and video encoding characteristics in dvb-h networks*, Packet Video Workshop (PV), 2010 18th International, Dec 2010, pp. 150–157.
186. P. Smolensky, *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1*, MIT Press, Cambridge, MA, USA, 1986, pp. 194–281.
187. P. Smolensky, *Information processing in dynamical systems: Foundations of harmony theory*, Parallel Distributed Processing: Volume 1: Foundations (D. E. Rumelhart, J. L. McClelland, et al., eds.), MIT Press, Cambridge, 1987, pp. 194–281.
188. J. Sogaard, S. Forchhammer, and J. Korhonen, *No-reference video quality assessment using codec analysis*, IEEE Transactions on Circuits and Systems for Video Technology **25** (2015), no. 10, 1637–1650.
189. Nitish Srivastava and Ruslan R Salakhutdinov, *Multimodal learning with deep boltzmann machines*, Advances in Neural Information Processing Systems 25 (F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, eds.), Curran Associates, Inc., 2012, pp. 2222–2230.
190. N. Staelens, D. Deschrijver, E. Vladislavleva, B. Vermeulen, T. Dhaene, and P. Demeester, *Constructing a no-reference h.264/avc bitstream-based video quality metric using genetic programming-based symbolic regression*, Circuits and Systems for Video Technology, IEEE Transactions on **23** (2013), no. 8, 1322–1333.
191. Cornelis J. Stam, *Modern network science of neurological disorders*, Nature Reviews Neuroscience **15** (2014), 683–685.

192. Steven H. Strogatz, *Exploring complex networks*, Nature **410** (2001), 268–276.
193. Richard S. Sutton and Andrew G. Barto, *Introduction to reinforcement learning*, 1st ed., MIT Press, Cambridge, MA, USA, 1998.
194. Richard S. Sutton, Csaba Szepesvri, Alborz Geramifard, and Michael Bowling, *Dyna-style planning with linear function approximation and prioritized sweeping*, Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, 2008.
195. Kevin Swersky, Daniel Tarlow, Ilya Sutskever, Ruslan Salakhutdinov, Richard S. Zemel, and Ryan P. Adams, *Cardinality restricted boltzmann machines.*, NIPS, 2012, pp. 3302–3310.
196. Huixuan Tang, N. Joshi, and A. Kapoor, *Blind image quality assessment using semi-supervised rectifier networks*, Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, June 2014, pp. 2877–2884.
197. Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis, *Two distributed-state models for generating high-dimensional time series*, Journal of Machine Learning Research **12** (2011), 1025–1068.
198. Matthew E. Taylor and Peter Stone, *Transfer learning for reinforcement learning domains: A survey*, J. Mach. Learn. Res. **10** (2009), 1633–1685.
199. Matthew E. Taylor, Peter Stone, and Yaxin Liu, *Transfer learning via inter-task mappings for temporal difference learning*, Journal of Machine Learning Research **8** (2007), no. 1, 2125–2167.
200. Tijmen Tieleman, *Training restricted boltzmann machines using approximations to the likelihood gradient*, Proceedings of the 25th International Conference on Machine Learning (New York, NY, USA), ICML '08, ACM, 2008, pp. 1064–1071.
201. Tijmen Tieleman and Geoffrey Hinton, *Using fast weights to improve persistent contrastive divergence*, Proceedings of the 26th Annual International Conference on Machine Learning (New York, NY, USA), ICML '09, ACM, 2009, pp. 1033–1040.
202. Fatih Turkmen, Jerry den Hartog, Silvio Ranise, and Nicola Zannone, *Analysis of XACML policies with SMT*, 4th International Conference on Principles of Security and Trust - (POST) Proceedings, 2015, pp. 115–134.
203. Fatih Turkmen, Simon N. Foley, Barry O'Sullivan, William M. Fitzgerald, Tarik Hadzic, Stylianos Basagiannis, and Menouer Boubekeur, *Explanations and relaxations for policy conflicts in physical access control*, 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013, 2013, pp. 330–336.
204. Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson, *Do deep convolutional nets really need to be deep and convolutional?*, arXiv preprint arXiv:1603.05691 (2016).
205. Remco van der Hofstad, *Random graphs and complex networks vol. i*, 2016, pp. 14–15.
206. M. Varela, T. Maki, L. Skorin-Kapov, and T. Hossfeld, *Towards an understanding of visual appeal in website design*, Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on, July 2013, pp. 70–75.
207. M. T. Vega, D. C. Mocanu, R. Barresi, G. Fortino, and A. Liotta, *Cognitive streaming on android devices*, 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2015, pp. 1316–1321.
208. M. T. Vega, D. C. Mocanu, J. Famaey, S. Stavrou, and A. Liotta, *Deep learning for quality assessment in live video streaming*, IEEE Signal Processing Letters **24** (2017), no. 6, 736–740.
209. Maria Torres Vega, Mocanu Decebal Constantin, and Antonio Liotta, *Unsupervised deep learning for real-time assessment of video streaming services*, Multimedia Tools and Applications (2017).
210. Maria Torres Vega, Emanuele Giordano, Decebal Constantin Mocanu, Dian Tjondronegoro, and Antonio Liotta, *Cognitive no-reference video quality assessment for mobile streaming services.*, QoMEX, IEEE, 2015, pp. 1–6.
211. Maria Torres Vega, Decebal Constantin Mocanu, and Antonio Liotta, *A regression method for real-time video quality evaluation*, Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media (New York, NY, USA), MoMM '16, ACM, 2016, pp. 217–224.

212. Maria Torres Vega, Decebal Constantin Mocanu, Stavros Stavrou, and Antonio Liotta, *Predictive no-reference assessment of video quality*, Signal Processing: Image Communication **52** (2017), 20 – 32.
213. Maria Torres Vega, Vittorio Sguazzo, Decebal Constantin Mocanu, and Antonio Liotta, *Accuracy of no-reference quality metrics in network-impaired video streams*, Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia (New York, NY, USA), MoMM 2015, ACM, 2015, pp. 326–333.
214. M.T. Vega, Shihuan Zou, D.C. Mocanu, E. Tangdiongga, A.M.J. Koonen, and A. Liotta, *End-to-end performance evaluation in high-speed wireless networks*, Network and Service Management (CNSM), 2014 10th International Conference on, Nov 2014, pp. 344–347.
215. VQEG, *Final report from the video quality experts group on the validation of objective models of video quality assessment*, 2003.
216. Cheng Wan, Xiaoming Jin, Guiguang Ding, and Dou Shen, *Gaussian cardinality restricted boltzmann machines*, Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
217. Pu Wang, Marta C. González, César A. Hidalgo, and Albert-László Barabási, *Understanding the spreading patterns of mobile phone viruses*, Science **324** (2009), no. 5930, 1071–1076.
218. Duncan J Watts and Steven H Strogatz, *Collective dynamics of 'small-world' networks*, Nature **393** (1998), 440–442.
219. K. Wehmuth and A. Ziviani, *Daccer: Distributed assessment of the closeness centrality ranking in complex networks*, Computer Networks **57** (2013), no. 13, 2536–2548.
220. Max Welling, Michal Rosen-zvi, and Geoffrey E. Hinton, *Exponential family harmoniums with an application to information retrieval*, Advances in Neural Information Processing Systems 17 (L.K. Saul, Y. Weiss, and L. Bottou, eds.), MIT Press, 2005, pp. 1481–1488.
221. Shimon Whiteson and Peter Stone, *Evolutionary function approximation for reinforcement learning*, Journal of Machine Learning Research **7** (2006), 877–917.
222. Stefan Winkler and Praveen Mohandas, *The evolution of video quality measurement: From PSNR to hybrid metrics*, IEEE Transactions on Broadcasting **54** (2008), no. 3, 660–668.
223. T. A. Witten and L. M. Sander, *Diffusion-limited aggregation, a kinetic critical phenomenon*, Phys. Rev. Lett. **47** (1981), 1400–1403.
224. S. Wuchty and P. Uetz, *Protein-protein interaction networks of e. coli and s. cerevisiae are similar*, Scientific Reports **4** (2014), no. 7187.
225. Jingtao Xu, Peng Ye, Yong Liu, and D. Doermann, *No-reference video quality assessment via feature learning*, Image Processing (ICIP), 2014 IEEE International Conference on, Oct 2014, pp. 491–495.
226. Zhongyuan Xu and Scott D. Stoller, *Mining attribute-based access control policies from logs*, Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings, 2014, pp. 276–291.
227. Jason Yosinski and Hod Lipson, *Visually debugging restricted boltzmann machine training with a 3d example*, Representation Learning Workshop, 29th International Conference on Machine Learning, 2012.
228. S. Zhang, M. H. Ang, W. Xiao, and C. K. Tham, *Detection of activities for daily life surveillance: Eating and drinking*, HealthCom 2008 - 10th International Conference on e-health Networking, Applications and Services, IEEE, July 2008, pp. 171–176.
229. Fang Zhou, Q. Claire, and R.D. King, *Predicting the geographical origin of music*, Data Mining (ICDM), 2014 IEEE International Conference on, Dec 2014, pp. 1115–1120.
230. K. Zhu, C. Li, V. Asari, and D. Saupe, *No-reference video quality assessment based on artifact measurement and statistical analysis*, Circuits and Systems for Video Technology, IEEE Transactions on **PP** (2014), no. 99, 1–1.

APPENDIX A

Sparsity in deep neural networks: a video quality assessment study case

Besides offering a practical example on how artificial intelligence methods can be used to solve real-world problems, this appendix serves two purposes:

(1) To exemplify the need of scalable Artificial Neural Networks (ANNs) in a difficult practical scenario, such as estimating the video quality perceived by the user during video broadcast transmissions over Digital Terrestrial Television (DTT). In this context, we are not able to directly use the video frames pixels as inputs to the ANN model, because these are too many (i.e. millions for each frame). To overcome this situation we utilize an alternative solution, well known in the machine learning community, and we manually select some features to represent the videos. On one side, this feature selection procedure has the disadvantage of being difficult to be automated as it needs the knowledge of human experts. On the other hand, it loses some important properties of the videos which in the end leads to lower performance.

(2) To highlight that the fully connected ANNs reach an interesting sparsity pattern of neurons connectivity after the training process, as it can be seen in Figure A.7 and briefly discussed in Section A.5.3. This sparse connectivity obtained after the training process, made us to question: "Why should we not try to create ANN models which are initialized with a sparse connectivity to be able to create larger ANN models with faster training and exploitation computational times?". This question led us to breakthroughs presented in Chapters 4 and 5.

A.1. Introduction

With the ever increasing demand for video services and applications, real-time video processing is one of the central issues in multimedia processing systems. Given the practical limitations in terms of resources (bandwidth, computational power, memory etc.), video signals need to be appropriately processed (e.g. compressed) to make them more suitable for transmission, storage and subsequent rendering. However, most of the mentioned processing will degrade the visual quality to varying extents. As a consequence, the end user may view a significantly modified video signal in comparison to the original source content. It is, therefore, important to measure the quality of the processed video signal and benchmark

This appendix is integrally based on: D.C. Mocanu, J. Pokhrel, J. Pablo Garella, J. Seppänen, E. Liotou, M. Narwaria: *No-reference Video Quality Measurement: Added Value of Machine Learning*, Journal of Electronic Imaging 24 (2015), no. 6, 061208.

Table A.1 – Description of video QoE objective estimation categories.

	Full-Reference (FR)	**Reduced-Reference (RR)**	**No-Reference (NR)**
Reference video	The reference video is available.	Only some information (e.g. metrics) extracted from the reference video are required	No information from the reference video is required.
Methodology	The quality is estimated based on a comparison between the reference and a processed video.	The quality is estimated based based on the information extracted from the reference video and a processed video.	The quality is estimated just on some information extracted from a processed video.
Accuracy (in general)	Higher than RR and NR.	Higher than NR. Lower than FR.	Lower than FR and RR.

the performance of different video processing algorithms in terms of video quality assessment. Video quality is essentially a component of the larger concept of Quality of Experience (QoE). It is therefore an intrinsically subjective measure and can depend on multiple factors including degree of annoyance (related to artifact visibility), aesthetics, emotions, past experience etc. [**2**]. Thus, subjective viewing tests remain the most reliable and accurate methods, given appropriate laboratory conditions and a sufficiently large subject panel. However, subjective assessment may not be feasible in certain situations (e.g. real-time video compression, transmission), and an objective approach is more suitable in such scenarios. While the performance of objective approaches may not accurately mimic the subjective opinion, it can still potentially provide approximate and relative estimates of video quality, in a given application.

