Sparsity, redundancy and robustness in artificial neural networks for learning and memory

Philippe Tigréat

▶ To cite this version:

Philippe Tigréat. Sparsity, redundancy and robustness in artificial neural networks for learning and memory. Neural and Evolutionary Computing [cs.NE]. Ecole nationale supérieure Mines-Télécom Atlantique, 2017. English. NNT : 2017IMTA0046 . tel-01812053

HAL Id: tel-01812053

https://theses.hal.science/tel-01812053

Submitted on 11 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UNIVERSITE BRETAGNE LOIRE

THÈSE / IMT Atlantique

sous le sceau de l'Université Bretagne Loire

pour obtenir le grade de

DOCTEUR D'IMT Atlantique

Spécialité : Signal, Image, Vision

École Doctorale Mathématiques et STIC

Présentée par

Philippe Tigréat

Préparée dans le département Electronique

Laboratoire Labsticc

Sparsity, Redundancy and Robustness in Artificial Neural Networks for Learning and Memory

Thèse soutenue le 16 octobre 2017
devant le jury composé de :

Catherine Douillard
Professeur, IMT Atlantique / présidente

Lionel Fillatre
Professeur, Université de Nice Sophia Antipolis / rapporteur

Hubert Cardot
Professeur, Ecole Polytechnique de l'Université de Tours / rapporteur

Martial Mermillod
Professeur, Université Pierre Mendès France / examinateur

Vincent Gripon
Chargé de recherche, IMT Atlantique/ examinateur

Claude Berrou
Professeur émérite, IMT Atlantique / directeur de thèse

Contents

Résumé 6

Abstract 12

Acknowledgements 17

1 Introduction **19**

2 Sparse Neural Associative Memories **23**

2.1 Introduction 24

2.2 Related work 25

2.2.1 Hopfield Networks 25

2.2.2 Willshaw Networks 26

2.2.3 Clustered Cliques Networks 26

2.3 Willshaw Networks and biological considerations 27

2.4 Preventing connections between neighbor neurons 31

2.5 Efficient use of the memory resource 35

2.6 Pattern completion 42

3 Robustness of Deep Neural Networks to Erasures in a Reading Task **54**

3.1 Introduction 54

3.1.1 Visual recognition of partially displayed words 55

3.1.2 Pre-existing models of reading 56

3.1.3 Deep Convolutional Neural Networks . . . 57
3.1.4 Previous study of the representation geometry and biological plausibility of CNNs . . . 59
3.2 Methods . . . 62
3.2.1 Data-set pre-processing . . . 62
3.2.2 The convolutional architecture employed . . . 62
3.2.3 Training procedure . . . 65
3.2.4 Testings to study the effect of the different letter feature types on recognition . . . 67
3.2.5 Study of the learned feature detectors and associated activation patterns 68
3.3 Results . . . 71
3.3.1 Display rate effects . . . 71
3.3.2 Relationship with lexical factors . . . 72
3.3.3 Effects of feature types . . . 73
3.3.4 Observations on the geometry of representations and the information carried by their activation patterns . . . 74
3.4 Discussion . . . 78
3.4.1 Reproducibility of lexical effects relative to word frequency . . . 78
3.4.2 Role of lexical factors . . . 78
3.4.3 Comparison with other models . . . 79
3.4.4 Major differences with human recognition behavior . . . 79
3.5 Conclusions . . . 81

4 Assembly Output Codes for Learning Neural Networks 83
4.1 Introduction . . . 83
4.2 Coding theory . . . 85
4.3 Methodology . . . 87