Objective quality estimation can be classified into three groups, i.e. Full-Reference (FR), Reduced-Reference (RR) and No-Reference (NR) [**37**] , as detailed in Table A.1. Among them, NR estimation is more challenging since it relies only on the processed signal. As a result, it is more related to detection and quantification of certain irregularities or absence of specific features which would be typically found in the reference video. It can also exploit application-specific features (e.g. bit rate) from the video bit stream in order to quantify quality, and there are existing works to this end, as discussed in the next section. Subjective estimation of video quality, on the other hand, involves a number of human observers rating the video quality on a fixed pre-defined scale, typically in controlled laboratory conditions. Excellent treatment of the various factors in video quality assessment is readily available in the form of standards and recommended practices [**1**].

An important aspect of any subjective study is the underlying variability in the collected ratings. This happens because the same stimuli typically do not receive the same rating by all the observers. This is of course expected since the notion of video quality is highly subjective, and this injects certain variability or inter-observer differences in the stimuli rating. While these are generally reported in subjective studies (in the form of standard deviations, confidence intervals etc.), a survey of literature reveals that they are not typically accounted for in objective quality prediction. As a result, a majority of works on objective quality estimation focus only on predicting a single score that may represent an average of all the

ratings per stimuli. Further, the prediction accuracies of objective methods are generally based on how close the objective scores are to the averaged subjective ratings (this is generally quantified by correlation coefficients, mean square error, scatter plots, etc.). However, the inherent subjective variability and its impact are not directly taken into account. This may potentially reduce the reliability of the objective estimates especially when there is larger disagreement (high variability) among subjects on the quality of a certain stimuli. Therefore, the aim of this appendix is to analyze this issue in more details, and subsequently present a NR video quality assessment method based on that. The presented approach is based on defining a reasonable measure of subjective data diversity and modeling it through the paradigm of machine learning.

The remainder appendix is organized as follows. Section A.2 first provides a brief review of machine learning based NR video quality measurement methods, and also outlines their limitations. We also present our contributions in this section. An analysis of the importance of diversity in subjective rating processes is presented in Section A.3. The proposed method and its application within a practical scenario are explained in Section A.4, and experimental verified in Section A.5. The next section presents relevant discussions about the results, while section A.7 draws conclusions.

A.2. Background and motivation

A.2.1. Previous work

Even though the research in NR video quality assessment is more than a decade old, we are still far from a general purpose NR quality indicator that can accurately predict video quality in all situations. The authors in [**68**] presented one of the first comprehensive method for estimating video quality based on neural networks. In this work, a methodology using Circular Back Propagation (CBP) neural networks is used for the objective quality assessment of motion picture expert group (MPEG) video streams. The work in [**109**] employed Convolutional Neural Networks (CNN) in order to estimate video quality. It differs from conventional neural network approach since it relies on the use of CNNs that allows a continuous time scoring of the video. A NR method was presented in [**225**], which is based on mapping frame level features into a spatial quality score followed by temporal pooling. The method developed in [**230**] is based on features extracted from the analysis of discrete cosine transform (DCT) coefficients of each decoded frame in a video sequence, and objective video quality was predicted using a neural network. Another NR video quality estimator was presented in [**190**], where symbolic regression based framework was trained on a set of features extracted from the received video bit-stream. Another recent method in [**188**] works on the similar principle of analyzing several features. These are based on distinguishing the type of codec used (MPEG or H.264/AVC), analysis of DCT coefficients, estimation of the level of quantization used in the I-frames etc. The next step is to apply Support Vector Regression (SVR) to predict video quality in NR fashion. The NR method proposed in [**99**] was based on polynomial regression model, where the independent variables (or features) were based on spatial and temporal quantities derived from video spatio-temporal complexity, bit rate and packet loss measurements. The works mentioned here by no means constitute the entire list

of contributions on the topic of NR video quality measurement but merely represent the most recent and relevant for the purpose of this appendix. The reader is encouraged to refer to survey papers, for example [**181**].

A.2.2. Limitations of existing methods

As mentioned, there has already been significant research work on NR video quality estimation especially for video compression applications. However, most of these methods share three common limitations related to their design and validation as enlisted below:

- Most of these methods rely only on mean opinion scores (MOS) or degradation MOS (DMOS) both for training and validation. This, to our mind is problematic since the MOS or DMOS (obtained by averaging raw scores for each observer) tend to neglect the variability inherently present in the subjective rating process.
- Most of these methods have been validated only on limited set of videos and lacked a comprehensive method evaluation from the viewpoint of its robustness to untrained content.
- Lastly, a majority of existing work focus only on video compression. Thus, they would be limited in their applicability to other applications (e.g. video transmission) where the fully decoded video content may not be available and so quality must be predicted only from the bit stream information.

A.2.3. Our contributions

In this appendix, we aim to address the limitations mentioned above. Thus, our main contribution is to perform statistical analysis on the performance of various machine learning methods (e.g. linear regression [**78**], decision trees for regression [**78**], random neural networks [**154**], deep belief networks [**20**]) in predicting video quality on a real-world database [**66**]. More specifically, in contrast to most of the existing works on NR video quality estimation, we focus on three aspects that have been largely ignored.

First, we model the diversity that inevitably exists in any subjective rating process, and we analyze statistically its relation with MOS. Thus, we attempt to take into account inter-observer differences since it will help in a better interpretation of how reliable the objective quality score is and what it conveys about the user satisfaction levels. Such an approach also adds significant value from a business perspective when it comes to telecom operators or internet service providers (ISPs), as will be further analyzed in the next section. Thus, in the proposed approach, we do not just train our method in an effort to maximize correlations with the average ground truth, but simultaneously allow the algorithm to learn the associated data variability. To our knowledge, this is the first work towards the design of an application-specific NR video quality estimator, which can provide additional output that can help to understand the meaning of the objective score under a given application scenario. The presented analysis will be therefore

of interest to the QoE community, which has largely focused only on MOS as the indicator of subjective video quality.

Secondly, we exploit the promising deep learning framework in our method and demonstrate its suitability for the said task, while we assess its prediction performance against three widely used machine learning algorithms and two statistical methods. Specifically, deep networks can benefit from unsupervised learning thus requiring less training data in comparison to the traditional learning methods. An analysis pertaining to the training of the deep networks weights is also presented to provide insights into the training process.

Finally, we focus on meaningful verification of the proposed method on several challenging video clips within the practical framework of DTT, which help to evaluate the proposed method against diverse content and distortion severities. We highlight that half of the video clips used for experiments (i.e. 200) come from a real-world video delivery chain with impairments produced by a real video transmission system and not produced by noise added artificially, thus representing a realistic scenario.

A.3. Exploring diversity in subjective viewing tests

It can be seen that a vast majority of objective studies rely only on the mean or average (MOS or DMOS) of the individual observer ratings. As we know, simple arithmetic mean is a measure of the central tendency but it tends to ignore the dispersion of the data. Expectedly, simple averaged based ratings have been contested in literature as they result in an information loss of how opinions of subjective assessment participants deviate from each other. The authors of [**95**] argue against averaging subjective judgments and suggest that taking into account the diversity of subjective views increases the information extracted from the given dataset. Authors of [**84**] apply this principle in their QoE study, where in addition to MOS a standard deviation of opinion scores (SOS) is studied. The mathematical relation between MOS and SOS is defined, and several databases for various applications are analyzed using SOS in addition to average user ratings.

A.3.1. Scattering of subjective opinions

The subjective tests remain the most reliable[1] approach to assess human factors such as degree of enjoyment (video quality). Still, expectedly some amount of inherent subjectivity will always be injected into the data collected from such studies. This can be attributed to several factors including the viewing strategy (some observers make decisions instinctively based on abstract video features while others may base their decision on more detailed scene analysis), emotions, past experience etc. For video quality evaluation, this means that while the individual observer ratings may indicate a general trend about perceived quality, they may still differ/disagree on the magnitude of such an agreement. Such diversity can provide valuable information that can be exploited for specific applications.

[1]Assuming these tests are conducted in proper viewing conditions (controlled lighting, well defined viewing distance/angles etc. for the considered application) and with a sufficiently large subject panel.

However, before that, it is necessary to quantify the said diversity (scattering) meaningfully and not merely rely on averaged measures such as MOS.

The deviation of individual ratings from the mean can for instance provide a measure of the spread i.e. standard deviation (SOS). Another related measure is the confidence interval which is derived from standard deviation and also depends on the number of observers. These have been often reported in subjective studies involving video quality measurement. But using these measures to supplement for objective quality prediction is not always interpretable in a stand alone fashion. For example, simply providing a standard deviation along with a predicted objective score does not allow a clear interpretation of what it may mean in the context of an application. This is partly due to the mathematical relation between MOS and standard deviation (high or low MOS always results in small deviation), and also because standard deviation does not indicate skewness of opinions scattered around the average value. Hence, it may be desirable to devise a more interpretable measure of quantifying the diversity of subjective opinion and more importantly what it may mean in the context of a particular application.

A.3.2. A new measure to quantify subjective uncertainty

It is known that low MOS for a given service indicates bad quality and therefore disappointment to the service, but even if MOS is high, we cannot know from this single value how many users are actually dissatisfied with the service. Moreover, not only do negative experiences affect customers more than positive experiences, but customers are also prone to share their negative experiences more likely than positive ones. Therefore we could see a negative experience of a single user to have a risk of avalanche where the negative experience is spread to several other current/potential customers who will see the service in a more negative light than before, without actually having bad experience with the service. As already highlighted, a majority of objective methods simply ignore the diversity of user opinions, and instead focus only on average ratings as their target. To overcome this, we first need to define a plausible way in order to exploit data uncertainty so that it adds value to the objective quality prediction. To that end, we studied various methods for expressing the diversity, and considering a business-oriented application, we found that an appropriate measure of profitability (which is of course the key goal of any business) can be derived from the answer to question "how many users are unsatisfied with the service". From service management and business point, satisfied users are less interesting than dissatisfied users. This is due to the fact that, from quality perspective, satisfied users require no quality management for their service (although this is not to say that satisfied users should not be considered at all in overall service marketing).

MOS is a straightforward indicator for expressing the opinion of a majority of users, but as discussed, this is hardly enough if we want to maintain service reputation and hold on to the current customer base. Therefore we introduce a new indicator along with MOS - Percentage of Dissatisfied Users (PDU) against MOS. It indicates the percentage of users who would give an opinion score less than certain threshold given a certain MOS score, i.e.

Table A.2 – MOS scores provided by 25 observers to a particular video clip.

2	5	4	4	4
4	3	3	2	2
3	2	4	2	4
3	5	3	5	4
4	3	5	2	5

$$PDU = \frac{\#(OS < th)}{N} \times 100 \tag{A.1}$$

where OS denotes the opinion score from an individual observer, th is the user-defined threshold, N is the total number of observers evaluating the given condition (service quality).

As an example, let us consider that 3 independent and random observers evaluated a sample (video stimulus) and gave scores 2, 5, and 5 (on a scale from 1 to 5, 5 denotes excellent quality). We can quickly calculate the MOS for this sample as 4, which is a fairly good score considering the defined scale of evaluation in this case. But we note that one individual gave a score of 2, which is very poor. Consequently, we can conclude that 33% of users were not satisfied (i.e. $PDU = 33\%$) with this sample, despite the MOS being high. It is therefore easy to realize the limitation of average based ratings (even with this somewhat limited example) where the MOS would conceal the fact that not all users were happy with the sample (despite a high MOS). We can also observe such effects on real subjective data shown in Table A.2. It represents the individual subjective opinion scores of 25 observers (this was as part of a subjective study conducted in our lab) for a processed video. We note that the mean of these individual ratings is 3.48 which is in the higher range (the scale of rating was from 1 to 5), and may lead to conclude that the video quality would be generally at least acceptable. Still, we note that $PDU = 24\%$ (when mean is considered as th) meaning that almost one-fourth of the customers/observers were dissatisfied with the video quality. This information should then be used to devise corrective actions. It can also be seen that the definition of PDU depends on the free parameter th, and hence it can be set by the service provider. This would depend on what quality level is considered intolerable and the actions required to avoid customer churn. In this appendix, we selected a value of 3, i.e. $th = 3$ (assuming a scale from 1 to 5), but especially for commercial applications where customers pay a monthly fee or pay per view, this number could be even higher. Hence, it can be customized.

Before we conclude this section, it is important to mention that the proposed measure PDU may not always be a function of MOS nor it may be directly related to standard deviation of the individual subjective ratings. So one cannot assume that a higher MOS will imply lower PDU or a lower MOS always implies a larger PDU. The reason is that different quality degradations may have different impacts on the consistency of user opinions. We can easily understand this with our previous example, where scores 2, 5, and 5 lead to a MOS of 4. However, we may have the same MOS in another situation. For instance if the scores were 4, 4, and 4, the resultant MOS would still be 4 but $PDU = 0$ in this case. Also, standard deviation may not be a substitute for PDU for two reasons. First, as already

stated the former may not be interpretable in a stand alone manner. Second, standard deviation can be similar for two very different MOSs in which case it does not provide any information on possible corrective measures. In contrast, similar PDU for two different MOSs may indicate a course correction (if PDU is high) irrespective of the MOS.

A.4. Application in no reference video quality estimation

In this section we demonstrate the practical utility of the proposed method in a NR scenario, within the framework of Digital Terrestrial Television. The proposed method follows similar design philosophy as some of the existing methods but there are some important differences that add value to our proposal. First, we exploit the framework of deep learning methods, which to our knowledge has not been exploited towards NR video quality measurement. Specifically, in the considered application, it is assumed that source video data is not available and quality needs to be predicted only from coded stream information. Secondly, our method is trained to provide PDU values in addition to objective quality. This allows the user to better interpret the reliability of the objective prediction especially from the viewpoint of satisfied/dissatisfied user percentage.

A block diagram of the proposed approach is shown in Figure A.1. Note that in the DTT scenario there can be multiple TV channels broadcasting signals over the air and these signals are pre-processed (source and channel coded) before transmission. Also the wireless channel (air) is ideally not transparent and hence will introduce errors in the relayed bitstream. All these will show up as spatio-temporal artifacts in the video that will be rendered to the end user. In order to model what the end user perceives regarding the quality of the rendered videos, we first extract features from channel streams and then develop a model based on machine learning, in order to provide objective scores as well as PDU. However, such system development will first require training data to set the model parameters. Therefore, we developed a simulated video database in which video quality was rated by human observers. In order to train the proposed method for a wide range of situations, video clips with different content, encoding settings and simulation of transmission errors were included in the said database. We also used videos captured from ISDB-T broadcast transmissions to validate and benchmark the proposed model. Hence, the model can be built from simulated data and applied in practice by extracting features from the code stream and obtain predicted MOS (i.e. objective quality score) as well as predicted PDU (i.e. % of dissatisfied users as predicted by the objective model).

We now describe the video database, features employed and the machine learning techniques employed for feature pooling.

A.4.1. Datasets

We used a recently published database with video clips and raw subjective scores of subjective video quality within the context of DTT. The database is extracted from [**66**] and is suitable to train, verify and validate video quality objective models in multimedia broadband and broadcasting operations, under the ISDB-T standard. Specifically under the Brazilian version of the standard, known as ISDB-Tb that uses H.264/AVC for video compression, Advanced Audio Coding

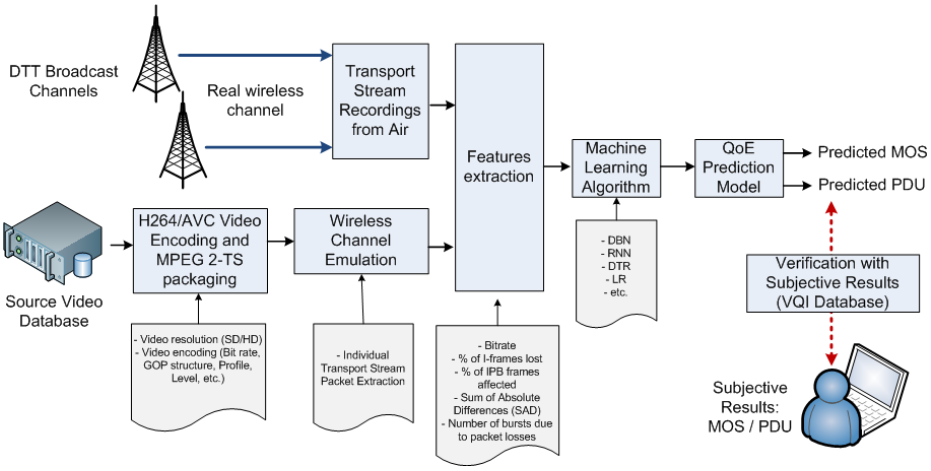


Figure A.1 – An overview of the proposed idea in practical video transmission network.

(AAC) for audio compression and MPEG-2 Transport Stream (TS) for packaging and multiplexing video, audio and data signals in the digital broadcasting system. The subjective tests in this database were conducted following the recommendation ITU-R BT.500-13 [**1**] using the Absolute Category Rating with Hidden Reference (ACR-HR) method. Subjective score collection was automated by employing a software based system [**93**]. The database includes two datasets with video clips that are 9 to 12 seconds in duration.

The first dataset consists of videos distorted by simulation of the video delivery chain. For this dataset, five High Definition (HD, resolution being 1920×1080) source (reference) sequences were used[2], namely "Concert", "Football", "Golf", "Foxbird" and "Voile". Each source video has undergone an encoding process with different encoding settings according to the ISDB-Tb standard using H.264/AVC and MPEG 2-TS for packaging. Then, a process of individual TS packet extraction was performed in order to simulate transmission errors. A total of 20 encoding and packet loss pattern conditions were generated for each source sequence providing $5 \times 20 = 100$ HD distorted video sequences. Since resolution is an important aspect in video quality, the same process was applied to down-sampled source video sequences, thus providing another 100 SD resolution (720×576) distorted sequences. Thus, the first dataset has 200 (100 HD and 100 SD) distorted video sequences. The encoding settings that have been imposed on the videos are: for SD (HD) videos: Profile = Main (High), Level = 3.1 (4.1), GOP length = 33, frame rate = 50fps and bit rate from 0.7 to 4 Mbps (3.5 to 14 Mbps). As for the different packet loss patterns, it was used 0% (no losses), 0.3% of losses with uniform distribution and 0.1% or 10% of packet losses within zero, one, two or three burst errors. For more details on the creation of this dataset, the interested reader can refer to [**66**].

[2]These were taken from http://www.cdvl.org and IRCCyN IVC 1080i Video Quality Database [**165**].

The second dataset, generated for validation purposes, includes only real recorded video clips from air from two different DTT Broadcast Channels. In this dataset, different encoding impairments and real packet losses patterns can be found in both HD and SD resolution (thus, there are 200 sequences, 100 HD and 100 SD). Each of the 200 video versions were evaluated by a human panel consisting of at least 18 viewers (27 for any HD video and 18 for any SD video) in a controlled environment. The MOS scale was used for these evaluations. All results were recorded in the database of [**66**] that is used here as well.

In this appendix, both datasets were used, i.e. a total of 400 video sequences distorted by encoding impairments and transmission errors. Also note that the content types (i.e. source sequences) in both datasets were different.

A.4.2. Feature set

In DTT the video signal is typically coded in H264/AVC or MPEG-2 and packetized in small packets of 188 bytes (TS packets) prior to being modulated and transmitted. In MPEG-2 compression the compressed video frames are grouped into Group of Pictures (GoP). Each GoP usually uses three types of frames, named: I-intra , P-predictive, and B-bidirectional. I frames are encoded with Intra-frame compression techniques while P and B frames use motion estimation and compensation techniques. I frames are used as reference frames for the prediction of P and B frames. The GoP size is given by the number of frames existing between two I frames. In the case of H264/AVC each frame can be split into multiple slices: I, P or B. Both compression techniques can be packaged in Transport Stream (TS) packets. Each TS packet contains 4 bytes of header and 184 of payload. The header contains, among other fields, a 4-bit long Continuity Counter that can be used to count the amount of packet losses in the received bit stream.

Our approach to select the features was based on previous no-reference methods such as the one described in [**92**]. For our method, the selected features are the following:

- **Bit rate:** The obtained video bit rate due to the encoding process (H.264/AVC) and the MPEG-2 TS packaging.
- **Percentage of I-frames lost:** The I-frames carry the most reliable and important information, compared to P and B frames. Also I frames help decode non I frames, therefore their partial or total loss due to transmission errors is a key quality degrading factor.
- **Percentage of I,P,B frames lost:** In addition to the most crucial I frames, we also use this metric to account for P and B frames directly hit by transmission errors (without any further distinctions though).
- **SAD (Sum of Absolute Differences):** The SAD of Residual Blocks is a spatio-temporal metric that for instance addresses the degree of complexity of a sequence of images to be compressed.
- **Number of bursts:** Transmission errors normally affect groups of frames. The amount of bursts was selected in order to quantify the number of sequential frames directly hit by transmission errors in a video transmission (e.g. first a IIBPP frames are directly hit by transmission errors and then a PBPIPIBBB), we employ the number of bursts as a factor for objective quality prediction.

These features are used as input to the ML algorithm, as depicted in Figure A.1. Otherwise put, they constitute the key QoE influence factors that we have identified, which will be used to build the ML-based QoE prediction model. Once a QoE model is built and put into practice, these features will be extracted from data streams and used as input for the QoE prediction. Of course, additional or different features can be used and hence the described method is scalable in terms of feature selection.

A.4.3. Feature pooling

We employed a number of feature pooling methods. These include both linear and non-linear models namely Linear Regression (LR), Decision Tree based Regression (DTR), Artificial Neural Networks (ANNs), and Deep Belief Networks (DBN).

A.4.3.1. *Random Neural Networks (RNN).* The first model under scrutiny is Random Neural Network (RNN), which combines classical ANNs with queuing networks. Similar to ANN, RNN is composed of different layers of interconnected processing elements (i.e. neurons/nodes) that cooperate to resolve a specific problem by instantaneously exchanging signals between each other and from/to the environment. RNN is well adapted for QoS/QoE learning [**154**] since it takes short training time as compared to ANN, is less sensitive to selection of hidden nodes as compared to ANN and can capture QoS/QoE mapping functions in a more robust and accurate way. The success of the use of RNN for learning is suggested in a number of works [**6**, **14**, **45**, **71**, **154**, **184**, **185**].

A.4.3.2. *Deep Belief Networks (DBN).* The second model studied in this appendix is inspired from Deep Learning (DL) [**20**], which makes small steps towards the mimicking of the human brain [**91**]. Technically, DL can be seen as the natural evolution of ANN [**106**]. Besides that, DL methods achieve very good results outperforming state-of-the-art algorithms, including classical ANN models (e.g. Multi Layer Perceptron), in different real-world problems such as multi-class classification [**104**], collaborative filtering [**176**], transfer learning [**28**], people detection [**149**], information retrieval [**70**], activity recognition [**141**] and so on. Hence, our goal was to investigate to what extent DL can be applied to the problem of NR video quality prediction. While some prior work of applying DL for image quality evaluation exists [**73**, **136**, **137**, **196**], a study of its effectiveness for NR video quality estimation especially in a multi-output scenario, as considered in this appendix, has not been reported in literature.