4.3.1 MNIST 87

4.3.2 SVHN 88

4.3.3 Assembly codes 88

4.3.4 Neural network settings 90

4.4 Results 91

4.4.1 Experimenting with random codes 91

4.4.2 MNIST - Shallow network 93

4.4.3 MNIST - Deep network 94

4.4.4 SVHN - Deep Network 94

4.5 Problems with a large number of classes 96

4.5.1 Memory constraints 96

4.5.2 Error-correcting decoding 97

4.5.3 Experiments 99

4.5.4 Results 101

5 Combination of Unsupervised Learning and Associative Memory 103

5.1 Introduction 104

5.2 Gradient descent learning 104

5.3 Sparse Clustered Associative Memories 106

5.3.1 CCNs 106

5.3.2 Sparse messages 106

5.3.3 Specific retrieval algorithms for sparse messages 108

5.3.4 Accelerations using GPUs 109

5.3.5 GWsTA 110

5.3.6 GLsKO 110

5.3.7 Experiments 114

5.4 Combination of learning and memory 118

6 Conclusion and Openings 124

6.1 Conclusion 124

6.2 Openings 127

6.2.1 Semi-supervised learning 127

6.2.2 Decision under uncertainty 129

Résumé

La problématique générale de la recherche en Intelligence Artificielle (IA) est de parvenir à reproduire les fonctions cognitives du cerveau humain au moyen des ordinateurs modernes. L'objectif à long terme est de répliquer l'ensemble des capacités cognitives humaines les plus sophistiquées, du raisonnement de haut niveau au maniement expert d'outils et d'appareils [1], en passant par la créativité intellectuelle et artistique. Les résultats obtenus ces dernières années [2–6] semblent annoncer une révolution technologique qui pourrait changer profondément la société et instituer de nouveaux paradigmes dans de nombreux domaines tels que l'économie, l'enseignement, la médecine ou la défense.

Certains chercheurs et entrepreneurs envisagent la possibilité qu'une intelligence artificielle de niveau humain puisse advenir autour des années 2030, tels Ray Kurzweil et Yann LeCun [7, 8]. Une proportion notable de chercheurs du domaine considèrerait comme probable l'avènement d'une telle intelligence au cours du XXIe siècle [9]. La seule certitude sur ce point est que les conséquences sur la vie humaine et le développement technologique seraient difficiles voire impossibles à anticiper.

La recherche en intelligence artificielle a démarré concrètement avec les travaux d'Alan Turing et de John Von Neumann, ainsi qu'avec la célèbre conférence de Dartmouth en 1956. Dés 1957, un modèle phare de réseau de neurones est inventé par Frank Rosenblatt, le perceptron. Les années suivantes voient le développement de nombreux algorithmes, dont certains visant à reproduire la mémoire associative, tels que le modèle de Willshaw (1969) [10]. Le temps passant, les résultats sont globalement décevants par rapport aux attentes initiales, et les financements se raréfient. Plusieurs décennies plus tard, le progrès du matériel informatique et notamment des processeurs graphiques (GPUs) permet d'appliquer des algorithmes déjà anciens sur des bases de données de dimension importante. Dans les années 2000, des modèles de réseaux de neurones tels que le perceptron, l'auto-encodeur, les réseaux de neurones convolutifs (CNN) reviennent au premier plan. Ils permettent alors d'obtenir

des gains en performances marquants sur des jeux de données tels que MNIST et surtout ImageNet [2, 11].

Mémoire et Apprentissage

Nous focalisons notre intérêt sur deux composantes fondamentales du comportement cognitif du cerveau, l'apprentissage et la mémoire. Les algorithmes de mémoires associatives [12–14] offrent la possibilité de stocker des éléments d'information et de les récupérer à partir d'une partie de leur contenu. En cela ils reproduisent le comportement de la mémoire cérébrale, et notamment de la mémoire à long-terme en réponse à l'évocation partielle d'un élément de connaissance ou d'un souvenir.

Les mémoires associatives à cliques en particulier permettent de stocker une quantité relativement grande d'information dans un espace de mémoire donné [10]. Chaque élément de mémoire étant enregistré par l'interconnexion complète des neurones qui le forment, le motif graphique créé est caractérisé par une surabondance de connexions. Cette redondance permet de reconstituer un message y compris lorsqu'une proportion notable de ses connexions est effacée.

Ces mémoires utilisent des connexions à poids binaires, contrairement à la majorité des modèles de réseaux de neurones artificiels. Cette caractéristique contribue à leur robustesse et s'accorde avec le fonctionnement de la mémoire à long-terme. Les éléments stockés dans celle-ci sont très probablement supportés par un principe matériel stable et peu sujet aux fluctuations rapides et permanentes de la règle de Hebb [15]. Ce principe repose donc plus vraisemblablement sur l'existence ou l'absence de connexions que sur des valeurs de poids continues et évolutives.

Plusieurs auteurs ont avancé l'idée que le codage de l'information dans le cerveau devait être analogique dans certaines régions et numérique dans d'autres, notamment dans les aires associées à la mémoire à long terme [16, 17]. Mochizuki et Shinomoto ont proposé une démonstration du caractère tantôt analogique ou numérique du codage employé dans différentes aires cérébrales [18]. Pour obtenir ce résultat, ces auteurs cherchent à inférer la fréquence de décharge dans un train d'impulsions par une estimation bayésienne empirique d'une part, et par l'apprentissage d'un modèle de Markov à états cachés d'autre part. La comparaison de la vraisemblance de ces estimateurs permet de discriminer entre des trains d'impulsions de fréquence continue résultant d'un codage analogique, et d'autres à fréquences discrètes produites par un codage numérique. Cette méthode semble mettre en évidence que l'information traitée dans le cortex visuel primaire et l'aire temporale médiale est analogique, tandis que les signaux produits par le corps géniculé latéral du thalamus sont numériques.

Des arguments théoriques ont également été avancés pour justifier l'intérêt d'un codage hybride, tantôt analogique ou numérique. Sarpeshkar affirme ainsi que si l'idée d'un fonctionnement en grande partie analogique est très largement acceptée, l'accumulation de bruit dans certaines régions tend à rendre un tel fonctionnement inefficace [16]. L'utilisation d'un codage numérique dans le thalamus serait donc cohérent avec son rôle particulier d'intégrateur et de relais de signaux entre plusieurs réseaux corticaux [19, 20].

Une autre conséquence probable du haut niveau de bruit dans certaines régions cérébrales est le codage distribué de l'information. Selon les rapports de signal à bruit, Sarpeshkar montre qu'un compromis peut être trouvé en terme de degré de distribution de l'information, sachant qu'un codage distribué réduit le coût énergétique de calcul global mais augmente les coûts de communications synaptiques [16].

Enfin, avec une fréquence de décharge ne dépassant pas 250 Hz, et des impulsions synaptiques voyageant à une vitesse proche de 20 mètres par seconde, le matériel neuronal est très lent en comparaison avec nos ordinateurs. Le cerveau effectue malgré cela des tâches complexes très rapidement grâce à des algorithmes parallèles très optimisés [21].

Pour passer d'une perception analogique du monde extérieur à une représentation parcimonieuse et plus compacte, nous nous sommes tournés vers l'apprentissage profond. Nous avons considéré que ces réseaux de neurones constituent une bonne option pour reproduire le rôle des canaux sensoriels, et qu'ils peuvent ainsi être interfacés avec les algorithmes de mémoires associatives pour reproduire la combinaison de ces deux fonctions cérébrales.

Ces réseaux permettent d'opérer une compression intelligente reposant sur des caractéristiques de bases apprises dans les données. Cela est bien plus plausible biologiquement que d'utiliser des filtres fixes pré-dessinés, comme l'ont montré plusieurs études sur le cortex visuel. Dans certaines expériences célèbres, des chats ont été exposés lors de leurs premières semaines uniquement à des orientations verticales, ou horizontales selon le groupe. Une fois libérés dans un environnement normal, ces chats se montraient aveugles aux orientations auxquelles ils n'avaient pas été exposés auparavant. Cela démontre que si le système visuel est forcément partiellement pré-cablé, ses filtres ne se mettent en place correctement que par une période d'apprentissage nécessaire [22, 23].