Specifically, in this appendix, we employed Deep Belief Networks (DBN) which are stochastic neural networks with more hidden layers and high generalization capabilities. They are composed by many, much simpler, two-layers stochastic neural networks, namely Restricted Boltzmann Machines (RBMs) [**186**] which are stacked one above the other in a deep architecture as depicted in Figure A.2. More precisely, a DBN consists of an input layer with real values (i.e. $\mathbf{x}$), a number of n hidden binary layers (i.e $\mathbf{h^1}$,...,$\mathbf{h^n}$), and an output layer (i.e. $\mathbf{y}$) with real-values. The neurons from different layers are connected by weights (i.e. $\mathbf{W^1}$, ..., $\mathbf{W^n}$, $\mathbf{W^o}$). Formally, a DBN models the joint distribution between the input layer $\mathbf{x}$

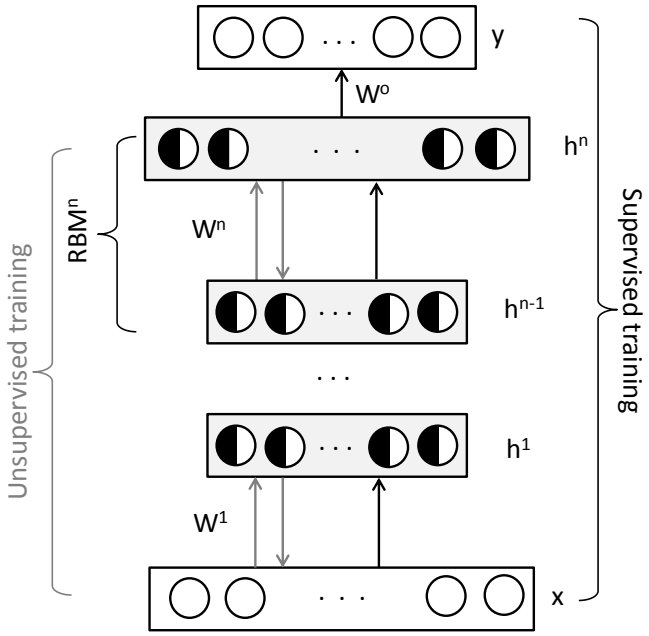


Figure A.2 – General architecture of DBN.

and the n hidden layers, as it is shown next:

$$P(\mathbf{x}, \mathbf{h^1}, ..., \mathbf{h^n}) = \Big(P(\mathbf{x}|\mathbf{h^1}) \prod_{k=1}^{n-2} P(\mathbf{h^k}|\mathbf{h^{k+1}})\Big) P(\mathbf{h^{n-1}}, \mathbf{h^n}) \tag{A.2}$$

, where $P(\mathbf{h^k}|\mathbf{h^{k+1}})$ is a conditional distribution of the input units conditioned on the hidden units of the RBM^{k+1}, $\forall 1 \leq k < n-1$, given by:

$$P(\mathbf{h^k}|\mathbf{h^{k+1}}) = \prod_j P(h_j^k|\mathbf{h^{k+1}}) \tag{A.3}$$

$$P(h_j^k = 1|\mathbf{h^{k+1}}) = \frac{1}{1 + e^{-\sum_l W_{jl}^{k+1} h_l^{k+1}}} \tag{A.4}$$

, and $P(\mathbf{h^{n-1}}, \mathbf{h^n})$ is the joint distribution of the two layers composing RBM^n, computed as:

$$P(\mathbf{h^{n-1}}, \mathbf{h^n}) = \frac{1}{\mathcal{Z}(\mathbf{W^n})} e^{\sum_{j,l} W_{jl}^n h_j^{n-1} h_l^n} \tag{A.5}$$

, with $\mathcal{Z}(\mathbf{W^n})$ being the partition function of RBM^n. For RBM^1, $P(\mathbf{x}|\mathbf{h^1})$ can be computed in a similar manner with $P(\mathbf{h^k}|\mathbf{h^{k+1}})$.

The learning of DBNs parameters (e.g. $\mathbf{W^k}$) is made in two phases, as described in [**82**]. The first one is the *unsupervised training* phase. Herein, the weights $\mathbf{W^1}$, ..., $\mathbf{W^n}$ are considered to be bidirectional and the model is trained in an unsupervised way to learn to reconstruct probabilistically the inputs as well as possible, by using just the input data. As it is shown in Figure A.2, in this phase just the neurons from the input and the hidden layers are involved. After this training phase, the hidden layers may perform automatically features extraction on inputs (i.e. the neurons which compose the hidden layers turn on or off when

some specific values in a subset of the input neurons set occur). The second phase is the *supervised training* and the neurons from all the layers are involved in it. Herein, the model learns to perform classification or regression. More exactly, the previous learned DBN model is transformed in a directed neural network from bottom to top. The weights $\mathbf{W^1}$, ..., $\mathbf{W^n}$ are initialized with the previous learned values, while $\mathbf{W^o}$ are randomly initialized. After that, the DBN model is trained to fit pairs of input and output data points, as best as possible, by using a standard neural network training algorithm, such as back-propagation [**174**]. However, the above represents just a high level description of the DBNs formalism with the scope of providing to the non-specialist reader an intuition about the mechanisms behind DBNs. The overview of the deep learning complete mathematical details do not constitute one of the goals of this appendix and the interested reader is referred to [**20**] for a thorough discussion.

A.5. Experimental results and analysis

This section presents experimental evaluation, and related analysis of the results obtained.

A.5.1. Test method setup

To assess the performance of our proposed method, we have considered two scenarios. First, we performed content-independent within dataset cross validation using the first video dataset (recall there are two datasets used in this study as discussed in the previous section). Since there are 5 different types of content, we performed a 5 fold cross-validation, where each fold represents one video type. In total, we repeated the experiments five times, each time choosing a different video to test the models, and the other four to train them. In the second scenario, we employed cross dataset validation: one dataset was used as training set and the other one as testing set. Hence we ensured that in both scenarios, train and test sets were content independent. In both scenarios, for all the machine learning algorithms analyzed, the inputs consist of features described in Section A.4.2.

A distinct advantage that DBN offers over other competing methods is that they can be effectively initialized with unlabeled data in the unsupervised learning phase, and the second phase involves labeled data. As a result, they would require much less labeled training data to achieve similar or better prediction performance. Clearly, this is desirable in the context of video quality estimation where the availability of labeled data (i.e. subjective video quality ratings) is limited for obvious reasons. Thus, we have used two DBN models which employed less labeled training data (i.e. pairs inputs-outputs) in the *supervised learning* phase, while in the *unsupervised learning* phase they were trained with all the data but without the need of the corresponding label. Besides that, we have analyzed DBN and RNN models with one output (i.e. the model is specialized to predict just MOS or just PDU) or with two outputs (i.e. the model is capable to predict both, MOS and PDU). More specifically, in all sets of experiments performed, we have used the following DBN and RNN models: DBN^1_{100} (it used 100% of the labeled training data and it had 1 output), DBN^2_{100} (it used 100% of the labeled training data and it had 2 outputs), DBN^1_{40} (it used 40% of the labeled training data chosen randomly and it had 1 output), DBN^2_{40} (it used 40% of the labeled training data chosen

randomly and it had 2 outputs), DBN_{10}^{1} (it used 10% of the labeled training data chosen randomly and it had 1 output), DBN_{10}^{2} (it used 10% of the labeled training data chosen randomly and it had 2 outputs), RNN^{1} (it had 1 output), and RNN^{2} (it had 2 outputs).

For the DBN models, we used 3 hidden layers with 10 hidden neurons on each of them. The learning rate (i.e. the factor which applies a greater or lesser portion of the weights adjustments computed in a specific epoch to the older weights computed in the previous epochs) was set to 10^{-3}, momentum (i.e. the factor which allows to the weights adjustments made in a specific epoch to persist for a number of epochs with the final goal to increase the learning speed) to 0.5, the weight decay (i.e. the factor which reduces overfitting to the training data, and shrinks the useless weights) to 0.0002, and the weights were initialized with $\mathcal{N}(0, 0.01)$ (i.e. Gaussian distribution). The number of training epochs in the *unsupervised training* phase was set to 200, while the number of training epochs in the *supervised training* phase using back-propagation was set to 1600. To ensure a smooth training, the data have been normalized to have zero mean and one unit variance as discussed in [**79**]. For the RNN models we used the implementation offered by Changlin Liu and Luca Muscariello[3]. For the LR and DTR implementations we have used the scikit-learn library [**166**].

Besides that, to assess the quality of the PDU predictions using the various machine learning techniques under scrutiny (which are applied directly on the features extracted from the videos), we tried to estimate also the PDU values by using two simpler statistical approaches in which we have exploited the sigmoid-like relation between MOS and PDU. Formally, for each video i from the testing set, we have estimated its PDU value, $\widehat{PDU_i}$, from a Gaussian probability density function, as follows:

$$\widehat{PDU_i} = P(1 \leq x \leq th) = \int_{1}^{th} \frac{1}{\sigma_i \sqrt{2\pi}} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}} \tag{A.6}$$

, where th represents the selected threshold for PDU[4], μ_i represents the MOS for the video i, and σ_i means the standard deviation of all individual subjective scores associated with video i. However, due to the fact that in a real video service it is impossible to obtain μ_i and σ_i in real-time, in our experiments we set μ_i to the MOS value predicted for the video i by the best performer among the machine learning techniques used. At the same time, we have estimated σ_i considering two cases: (1) a fixed value given by the mean value of all standard deviations, computed each of them on the individual subjective scores associated with each video from the training set (method dubbed further FixSig); (2) a variable value given by a Gaussian curve fitted on the MOS values of the videos from the training set and their corresponding standard deviation (method dubbed further FitSig) and the previous discussed μ_i.

The performance was assessed using Pearson (PCC) and Spearman (SRCC) correlation coefficients, and the root mean squared error (RMSE) values. Note that we employed the mentioned performance measures for both MOS and PDU prediction accuracies. To serve as a benchmark, we also computed the results using peak-signal-to-noise (PSNR), which is still a popular FR method. The results

[3]https://code.google.com/p/qoe-rnn/, Accessed on March 7th, 2015.

[4]Please recall that in this appendix th is set to 3.

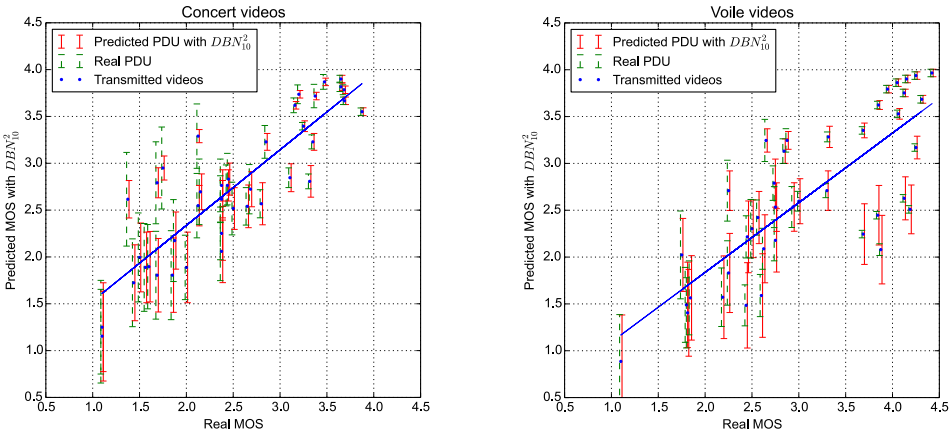


Figure A.3 – Cross-validation results snapshot. The real MOS and PDU values plotted against the predicted MOS and PDU values using DBN_{10}^2 on the best performers (i.e "Concert" videos) and on the worst performers (i.e. "Voile" videos). Each point represents an impaired video.

(correlations, RMSE) for PSNR were computed after the non-linear transformation recommended in [**215**]. The reader will however recall that in the considered application, decoded video data is assumed to be unavailable, and hence objective methods that require pixel data cannot be employed in practice.

A.5.2. Test results

The results for the first scenario i.e. 5-fold cross validation are presented in Table A.3, in which we have reported the RMSE and correlation values for each fold as well as the average over the 5 folds. We can observe that while all the methods achieve statistically similar performances for MOS prediction accuracies, DBNs perform better in predicting PDU. To obtain further insights, we have plotted in Figure A.3 the outcomes of DBN_{10}^2 on two content types namely "Concert" and "Voile". In these plots, the blue dots show the locations of subjective MOS vs the predicted MOS (obviously they will lie on the 45° in case of perfect prediction) while the error bars represent PDU. We have shown the results only for DBN_{10}^2 due to the fact that it is probably the most interesting model because it uses only 10% labeled training data and hence is practically more robust against the amount of labeled training data available. Moreover, recall that DBN_{10}^2 outputs both MOS and PDU simultaneously from single training unlike other models which need to be trained twice on subjective MOS and actual PDU. Hence, it is able to predict both values at the same time. It can be observed in both plots that the blue dots lie close to the main diagonals (which represent the perfect predictions for the MOS values). Moreover, predicted PDU is close to the actual PDU, although the accuracy is less in case of "Voile" sequence at higher subjective MOS.

The results for the second test scenario (cross-dataset validation) are presented in Tables A.4 and A.5. One can again see that DBNs tend to perform better considering both MOS and PDU predictions. Note that PSNR results cannot be computed in case of Table A.4 because the videos were registered from air

Table A.3 – Performance evaluation with 5-fold cross-validation.

Videos	Concert				Foot				Golf				Ntia				Voile				Average			
	MOS		PDU		MOS		PDU		MOS		PDU		MOS		PDU		MOS		PDU		MOS		PDU	
Metrics	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
PSNR	0.30	0.93	n/a	n/a	0.42	0.91	n/a	n/a	0.53	0.87	n/a	n/a	0.47	0.90	n/a	n/a	0.47	0.86	n/a	n/a	0.44±0.08	0.89±0.03	n/a	n/a
LR	0.60	0.76	0.26	0.70	0.60	0.77	0.25	0.71	0.55	0.82	0.22	0.80	0.70	0.79	0.24	0.78	0.70	0.71	0.26	0.71	0.63±0.08	0.77±0.04	0.25±0.01	0.71±0.04
DTR	0.61	0.79	0.34	0.57	0.54	0.81	0.27	0.69	0.77	0.71	0.31	0.67	0.68	0.80	0.25	0.78	0.60	0.79	0.34	0.67	0.64±0.08	0.78±0.03	0.30±0.04	0.67±0.07
RNN^1	0.4	0.85	0.23	0.79	0.52	0.83	0.23	0.76	0.45	0.86	0.20	0.84	0.57	0.88	0.17	0.86	0.59	0.81	0.21	0.80	0.51±0.08	0.85±0.02	0.21±0.03	0.81±0.04
RNN^2	0.54	0.84	0.23	0.78	0.54	0.82	0.23	0.75	0.49	0.87	0.19	0.84	0.61	0.88	0.17	0.86	0.61	0.79	0.21	0.80	0.56±0.06	0.84±0.04	0.21±0.03	0.80±0.05
DBN^1_{100}	0.49	0.83	0.21	0.79	0.52	0.82	0.21	0.78	0.54	0.83	0.21	0.82	0.60	0.84	0.19	0.84	0.66	0.78	0.22	0.79	0.56±0.06	0.82±0.02	0.21±0.01	0.80±0.02
DBN^2_{100}	0.47	0.85	0.20	0.82	0.54	0.82	0.22	0.77	0.50	0.85	0.20	0.83	0.64	0.81	0.21	0.81	0.61	0.80	0.22	0.79	0.55±0.06	0.83±0.02	0.21±0.01	0.80±0.02
DBN^1_{10}	0.44	0.83	0.20	0.81	0.54	0.83	0.22	0.78	0.53	0.83	0.21	0.82	0.58	0.85	0.19	0.84	0.63	0.80	0.22	0.78	0.54±0.06	0.83±0.02	0.21±0.01	0.81±0.02
DBN^2_{10}	0.52	0.82	0.23	0.78	0.52	0.83	0.22	0.78	0.55	0.84	0.21	0.81	0.59	0.84	0.20	0.82	0.57	0.82	0.19	0.81	0.55±0.02	0.83±0.01	0.21±0.01	0.80±0.02
DBN^1_{10}	0.49	0.82	0.23	0.80	0.56	0.83	0.22	0.78	0.59	0.82	0.20	0.83	0.61	0.84	0.22	0.84	0.80	0.79	0.19	0.78	0.61±0.10	0.82±0.02	0.21±0.01	0.81±0.02
DBN^2_{10}	0.46	0.86	0.22	0.82	0.66	0.82	0.24	0.75	0.57	0.81	0.26	0.72	0.68	0.80	0.23	0.80	0.70	0.80	0.26	0.77	0.61±0.08	0.82±0.03	0.24±0.01	0.78±0.03
FixSig	n/a	n/a	0.20	0.83	n/a	n.a	0.28	0.72	n/a	n/a	0.28	0.78	n/a	n/a	0.22	0.84	n/a	n/a	0.39	0.73	n/a	n/a	0.27±0.06	0.78±0.05
FitSig	n/a	n/a	0.19	0.84	n/a	n.a	0.27	0.73	n/a	n/a	0.28	0.77	n/a	n/a	0.22	0.84	n/a	n/a	0.38	0.75	n/a	n/a	0.26±0.06	0.79±0.04

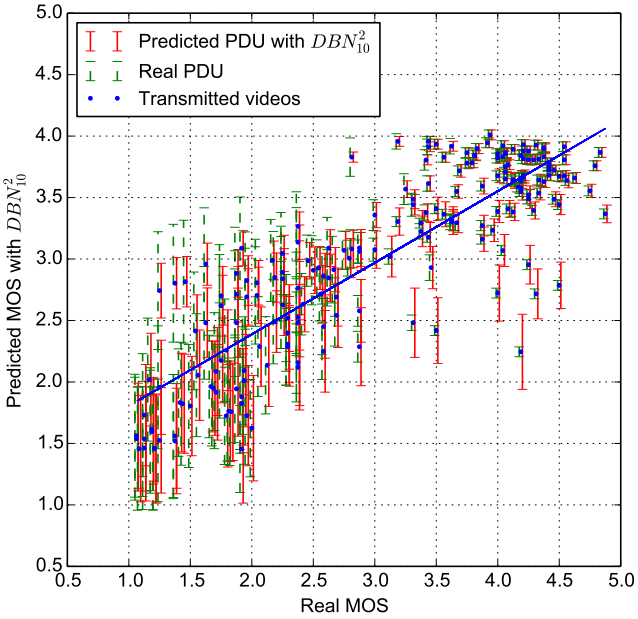


Figure A.4 – Results for the second dataset (note that the system was trained using only the first dataset). The real MOS and PDU values plotted against the predicted MOS and PDU values using DBN_{10}^2. Each point represents an impaired video.

Table A.4 – Cross dataset validation. The system was trained with sequences from the first dataset (200 sequences, 100 HD and 100 SD), and the test set consisted of 200 videos taken from air (the second dataset).

Metrics	MOS			PDU		
	RMSE	PCC	SRCC	RMSE	PCC	SRCC
LR	0.70	0.82	0.81	0.24	0.81	0.82
DTR	0.62	0.83	0.80	0.24	0.80	0.81
RNN^2	0.77	0.84	0.85	0.18	0.90	0.84
DBN_{100}^2	0.58	0.87	0.85	0.20	0.87	0.83
DBN_{40}^2	0.61	0.88	0.83	0.19	0.90	0.84
DBN_{10}^2	0.60	0.86	0.82	0.19	0.88	0.83
FixSig	n/a	n/a	n/a	0.27	0.83	0.81
FitSig	n/a	n/a	n/a	0.28	0.84	0.81

and hence the source (reference) video is unavailable. Hence, these results are relevant for a practical end-to-end video delivery chain where FR methods cannot be employed. Finally, the MOS-PDU plot for the scenario considered in Table A.4 is shown in Figure A.4 (for DBN_{10}^2). This allows the reader to judge the scatter around the diagonal as well as compare the actual and predicted PDU values.

In both test scenarios, we may observe that DBNs perform better for PDU predictions that any other methods in terms of the all evaluation metrics. Besides that, it is interesting to note that even the two simpler statistical methods performs quite well, being able to predict PDUs with a good correlation factors, but having some flaws in the case of the RMSE metric. Moreover, we would like to highlight

Table A.5 – Cross dataset validation. The system was trained with 200 videos taken from air (second dataset), and the test set consisted of 200 sequences (100 HD and 100 SD) from the first dataset.

Metrics	MOS			PDU		
	RMSE	PCC	SRCC	RMSE	PCC	SRCC
LR	0.75	0.75	0.77	0.27	0.72	0.77
DTR	0.77	0.69	0.65	0.31	0.66	0.68
RNN^2	1.25	0.78	0.76	0.24	0.77	0.77
DBN^2_{100}	0.60	0.81	0.81	0.23	0.76	0.80
DBN^2_{40}	0.63	0.79	0.78	0.25	0.74	0.77
DBN^2_{10}	0.65	0.80	0.78	0.24	0.76	0.79
FixSig	n/a	n/a	n/a	0.29	0.62	0.71
FitSig	n/a	n/a	n/a	0.29	0.75	0.77

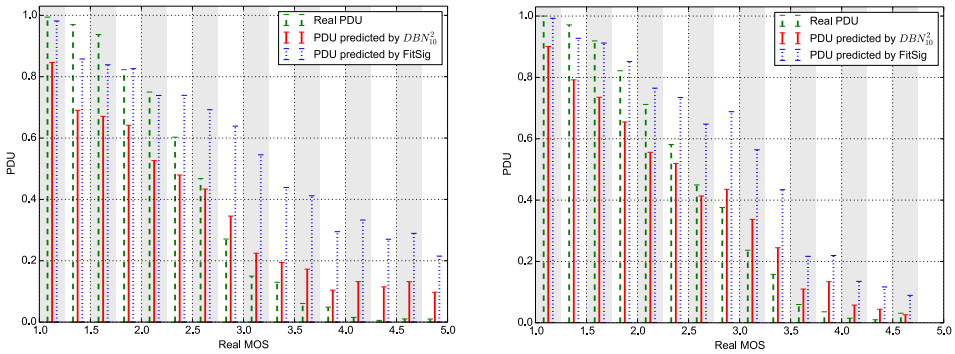


Figure A.5 – Comparison of the real PDU with the predictions made by DBN^2_{10} and FitSig. Each PDU bar represents the mean values of the PDUs situated in the light gray or in the white areas, respectively. In the left plot, the system was trained with the 200 sequences (100 HD and 100 SD) from the first dataset, and the test set consisted of the 200 videos taken from air (second dataset), while in the right plot the training and the testing sets were reversed.

that in our experiments FitSig proven to be more robust than its counterpart FixSig, especially when the subjective studies came from different datasets, due to its better representational power given by a better fitted standard deviation σ_i. For a better insight into the differences between DBNs and the statistical approaches in Figure A.5 we plot the results of DBN^2_{10} and FitSig in the case of the cross dataset validation scenario. Herein, it is interesting to see that at small MOS values FitSig performs better than DBN^2_{10}, while at MOS values usually higher than 2.5, DBNs perform much better. Similarly, we have observed the same behavior also for the other DBN models on one side and FixSig and FitSig on the other side in both test scenarios, the 5-fold cross validation and the cross dataset validation. These, corroborated with the fact that FixSig and FitSig still need an external prediction method to estimate μ_i, make DBNs the most suitable method to predict PDU.

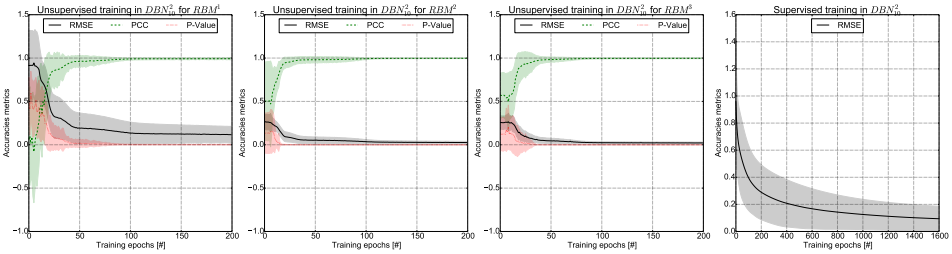


Figure A.6 – The behavior of DBN^2_{10} during the training on the first video dataset. The first three plots depict the *unsupervised training* phase for each RBM belonging to the DBN^2_{10}, while the last one presents the *supervised training* phase in which the DBN^2_{10} was trained using back-propagation. The straight lines represent the mean and the shaded areas reflect the standard deviation computed for all the data points.