Plan

Dans le chapître 2, nous présentons un modèle de mémoire associative neurale inspiré des réseaux de Willshaw, dans lequel une contrainte de connectivité est introduite entre neurones voisins pour reproduire le phénomène biologique d'inhibition latérale observé dans le cerveau. Cette contrainte apporte une amélioration de la performance de récupération des messages et un stockage plus efficace de l'information.

Dans le chapître 3, afin d'étudier la plausibilité biologique des réseaux profonds convolutifs, une architecture convolutive a été appliquée sur une tâche de lecture de mots partiellement affichés suivant des conditions similaires à une précédente étude de psychologie impliquant 23 sujets humains. Cette expérimentation a mis en évidence les similarités de comportement du réseau convolutif avec les sujets humains concernant le taux d'affichage et la fréquence lexicale des mots, ainsi que la réponse préférentielle à certaines caractéristiques visuelles des lettres.

Dans le chapître 4, une nouvelle méthode de représentation des catégories dans les réseaux profonds est proposée. En associant aux différentes catégories des assemblées de neurones recouvrantes plutôt qu'un neurone unique selon la manière habituelle, la performance du système est favorisée et il peut être interfacé avec d'autres algorithmes, tels que les mémoires associatives neurales ou encore un code correcteur d'erreur. De plus, pour les problèmes à grand nombre de classes, cette méthode d'encodage des classes permet de réduire significativement les dimensions d'un réseau.

Dans le chapître 5, une méthode d'interfaçage des réseaux de neurones profonds non supervisés avec les mémoires associatives neurales est présentée. Les réseaux profonds étant exécutés sur GPU, une implémentation parallèle du modèle de mémoire associative est requise pour une exécution fluide du système combiné. Des méthodes d'implémentation sur GPU de deux algorithmes de récupération de messages sont proposées.

Abstract

The general problematic of research in Artificial Intelligence (AI) is to manage to reproduce the cognitive functions of the human brain by means of modern computers. The long term objective is to replicate all the most sophisticated human cognitive abilities, from high-level reasoning to expert handling of tools and devices [1], through to intellectual and artistic creativity. The results obtained these last few years [2–6] seem to announce a technological revolution that could profoundly change society and impose new paradigms in numerous domains such as economy, education, medicine or defense.

Some researchers and entrepreneurs consider the possibility that a human-level artificial intelligence may happen around the years 2030s, such as Ray Kurzweil and Yann Lecun [7,8]. A notable proportion of researchers of the domain would consider likely the advent of such an intelligence during the 21st century [9]. The only certainty on this matter is that the consequences on human life and technological development would be difficult if not impossible to anticipate.

Research in artificial intelligence started concretely with the works of Alan Turing and John Von Neumann, as well as the notorious Dartmouth conference in 1956. As soon as 1957, a seminal neural network model is invented by Frank Rosenblatt, the perceptron. The following years see the development of numerous algorithms, some of which aiming to reproduce associative memory, such as the Willshaw model (1969) [10]. Over time, the results are disappointing overall with regards to initial expectations, and research credits become increasingly scarce. Several decades later, the progress of computer hardware and notably graphical processors (GPUs) allows the application of already ancient algorithms on high-dimensional data-sets. In the years 2000s, neural network models such as the perceptron, the auto-encoder, and convolutional neural networks (CNN) come to the fore. They then allow to obtain notable performance improvements on data-sets such as MNIST and ImageNet [2, 11].

Memory and Learning

We focus our interest on two fundamental components of the cognitive behavior of the brain, learning and memory. Algorithms of associative memory [12–14] offer the possibility to store information elements and to retrieve them using a sub-part of their content. In this, they reproduce the behavior of cerebral memory, and notably of long-term memory in response to the partial evocation of a knowledge element or a memory.