Table A.6 – Analytic study of the relations between the DBNs weights in different learning phases. The assessment metrics are computed between the weights of the DBN under scrutiny after the *supervised learning* phase and their corresponding values obtained after *unsupervised learning* phase and before the *supervised* one.

Training Set	Model	$\mathbf{W^1}$			$\mathbf{W^2}$			$\mathbf{W^3}$		
		RMSE	PCC	SRCC	RMSE	PCC	SRCC	RMSE	PCC	SRCC
First Dataset	DBN^2_{100}	0.17	0.98	0.96	0.49	0.93	0.93	0.30	0.97	0.98
	DBN^2_{40}	0.20	0.97	0.95	0.54	0.92	0.91	0.36	0.96	0.97
	DBN^2_{10}	0.22	0.96	0.94	0.56	0.91	0.90	0.38	0.96	0.96
Second Dataset	DBN^2_{100}	0.11	0.99	0.99	0.23	0.99	0.98	0.26	0.98	0.98
	DBN^2_{40}	0.15	0.99	0.98	0.26	0.98	0.98	0.29	0.98	0.97
	DBN^2_{10}	0.14	0.99	0.98	0.26	0.98	0.97	0.27	0.98	0.98

A.5.3. Learning of weights in deep belief networks

To understand better how deep learning works, in Figure A.6, the behavior of DBN^2_{10} during the training on the first video dataset is plotted. It can be observed that in the *unsupervised learning* phase the model learns to reconstruct the inputs well after approximately 50 training epochs, and after roughly 100 training epochs it reconstructs them very precisely, independently of the RBM under scrutiny (RBM^1,RBM^2,RBM^3). More than that, the same plot suggests a clear correlation between the three performance metrics used over the training epochs to assess the learning process, such that when the averaged RMSE and P-value tends to get closer to zero, the averaged PCC value tends to get closer to one, showing overall a perfect correlation between them. Further on, in the *supervised learning* phase, DBN^2_{10} learns with back-propagation to predict the training outputs with a very small error after about 800 training epochs. We would like to highlight, that all the DBN models discussed in this appendix, independently on the scenario, had a similar behavior as the one described previously for DBN^2_{10}.

Furthermore, we have analyzed the most important free parameters (i.e. the weights $\mathbf{W^1}$, $\mathbf{W^2}$, $\mathbf{W^3}$, and $\mathbf{W^o}$) of the DBN models used in this experiment. The relations between these parameters are exemplified visually in Figure A.7, and presented in Table A.6. In both, it can be observed, that practically the weights learned during the *unsupervised training* phase do not change too much after the *supervised training* phase, independently if we study DBN^2_{100}, DBN^2_{40}, or DBN^2_{10}.

This probably explains why in the literature, the latter one is called "fine tuning". At the same time, the fact that the weights of the three fine tuned DBNs end up in a region very close to the one discovered by the initial unsupervised learning procedure reflects also why a DBN which uses just 10% of the labeled data for the back propagation training has a similar performance with one which uses 100% of the labeled data. Besides that, the sparsity patterns of the weights reflect which input neurons contribute more to any hidden neurons. As an example, we can observe that the neuron number 8 from $\mathbf{h^1}$ is affected just by neurons 3 (i.e. % of total frames lost) and 5 (i.e. # of bursts) from $\mathbf{x}$, or in other words the DBNs models automatically find a correlation between % of total frames lost and # of bursts. Similarly, we can deduce that the 10^{th} hidden neuron from $\mathbf{h^1}$ represents a relation between all the 5 input features used. It is worth highlighting, that using similar cascade deductions, one might discover why the neuron number 9 from $\mathbf{h^3}$ has such a strong impact on both neurons (i.e. MOS and PDU) from the output layer $\mathbf{y}$.

A.6. Discussion

One of the main aims of this appendix has been to demonstrate how objective quality prediction can be augmented by considering variability of subjective data. Particularly, we have shown how machine learning can add value to objective video quality estimation by considering a two-output DBN model. Hence, we train the model not only to predict MOS but also to put subjective variability into observation. Consequently, we are able to deepen our understanding of the service in question from two perspectives: overall service quality and the satisfaction of the customer base. Utilizing percentage below threshold instead of standard deviation or other typical mathematical scattering indicators unveils the answer to the question "how many users are not happy with the service" instead of "are users on average happy with the service". These two perspectives have a profound difference when it comes to quality management, as quality does not translate directly into business success: slightly bad quality does not mean slightly decreased market share. In some cases it can be the differentiating factor between success and failure. Meeting the needs of all customers and detecting and dealing with customer dissatisfaction are key components in service quality management, especially when we consider things of high abstraction level such as quality of experience.

During the course of the study we learned that there is a rough sigmoid-like correlation between MOS and uncertainty of MOS for this dataset. This observation cannot be generalized for all datasets and selected features, but it is nonetheless notable that when MOS drops around the selected threshold of satisfaction, the number of dissatisfied users increases the fastest. Different features may pose different kind of relations depending on how opinions of subjects vary due to particular feature. This phenomenon becomes more apparent if participants of subjective assessment are selected from different regions, age groups, cultures and backgrounds. This was noted for example in [**206**] where authors studied website aesthetics and discovered a major difference between Asian and non-Asian users in perception of website visual appeal. We propose that this may also apply to certain quality aspects where some user groups perceive some quality degradation as much worse than other users.

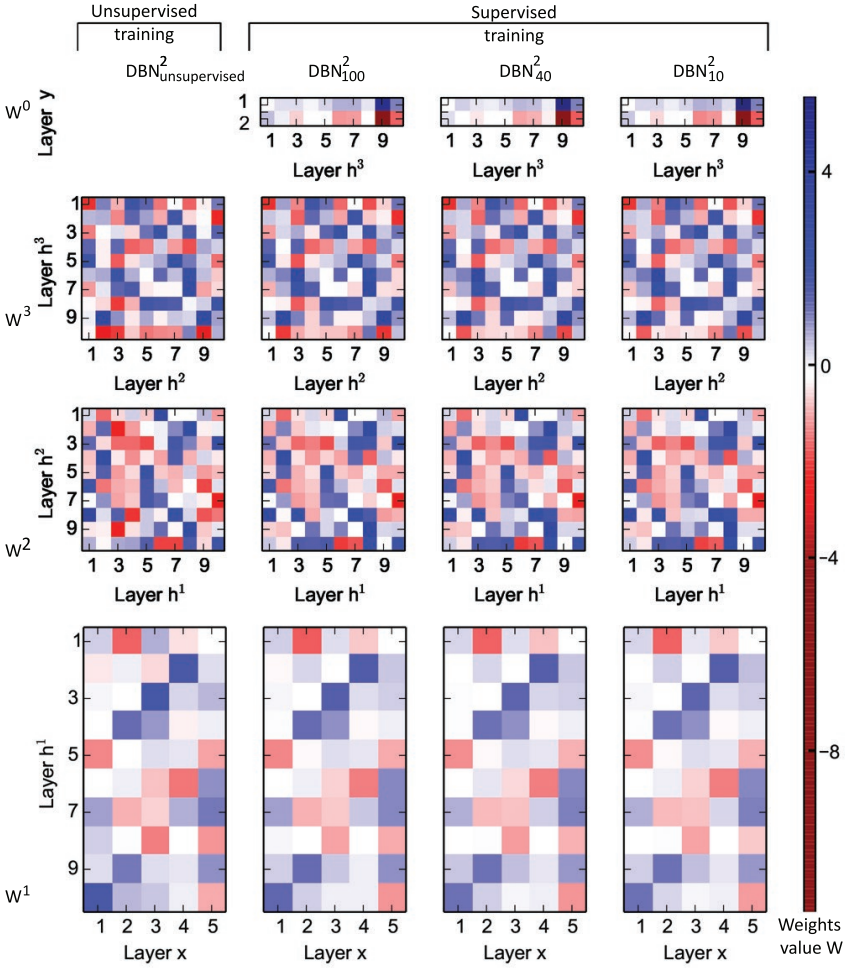


Figure A.7 – The values of the weights in the DBNs models when the training was done on the first data set. The values on the x-axis and y-axis represent the index of neurons from that specific layer. The neurons from the input layer **x** represent the following features: real bit rates (i.e. 1^{st} neuron), % of I-frames lost (i.e. 2^{nd} neuron), % of total frames lost (i.e. 3^{rd} neuron), SAD (i.e. 4^{th} neuron), # of bursts (i.e. 5^{th} neuron). The first column reflects the weights of the $\text{DBN}^2_{unsupervised}$ obtained after the *unsupervised training* phase, while the last three columns represent the weights of the DBNs obtained after the *supervised training* phase. Moreover, on rows, the bottom one represents the DBNs inputs, while the top one represents the DBNs outputs. The dark red in the heat maps represents weights values closer to -11, while the white depicts weights values around 0, and the dark blue shows weights values towards 6. It is interesting to see that after the training process many of the weights become very close to 0.

User dissatisfaction information can be utilized in many ways in practice. Traditional management mechanisms such as traffic shaping, admission control or handovers can be further enhanced to also include a "risk threshold" for user dissatisfaction in addition to MOS threshold. For instance, let us assume a QoE managed service where a provider is able to automatically monitor the service per-user level. The provider uses a machine-learning model which outputs two values, objective MOS and probability that the user is not satisfied. The management mechanism can step in to improve the user experience if either the estimated MOS drops below a certain threshold, or if the estimated dissatisfaction level rises above a certain value (for example, MOS is required to remain above 3 and risk that the user opinion is below 3 must be less than 5%).

But what may be even more useful for the service provider is the overall MOS and dissatisfaction percentage throughout the service. This also helps providers to reflect how the service is doing competition-wise and if they can expect user churn in the near future. Holistic, real-time monitoring may also help to indicate serious faults and problems either with the service or the transfer network and help to act accordingly. Operators can therefore react to user dissatisfaction before customers either terminate their service subscription or burden customer service.

A.7. Concluding thoughts

While the problem of objective video quality assessment has received considerable research attention, most existing works tend to focus only on averaged ratings. As a result, valuable information generated as a result of inter-observer differences (i.e. subjective variability) is simply lost in objective quality prediction. This appendix attempted to introduce and analyze one such instance of how the scattering of subjective opinions can be exploited for business-oriented video broadcasting applications. This was accomplished by first analyzing and formulating interpretable measure of user dissatisfaction which may not always be reflected in averaged scores. To put the idea into practice, we then explored the deep learning framework and jointly modeled the averaged scores and user dissatisfaction levels so that the predicted objective video quality score is supplemented by user satisfaction information. At the same time we showed that by using deep belief networks the amount of subjective studies required to learn to make accurate predictions, which outperform clearly the other machine learning models considered for comparison in this appendix (i.e. linear regression, regression trees, and random neural networks), may be reduced up to 90%. This will be useful in a typical video broadcasting system where customer (user) churn needs to be continuously monitored. We also demonstrated a practical implementation of our ideas in the context of video transmission. We designed the system so that video quality and user dissatisfaction can be predicted from data bit stream without the need of the fully decoded signal. This greatly facilitates real time video quality monitoring since objective quality can be predicted from the code stream.

APPENDIX B

Algorithms

This appendix lists the main algorithms proposed in this thesis.

```
% Initialization;
set graph G = (V, E);
set Φ_0^n for each node n ∈ V, number of thieves per node;
set T = log^2|V|; % the number of epochs to stop the algorithm;
% Run the game;
for each epoch e = 1 : T do
    Set Ψ_e^l = 0, for each link l;
    for each thief a do
        if a in 'empty' state then
            set n to the actual position of a;
            a moves to next node m, given by the probability p_a^nm;
            if m ∈ ϒ_a then
                Eliminate cycle from ϒ_a;
            else
                Add m to the end of ϒ_a;
            end
            if m has vdiamonds then
                a set state to 'loaded';
                Φ_e^m = Φ_e^m − 1;
            end
        end
        if a in 'loaded' state then
            set n to the actual position of a; % the last node from ϒ_a;
            a moves to last but one node m from ϒ_a;
            a removes node n from ϒ_a;
            set Ψ_e^l = Ψ_e^l + 1 where l is the link between n and m;
            if m is the home node of a then
                Φ_e^m = Φ_e^m + 1;
                a set state to 'empty';
            end
        end
    end
end
% Computes centralities;
Computes the centrality of each node n, Φ̄_T^n = (1/T) Σ_{e=0}^{T} Φ_e^n ;
Computes the centrality of each link l, Ψ̄_T^l = (1/T) Σ_{e=0}^{T} Ψ_e^l ;
```

Algorithm B.1: Game of Thieves (GOT) algorithm. For more details please see Chapter 2.