Clique-based associative memories in particular allow to store a relatively large quantity of information in a given memory space [10]. Each memory element being stored by the full inter-connection of neurons composing it, the graphical pattern created is characterized by an abundance of connections. This redundancy allows to reassemble a message, including when a notable proportion of its connections is erased.

These memories use binary-weighted connections, to the contrary of the majority of artificial neural network models. This feature contributes to their robustness and is in accordance with the behavior of long-term memory. The elements stored in this memory are very likely to be supported by a material principle that is stable and not very susceptible to the rapid and permanent fluctuations of Hebbian learning [15]. This principle is thus more likely to rely on the existence or absence of connections rather than on continuous and evolving weights values.

Several authors have put forward the idea that information coding in the brain should be analog in certain areas and digital in others, notably in areas associated with long-term memory [16, 17]. Mochizuki and Shinomoto have proposed a demonstration of the analog or digital nature of the coding used in different brain areas [18]. To get to this result, these authors attempt to infer the spiking rate in a spike train by an empirical Bayes method on the one hand, and by learning a hidden Markov model on the other hand. The comparison of the likelihood of these estimators allows to discriminate between spike trains with continuous frequency resulting from an analog coding scheme, and others with discrete frequency produced by a digital encoding. This method seemingly brings to light that the information treated in the primary visual cortex and the medial temporal area is analog, whereas the signals produced by the lateral geniculate nucleus in the thalamus are digital.

Theoretical arguments have also been made to justify the interest of hybrid coding, partly analog and digital. Sarpeshkar thus states that if the idea of a largely analog functioning is widely accepted, the accumulation of noise in certain areas tends to render such an operation inefficient [16]. The use of a digital encoding in the thalamus would thus be consistent with its particular role of signal integrator and hub between several cortical networks [19, 20].

Another likely consequence of the high level of noise in certain brain areas is the distributed encoding of information. Depending on the signal to noise ratio, Sarpeshkar shows that a trade-off can be found in terms of the degree of distribution of information, knowing that a distributed encoding lowers the global energetic cost of computations but raises the synaptic communications costs [16].

Lastly, with a spiking rate below 250 Hz, and synaptic spikes traveling at a speed close to 20 meters per second, the neuronal material is very slow in comparison with our computers. The brain however executes complex tasks very quickly thanks to highly optimized parallel algorithms [21].

To transition from an analog perception of the outside world to a sparse and more compact representation, we turned ourselves towards deep learning. We considered that these neural networks constitute a good option to reproduce the role of sensory channels, and that they can thus be interfaced with associative memory algorithms in order to reproduce the combination of these two cerebral functions.

These networks allow to operate a smart compression relying on basic features learned in the data. This is much more biologically plausible than to use fixed hand-engineered filters, as have demonstrated several studies of the visual cortex. In certain famous experiments, cats were exposed solely to vertical orientations, or horizontal depending on the group, during their first weeks. Once freed into a normal environment, these cats were blind to the orientations they had not been formerly exposed to. This demonstrates that if the visual system is necessarily partly pre-wired, its filters can only properly establish themselves through a necessary learning period [22, 23].

Plan

In Chapter 2, we present a neural associative memory model inspired by Willshaw networks, in which a connectivity constraint is introduced between neighbouring neurons in order to reproduce the biological phenomenon of lateral inhibition observed in the brain. This constraint brings an improvement of performance in message retrieval and a more efficient storage of information.

In Chapter 3, in order to study the biological plausibility of deep convolutional networks, a convolutional architecture was applied on a task of reading partially displayed words under similar conditions as those in a former psychology study involving 23 human subjects. This experiment has put in evidence the similarities in the behavior of the convolutional network with the human subjects regarding the display rate and the lexical frequency of words, as well as the preferential response to certain visual features in letters.