%% Initialization of the various parameters
Set n_h, n_v, n_{Gs}, n_{CD}, n_E, n_B, $n_{\hat{B}}$, α, ρ, ξ
Initialize RBM parameters Θ_0(i.e., $\mathbf{W}_0$, $\mathbf{a}_0$, $\mathbf{b}_0$)$\sim \mathcal{N}(0, \sigma)$
Set $\Delta\Theta_0^{n_E} = 0$
Set $t = 1$, $\mathbf{B}_t = \varnothing$
%% A continuous loop to handle sequential incoming data
while *system is running* **do**
    Observe a new data point $\mathbf{d}$
    Add $\mathbf{d}$ to $\mathbf{B}_t$
    if $\mathbf{B}_t$ *contains* n_B *observed data points* **then**
        Set $\hat{\mathbf{B}}_t = \varnothing$
        %% Generate new data points with the RBM
        if $t > 1$ **then**
            for $i = 1 : n_{\hat{B}}$ **do**
                %% Run Gibbs sampling
                Initialize $\mathbf{h} \sim \mathcal{U}(0, 1)$
                for $k = 1 : n_{Gs}$ **do**
                    Infer $P(\mathbf{v} = 1|\mathbf{h}, \Theta_{t-1})$
                    Infer $P(\mathbf{h} = 1|\mathbf{v}, \Theta_{t-1})$
                end
                Add $\mathbf{v}$ to $\hat{\mathbf{B}}_t$
            end
        end
        %% Update parameters
        Set $\Theta_t^0 = \Theta_{t-1}$ and $\Delta\Theta_t^0 = \Delta\Theta_{t-1}^{n_E}$
        for $e(epoch) = 1 : n_E$ **do**
            %% Create a training batch from $\mathbf{B}_t$ *and* $\hat{\mathbf{B}}_t$
            Set $\mathbf{V} = \mathbf{B}_t \cup \hat{\mathbf{B}}_t$
            Infer $P(\mathbf{H} = 1|\mathbf{V}, \Theta_t^{e-1})$
            %% Collect positive statistics Ψ^+
            Compute Ψ^+ from $\mathbf{V}$ and $\mathbf{H}$
            for $k = 1 : n_{n_{CD}}$ **do**
                Infer $P(\mathbf{V} = 1|\mathbf{H}, \Theta_t^{e-1})$
                Infer $P(\mathbf{H} = 1|\mathbf{V}, \Theta_t^{e-1})$
            end
            %% Collect negative statistics Ψ^-
            Compute Ψ^- from $\mathbf{V}$ and $\mathbf{H}$
            %% Perform parameters update
            $\Delta\Theta_t^e = \rho\Delta\Theta_t^{e-1} + \alpha[(\Psi^+ - \Psi^-)/(n_B + n_{\hat{B}}) - \xi\boldsymbol{\Theta}_t^{e-1}]$
            $\Theta_t^e = \Theta_t^{e-1} + \Delta\Theta_t^e$
        end
        Set $\Theta_t = \Theta_t^{n_E}$
        %% Clean the memory
        Delete $\hat{\mathbf{B}}_t$, $\mathbf{B}_t$ from memory
        %% Advance to the next time step
        Set $t = t + 1$, $\mathbf{B}_t = \varnothing$
    end
end

Algorithm B.2: Online Contrastive Divergence with Generative Replay. Note that OCD_{GR} only stores the last variant of Θ_t^e and $\Delta\Theta_t^e$ in memory. Still, we notate them as being indexed by t for a better illustration of the time and training epochs dimensions. For more details please see Chapter 3.

```
%% define $n_v$, $n_h$ (number of visible and hidden neurons, respectively);
%% assumptions: $n_v > 4$ and $n_h > 4$ (we consider non trivial cases);
%% define $\sigma^{neigh}$, $\phi$ (parameters to control the nodes local connectivity);
%% initialization;
set $L^{th} = log(n_v + n_h)$;%% set a threshold for the small-world topology;
%% topology generation;
repeat
    generate randomly $S^{PL}$, a power law degree sequence of size $n_v + n_h$ with the minimum degree of 4;
    sort $S^{PL}$ in descending order;
    set $S^v$=[] and $S^h$=[]; %% sequences to store the degree of the visible and hidden nodes,respectively;
    $i$=1;
    while $i \leq 2 \times min(n_v, n_h)$ do
        $S^v$.append($S^{PL}[i]$);
        $S^h$.append($S^{PL}[i+1]$);
        i=i+2;
    end
    if ($n_v > n_h$) then
        $S^v$.append($S^{PL}[2 \times n_h : end]$);
    end
    else
        $S^h$.append($S^{PL}[2 \times n_v : end]$);
    end
    if $sum(S^v) < sum(S^h)$ then
        add $sum(S^h) - sum(S^v)$ degrees equally distributed among the visible nodes;
    end
    else
        add $sum(S^v) - sum(S^h)$ degrees equally distributed among the hidden nodes;
    end
    $G$ =createBipartiteGraphUsingHavelHakimiProcedure($S^v$, $S^h$) [76];
    for $o$=1:$\phi$ do
        for $i$=1:$n_v$ do
            while a finite number of trials do
                $j = \lceil \mathcal{N}((i \times n_h)/n_v, \sigma^{neigh}) \rceil$;%% sampled from a Gaussian distribution;
                if $0 \leq j \leq n_h$ then
                    addEdge $(i, j)$ to $G$;
                    break;
                end
            end
        end
        for $j$=1:$n_h$ do
            while a finite number of trials do
                $i = \lceil \mathcal{N}((j \times n_v)/n_h, \sigma^{neigh}) \rceil$;%% sampled from a Gaussian distribution;
                if $0 \leq i \leq n_v$ then
                    addEdge $(i, j)$ to $G$;
                    break;
                end
            end
        end
    end
    $L$ =computeAverageShorthestPath($G$);
until $L \leq L^{th}$;
%% fit the topology to the data;
re-arrange the visible nodes in $G$ s.t. the ones with higher degree correspond to data features with higher std. dev.;
%% topology utilization;
use the visible nodes from G as the visible layer in XBM (or GXBM);
use the hidden nodes from G as the hidden layer in XBM (or GXBM);
use the edges from G as the weights in XBM (or GXBM);
```

Algorithm B.3: Pseudo-code of the algorithm used to generate the topology of the XBM and GXBM models. For more details please see Chapter 4.

```
%Initialization;
initialize ANN model;
set ϵ and ζ;
for each bipartite fully-connected (FC) layer of the ANN do
    replace FC with a Sparse Connected (SC) layer having a Erdős-Rényi
    topology given by ϵ and Eq.5.1;
end
initialize training algorithm parameters;
%Training;
for each training epoch e do
    perform standard training procedure;
    perform weights update;
    for each bipartite SC layer of the ANN do
        remove a fraction ζ of the smallest positive weights;
        remove a fraction ζ of the highest negative weights;
        if e is not the last training epoch then
            add randomly new weights (connections) in the same amount as the
            ones removed previously;
        end
    end
end
```

Algorithm B.4: Sparse Evoluationary Training (SET) pseudocode. For more details please see Chapter 5.

Abbreviations

Abbreviation	Description
ABAC	Attribute-Based Access Control
AI	Artificial Intelligence
AIS	Annealed Importance Sampling
ANN	Artificial Neural Network
AUC	Area under the curve
BC	Betweenness Centrality
BGP	Border Gateway Protocol
Ca-RBM	Cardinality Restricted Boltzmann Machine
CC	Connected Components
CBP	Circular Back Propagation
CD	Contrastive Divergence
CNN	Convolutional Neural Network
CFBC	Current Flow Betweenness Centrality
DBN	Deep Belief Network
DCT	Discrete Cosine Transformation
DFFW-CRBM	Disjunctive Factored Four Way Conditional Restricted Boltzmann Machine
DL	Deep Learning
DMOS	Degradation Mean Opinion Scores
DRL	Deep Reinforcement Learning
DTR	Decision Tree based Regression
DTT	Digital Terrestrial Television
ER	Experience Replay
FFW-CRBM	Factored Four Way Conditional Restricted Boltzmann Machine
FixProb	Fixed Probability
FR	Full-Reference
FTrRBM	Factored Transfer Restricted Boltzmann Machine
GC	Giant Component
GoP	Group of Pictures
GOT	Game of Thieves
GRBM	Gaussian Restricted Boltzmann Machine
GXBM	Gaussian compleX Boltzmann Machine
GR	Generative Replay
GPU	Graphics processing unit
HD	High Definition

IOT	Internet of Things
ISP	Internet Service Provider
LR	Linear Regression
MCMC	Monte-Carlo Markov Chain
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multi Layer Perceptron
MOS	Mean Opinion Score
MPEG	Motion Picture Expert Group
NR	No-Reference
NRP	Node Removal Procedure
NS	Network Science
OCD_{GR}	Online Contrastive Divergence with Generative Replay
PCC	Pearson Correlation Coefficient
PDU	Percentage of Dissatisfied Users
PSNR	Peak Signal to Noise Ratio
QoE	Quality of Experience
RBM	Restricted Boltzmann Machine
RBM_{OCD}	RBM trained with OCD_{GR}
RBM_{ER-ML}	RBM trained using Experience Replay with a Memory Limit
RBM_{ER-IM}	RBM trained using Experience Replay with Infinite Memory
RBMSim	Restricted Boltzmann Machine Similarity Measure
RGB	Red Green Blue
RL	Reinforcement Learning
RMSE	Root-mean-square error
RNN	Random Neural Network
RR	Reduced-Reference
SAD	Sum of Absolute Differences
SC	Sparse Connected layer
SD	Standard Definition
SET	Sparse Evolutionary Training
SET-RBM	Restricted Boltzmann Machine with Sparse Evolutionary Training
SET-MLP	Multi Layer Perceptron with Sparse Evolutionary Training
SGD	Stochastic Gradient Descent
SOC	Second Order Centrality
SOS	Standard deviation of Opinion Scores
SMcCD	Sequential Markov chain Contrastive Divergence
SRCC	Spearman Correlation Coefficient
SVR	Support Vector Regression
TL	Transfer Learning
TrPrTr	Train Prune Train
TS	Transport Stream
WSN	Wireless Sensor Networks
XBM	compleX Boltzmann Machine

List of publications

The work described in Chapters 2-5 represents a selection of all the work carried out during the course of this PhD study, focusing on fundamental research; while a prominent application is described in Appendix A. The complete list of publications resulted from this PhD research is listed below.

Journal publications

(1) D.C. Mocanu, H. Bou Ammar, L. Puig, E. Eaton, A. Liotta: *Estimating 3D Trajectories from 2D Projections via Disjunctive Factored Four-Way Conditional Restricted Boltzmann Machines*, Pattern Recognition, Elsevier, 2017.

(2) M. Torres Vega, D.C. Mocanu, J. Famaey, S. Stavrou, A. Liotta: *Deep Learning for Quality Assessment in Live Video Streaming*, IEEE Signal Processing Letters, IEEE, 2017.

(3) M. Torres Vega, D.C. Mocanu, A. Liotta: *Unsupervised Deep Learning for Real-Time Assessment of Video Streaming Services*, Multimedia Tools and Applications, Springer, 2017.

(4) M. Torres Vega, D.C. Mocanu, S. Stavrou, A. Liotta: *Predictive No-Reference Assessment of Video Quality*, Signal Processing: Image Communication, Elsevier, 2017.

(5) D.C. Mocanu, E. Mocanu, P.H. Nguyen, M. Gibescu, A. Liotta: *A topological insight into restricted Boltzmann machines*, Machine Learning, ECML-PKDD 2016 special issue, Springer, 2016.

(6) M. Torres Vega, V. Sguazzo, D.C. Mocanu, A. Liotta: *An experimental survey of no-reference video quality assessment methods*, International Journal of Pervasive Computing and Communications, Emerald (**highly commended paper award**), 2016.

(7) D.C. Mocanu, H. Bou Ammar, D. Lowet, K. Driessens, A. Liotta, G. Weiss, K. Tuyls: *Factored Four Way Conditional Restricted Boltzmann Machines for Activity Recognition*, Pattern Recognition Letters, Elsevier, 2015.

(8) D.C. Mocanu, J. Pokhrel, J. Pablo Garella, J. Seppänen, E. Liotou, M. Narwaria: *No-reference Video Quality Measurement: The Added Value of Machine Learning*, Journal of Electronic Imaging, SPIE, 2015.

Journal publications (under review / to be submitted)

(9) D.C. Mocanu, G. Exarchakos, A. Liotta: *Decentralized Dynamic Understanding of Hidden Relations in Complex Networks*, Nature Communications, Springer-Nature (under review), 2017.

(10) D.C. Mocanu, E. Mocanu, P. Stone, P.H. Nguyen, M. Gibescu, A. Liotta: *Evolutionary Training of Sparse Artificial Neural Networks: A Network Science Perspective*, Science, AAAS (under review), 2017.

(11) E. Mocanu, D.C. Mocanu, P.H. Nguyen, A. Liotta, M. Webber, M. Gibescu, J.G. Slootweg: *On-line Building Energy Optimization using Deep Reinforcement Learning*, IEEE Transactions on Smart Grid, IEEE (under review), 2017.

(12) D.C. Mocanu, M. Torres Vega, E. Eaton, P. Stone, A. Liotta: *Online contrastive divergence with generative replay: Experience replay without storing data*, To be submitted for possible publication in Neural Networks, Elsevier; Preprint: CoRR, 2016.

Peer-reviewed conference publications

(13) D.C. Mocanu, E. Mocanu, P.H. Nguyen, M. Gibescu, A. Liotta: *Big IoT data mining for real-time energy disaggregation in buildings (extended abstract)*, Proceedings of the 26th Machine Learning Conference of the Benelux (BeNeLearn), Eindhoven, Netherlands, 2017.

(14) M. Torres Vega, C. Perra, D.C. Mocanu, A. Liotta: *Effect of Lossy Networks on Stereoscopic 3D-Video Streams*, Proc. of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Cagliari, Italy, 2017.

(15) D.C. Mocanu: *On the synergy of network science and artificial intelligence*, Proc. of International Joint Conference on Artificial Intelligence (IJCAI), New York, USA (**travel award**), 2016.

(16) D.C. Mocanu, E. Mocanu, P.H. Nguyen, M. Gibescu, A. Liotta: *Big IoT data mining for real-time energy disaggregation in buildings*, Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary (**student travel grant**), 2016.

(17) D.C. Mocanu, E. Mocanu, P.H. Nguyen, M. Gibescu, A. Liotta: *A topological insight into restricted Boltzmann machines (extended abstract)*, Proc. of Benelux Conference on Artificial Intelligence (BNAIC), Amsterdam, Netherlands, 2016.

(18) M. Torres Vega, D.C. Mocanu, A. Liotta:*A Regression Method for real-time video quality evaluation*, Proc. of International Conference on Advances in Mobile Computing & Multimedia (MoMM), Singapore, 2016.

(19) K. Borchert, M. Hirth, T. Zinner, D.C. Mocanu: *Correlating QoE and Technical Parameters of an SAP System in an Enterprise Environments*, Proc. of 28th International Teletraffic Congress (ITC 28), Würzburg, Germany, 2016.

(20) M. Chincoli, A.A. Syed, D.C. Mocanu, A. Liotta:*Predictive Power Control in Wireless Sensor Networks*, Proc. of IEEE International Conference on Internet-of-Things Design and Implementation (IOTDI), Berlin, Germany, 2016.

(21) M. Torres Vega, V. Sguazzo, D.C. Mocanu, A. Liotta: *Accuracy of No-Reference Quality Metrics in Network-impaired Video Streams*, Proc. of International Conference on Advances in Mobile Computing & Multimedia (MoMM), Brussels, Belgium (**best paper award**), 2015.

(22) D.C. Mocanu, H. Bou Ammar, D. Lowet, K. Driessens, A. Liotta, G. Weiss, K. Tuyls: *Factored Four Way Conditional Restricted Boltzmann*

Machines for Activity Recognition (Type B-paper), Proc. of Benelux Conference on Artificial Intelligence (BNAIC), Hasselt, Belgium, 2015.

(23) E. Mocanu, D.C. Mocanu, H. Bou Ammar. Z. Zivcovik, A. Liotta, E. Smirnov: *Inexpensive user tracking using Boltzmann Machines (Type B-paper)*, Proc. of Benelux Conference on Artificial Intelligence (BNAIC), Hasselt, Belgium, 2015.

(24) D.C. Mocanu, H. Bou Ammar, G. Exarchakos, A. Liotta: *Reduced Reference Image Quality Assessment via Boltzmann Machines*, Proc. of IEEE/IFIP International Symposium on Integrated Network and Service Management (IM), Ottawa, Canada, 2015.

(25) M. Torres Vega, D.C. Mocanu, R. Barressi, G. Fortino, A. Liotta: *An Adaptive Streaming Application for Wireless Android-Based Devices*, Proc. of IEEE/IFIP International Symposium on Integrated Network and Service Management (IM), Ottawa, Canada, 2015.

(26) D.C. Mocanu, F. Turkmen, A. Liotta: *Towards ABAC Policy Mining from Logs with Deep Learning*, Proc. of 18th International Multiconference Information Society (IS), Ljubljana, Slovenia, 2015.

(27) M. Torres Vega, E. Giordano, D.C. Mocanu, D. Tjondronegoro, A. Liotta: *Cognitive No-Reference Video Quality Assessment for Mobile Streaming Services*, Proc. of International Conference on Quality of Multimedia Experience (QoMEX), Costa Navarino, Greece, 2015.

(28) D.C. Mocanu , M. Torres Vega, A. Liotta:*Redundancy reduction in wireless sensors via centrality metrics*, Proc. of IEEE International Conference on Data Mining Workshops (ICDMW), Atlantic City, USA, 2015.

(29) D.C. Mocanu , A. Liotta, A. Ricci, M. Torres Vega, G. Exarchakos: *When does lower bitrate give higher quality in modern video services?*, Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS), Krakow, Poland, 2014.

(30) D.C. Mocanu , G. Exarchakos, A. Liotta: *Deep Learning for objective QoE assessment of 3D images*, Proc. of IEEE International Conference on Image Processing (ICIP), Paris, France, 2014.

(31) H. Bou Ammar, E. Eaton, M. Taylor, D.C. Mocanu, K. Driessens, G. Weiss, K. Tuyls: *An Automated Measure of MDP Similarity for Transfer in Reinforcement Learning*, Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI) Workshops, Qubec, Canada, 2014.

(32) E. Mocanu, D.C. Mocanu, H. Bou Ammar, Z. Zivcovik, A. Liotta, E. Smirnov: *Inexpensive user tracking using Boltzmann Machines*, Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, USA, 2014.

(33) D.C. Mocanu, G. Exarchakos, A. Liotta: *Node Centrality Awareness via Swarming Effects*, Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, USA, 2014.

(34) M. Torres Vega, S. Zuo, D.C. Mocanu, E. Tangdiongga, A.M.J. Koonen, A. Liotta: *Network Performance Assessment with Quality of Experience Benchmarks*, Proc. of International Conference on Network and Service Management (CNSM), Rio de Janeiro, Brazil, 2014.

(35) D.C. Mocanu, G. Santandrea, W. Cerroni, F. Callegati, A. Liotta: *End-to-End Performance Evaluation in High-Speed Wireless Networks*, Proc.

of International Conference on Network and Service Management (CNSM), Rio de Janeiro, Brazil, 2014.

(36) H. Bou Ammar, D.C. Mocanu, M. Taylor, K. Driessens, G. Weiss, K. Tuyls: *Automatically Mapped Transfer Between Reinforcement Learning Tasks via Three-Way Restricted Boltzmann Machines (Type B paper)*, Proc. of Benelux Conference on Artificial Intelligence (BNAIC), Delft, Netherlands, 2013.

(37) A. Liotta, D.C. Mocanu, V. Menkovski, L. Cagnetta, G. Exarchakos: *Instantaneous Video Quality Assessment for lightweight devices*, Proc. of International Conference on Advances in Mobile Computing & Multimedia (MoMM), Vienna, Austria, 2013.

(38) R. Kotian, G. Exarchakos, D.C. Mocanu, A. Liotta: *Predicting Battery Depletion of Neighboring Wireless Sensor Node*, Proc. of International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Vietri sul Mare, Italy, 2013.

(39) H. Bou Ammar, D.C. Mocanu, M. Taylor, K. Driessens, G. Weiss, K. Tuyls: *Automatically Mapped Transfer Between Reinforcement Learning Tasks via Three-Way Restricted Boltzmann Machines*, Proc. of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD), Prague, Czech republic, 2013.

Other conferences

(40) D.C. Mocanu, G. Exarchakos, A. Liotta: *The double link between network science and artificial intelligence. A key to scalable learning solutions?*, European Data Forum (EDF), Eindhoven, Netherlands, 2016.

(41) D.C. Mocanu, G. Exarchakos, A. Liotta: *Learning networks for complex interconnections*, IEEE Student Branch Eindhoven, Netherlands, 2016.

(42) D.C. Mocanu, H. Bou Ammar, D. Lowet, K. Driessens, A. Liotta, G. Weiss, K. Tuyls: *Factored four-way conditional restricted Boltzmann machines (FFW-CRBMs) for activity recognition*, Dutch Foundation for Neural Networks (SNN), Symposium on Intelligent Machines, Nijmigen, Netherlands, 2015.

Curriculum vitae

Decebal Constantin Mocanu was born on May 20^{th}, 1980, in Mizil, Romania. During high school at the National College Mihai Viteazul, Ploiesti, where he graduated in 1998 with a baccalaureate diploma, he qualified several times to the Romanian national Olympiad in Informatics, winning the 3^{rd} prize in 1997. Further on, he continued studying Informatics at the University Politehnica of Bucharest, gaining the Licensed Engineer degree in Computer Science in 2010. In parallel with his engineering studies, between 2001 and 2010, Decebal started MDC Artdesign SRL (a software house specialized in web development), worked as a computer laboratory assistant at the University Nicolae Titulescu, and was a software engineer at Namedrive LLC.

In January 2011, Decebal moved from Romania to Netherlands, where he joined the master program in Artificial Intelligence at Maastricht University. During the whole period of his master studies, Decebal worked as a part time software developer at We Focus BV in Maastricht. In the last year of his master studies, he also worked as an intern at Philips Research in Eindhoven, where he prepared his internship and master thesis projects. He obtained the MSc degree in May 2013, being rewarded with the 1^{st} prize "Master AI Thesis Award" from Maastricht University, for his thesis titled "Activity recognition using four way conditional restricted Boltzmann machine and 3D sensors".

In September 2013, Decebal started his PhD research in Network Science and Artificial Intelligence at Eindhoven University of Technology, the Netherlands. While enrolled in his doctoral studies, he performed three research visits. In 2014, he worked for three months as a visiting scholar at University of Pennsylvania, USA; in 2015, he made a one-week research visit at Julius Maximilian University of Würzburg, Germany; and, in 2016, he was for three months a visiting researcher at University of Texas at Austin, USA.

During his PhD, Decebal has co-authored over 40 peer-reviewed journals and conference papers. He is an active member of the scientific community. He was a program committee member at nine workshops and conferences, and he normally serves as a reviewer for a number of conferences and journals including Machine Learning, IEEE Transactions of Cybernetics, IEEE Transactions of Multimedia, Pattern Recognition Letters, Information Fusion, Journal of Electronic Imaging, Journal of Medical Imaging, NIPS. He gave a few invited talks at Eindhoven University of Technology, the Netherlands; Julius Maximilian University of Würzburg, Germany; University of Texas at Austin, USA; and Nokia Bell Labs. He received several grants, scholarships, and awards during his PhD, including a best paper award at MoMM 2015, and a travel award at IJCAI 2016 from the Artificial Intelligence journal.