In Chapter 4, a new method for representing categories in deep networks is presented. By associating different categories with overlapping neuron assemblies rather than a single neuron in the usual manner, the performance of the system is enforced and it can be interfaced with other algorithms, such as neural associative memories or an error-correcting code. Moreover, for problems with a large number of classes, this method for encoding classes allows to reduce significantly the dimensions of a network.

In Chapter 5, a method for interfacing deep unsupervised networks with neural associative memories is introduced. Deep networks being executed on GPU, a parallel implementation of the associative memory model is required for a fluid execution of the combined system. GPU implementation methods for two messages retrieval algorithms are presented.

Acknowledgements

To begin with, I thank my Ph.D. advisor Claude Berrou for the favorable material conditions he allowed me to work in, and for creating a dynamic and intellectually stimulating environment with the NEUCOD team. Claude has demonstrated enduring trust in my work, and I learned by his side the necessity to always believe in one's ideas and to keep promoting them.

Vincent Gripon brought me his support and ideas that led to publications.

Pierre-Henri Horrein provided practical advice that saved me a lot of time, and also helped me significantly with a conference paper.

I thank the members of the jury, particularly reviewers Lionel Fillatre and Hubert Cardot.

Michel Jézéquel was there to listen to me and advise me when I needed it.

Catherine Blondé brought her help in dealing with many administrative obligations.

I want to thank Jean Chaoui, who managed me optimally during my final year internship in Télécom Bretagne. Working with Jean was easy, motivating and it taught me a lot. I also thank Chafiaa Hamitouche who supervised this internship and who helped me again down the road.

I thank Olivier Dufor, for his friendly support during these years and for his help with a chapter of this work, and his wife Deok-Hee for associating me with her research on reading.

Alexis Lechat, student at Télécom Bretagne and an intern in the electronics department for two semesters, did excellent work and helped me tremendously with a chapter of this thesis.

Both Xiaoran Jiang and Carlos Lassance were instrumental in this work. We used to work together daily on Deep Learning ideas, in 2015 for Xiaoran and in 2016 for Carlos.

I thank Grégoire Mercier for his advice on methods of decision under uncertainty.

The friendly support I received from Max Sobroza was much appreciated, especially in

2016 when the pressure was not lacking.

Stephen Larroque also brought me useful support, along with some rich discussions where I could appreciate his vast scientific knowledge.

I thank Ehsan Sedgh Gooya for the friendly chats we had along the years spent working on this subject. I also thank Bartosz Boguslawsky and Behrooz Aliabadi, and all the members of the NEUCOD environment.

My thanks obviously go to some dear friends who supported me over the years, among which Amael, Deepak, Dalel, Gwendolyn, Zhuldyz, and Pedro.

Finally I thank my whole family and especially my father whom I talked to regularly, and who advised me on how to handle multiple situations during this very special work period.

Chapter 1

Introduction

Animal intelligence has emerged on our planet through an evolutionary process spanning hundreds of millions of years. Notable milestones along the way were the first reflex arcs (580 millions of years ago), the first central nervous system (550MY ago), the reptilian brain managing breathing, heartbeat and survival instinct (265MY ago), the limbic system dealing with primary emotions (225MY ago), and finally the neocortex (80MY ago). The latter allowed mammals to elaborate sophisticated tools and strategies, and to progressively develop complex languages between 100.000 and 50.000 years ago.

From the early stages of development of Computer Science, the idea of recreating human-like thinking into machines has occupied the minds of researchers and engineers. Even though the idea of endowing artificial beings with reason is ancient, the modern quest for Artificial Intelligence really began with the works of Alan Turing and John Von Neumann, and the Dartmouth Conference in 1956. The initial enthusiasm of notorious attendees like John McCarty, Marvin Minsky and Claude Shannon has since been dampened by two periods of research setbacks and subsequent scarce fundings, called "AI winters". Nonetheless, the idea that most human cognitive abilities may one day be reproduced *in silico* has nothing but gained terrain.

One obvious reason for this is the propagation of Artificial Narrow Intelligences, which are software programs specialized for one task, and capable of performing it better than humans. Behind famous examples like Deep Blue, Watson [4], AlphaGo [5], and self-driving cars [1], systems from this category are commonplace, from the supercomputers used for weather forecasting, to the planning and analysis systems used in the military to optimize logistics, through to air traffic management software. Even low-scale mathematic softwares allow anyone to solve complex equations with many variables in the blink of an eye, illustrating Moravec's paradox that high-level reasoning can be way easier to recreate artificially than low-level sensory-motor functions [24]. Although initially surprising, this finding can in part be explained by the fact that sensory-motor abilities only seem trivial to us because they have been forged in our body structure through hundreds of millions of years of evolution. Their execution is thus largely unconscious and automatic. High-level concepts have only kept our minds busy for a few thousands of years, and therefore appear more difficult to us not because they are intrinsically hard, but because they are relatively new.

High-level thinking also simply involves less computations than perception and motor skills. Recently, researchers have suggested that neuronal communications may be digital in some parts of the brain and analog in others [18]. One could assume that mental information may be supported by digital processes. Long-term memory, in particular, has to be very robust to physico-chemical perturbations, and therefore likely relies on encoding schemes using discrete values and important amounts of redundancy [17].

Sensory systems, though, use more energy and comprise complex hard-wired chains of treatment. The foremost example of this is the visual system and its numerous areas. Our senses can perceive tiny variations in the environment and can thus be likened to analog systems. They treat complex, high-dimensional input flows and extract sparse and low-dimensional information out of it through a complex arrangement of filters.

The compression, sparsification and memorization performed by the nervous system evoke a Shannon-Weaver model. The two major steps in the function of such a model are an initial removal of redundancy present in the input, followed by the addition of useful redundancy to ensure robustness to alterations.

Deep Learning is a family of algorithms that has generated lots of interest in recent years. These methods rely on hierarchies of neurons arranged in layers, where inter-layer connection weights get optimized through a gradient descent procedure with the goal of mapping input vectors to output objectives as precisely as possible. This principle is not new, and it is mainly the progress of the hardware that has allowed to use it on large scale datasets, and to eventually establish new state-of-the-art performances on a vast array of problems.

Neural Associative Memories are models of human memory that can complete patterns when provided some of their subparts. NAMs typically involve formal neurons with binary activity. The most efficient models employ sparse and distributed codes, as they use assemblies of neurons sampled in a pool to represent messages. The large number of combinations brings an important diversity of possible messages. More than this, synaptic connections are drawn between all elements of these assemblies to form cliques, which are strongly robust to partial erasure because of their graphic redundancy.

We propose that a combination of Deep Learning and clique-based associative memories can be a plausible way to reproduce the brain functions of feature extraction from sensory inputs and storage in memory. Inspired by the Shannon-Weaver model, this idea has been a guideline through this work.

In Chapter II, we present a model of clique-based associative memory that takes into account the phenomenon of lateral inhibition observed in the sensory areas of the brain.

Chapter III investigates the behavior of a deep convolutional network in a word reading task, in comparison with human subjects tested on the same data. This comparison of human and artificial readers attempts to study the plausibility of the deep convolutional network model on a functional level.

In Chapter IV, a new method of category representation in supervised learning neural networks is presented, based on assemblies of neurons. This approach modifies the learning process and brings noticeable improvements in performance. It comes with practical advantages for problems with a large number of categories, in terms of memory requirements and computational complexity. It also brings the possibility to interface the network with a clique-based associative memory, or with an error-correcting decoder.

In Chapter V, we present a way to combine an unsupervised learning neural network with a clique-based associative memory. The existing libraries for programming deep neural networks leverage the massive parallelism of graphics hardware. For this reason, a method for running the associative memory on GPU as well is presented, that makes its execution time comparable to that of the learning network.

Chapter 2

Sparse Neural Associative Memories

Willshaw networks are a type of associative memories with a storing mechanism characterized by a strong redundancy. Namely, all the subparts of a message get connected to one another. We introduce an additional specificity, by imposing the constraint of a minimal space separating every two elements of a message. This approach results from biological observations, knowing that in some brain regions, a neuron receiving a stronger stimulation can inhibit its neighbors within a given radius. Theoretical arguments are derived to quantify the benefits of this method in terms of memory usage as well as pattern completion ability. We experiment with different values of the inhibition radius introduced, and we study its impact on the error rate in the retrieval of stored messages. We show that this added constraint can result in significatively better performance of the Willshaw network, either when reducing its set of connections, or when extending its set of neurons while maintaining the memory resource.

2.1 Introduction

Associative memories are a type of computer memories that are part of the broader category of content-addressable memories. Where addressable memories associate an address with a piece of data, associative memories have the characteristic of associating patterns to one another. Among this group, we distinguish between hetero-associative memories, and auto-associative memories. An hetero-associative memory will associate together patterns in pairs. For instance, if the pattern p_1 was associated with pattern p_2, the request p_1 will bring the response p_2. Auto-associative memories follow a different principle, as they will associate a pattern with itself. The main use case of these memories is pattern completion, where a request made of a subpart of a stored message will get as response the completed pattern. Associative memories can be found in several types of real-world applications, such as database engines [25], network routers [26], data compression devices [27], and computer vision systems [28, 29].

Today, it is widely accepted that the working principle of the brain can often be likened to the operation of a set of associative memory. The focus is put here on the phenomenon observed in biological neural networks, called lateral inhibition [30]. It can also be referred to as surround suppression [31]. This translates in the inhibition exerted by some neurons on their close neighbors when these have an activity inferior to their own. Starting from the Willshaw model [10], we propose a neural associative memory that is improved in terms of plausibility, by the introduction of local inhibition that results in the prohibition of short-range connections. We show that this modification brings a performance improvement in the retrieval of stored messages. This also brings insight regarding the good performances of clustered associative memories [14].

Section 2.2 introduces three associative memory models with relevant relationship to this work. Section 2.3 gives a formal presentation of Willshaw networks, including the usual message retrieval algorithm, and biological considerations motivating the modifications we introduce. Section 2.4 details modifications in our implementation as compared to the classic Willshaw model, including the constraint applied on the space between connected neurons. Section 2.5 provides theoretical arguments showing the better usage of memory brought by this constraint. Section 4.4 presents the results we obtain in pattern completion, and gives some theoretical explanations.

2.2 Related work

2.2.1 Hopfield Networks

The prominent model for associative memories was introduced by John Hopfield [12, 32]. Hopfield networks are made of a set of N neurons that are fully interconnected. The training of these networks, given n binary vectors x^μ of length N, consists in modifying the weight matrix W according to the formula:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{n} x_i^\mu x_j^\mu, \tag{2.1}$$

where element w_{ij} at the crossing between line i and column j of W is the real-valued connection weight from neuron i to neuron j.

As connections are reciprocal and not oriented, we have:

$$w_{ij} = w_{ji} \quad \forall i, j \in [\![1, N]\!] \tag{2.2}$$

for any indices i and j in the list of neurons, which makes W symmetrical.

The binary values considered for the stored messages are usually -1 and 1, but can be adapted to work with other binary alphabets. The Hopfield model has a limited efficiency, on which asymptotic limits have been detailed in [33]. The limits of the model can be explained by the facts that each entry of the matrix is modified at every time step of the storing procedure, and that the changes are made in both directions and can, therefore, cancel each other out. This overfitted characteristics of associative memories is very different from that observed in learning applications. Indeed, an overfitted learning system recognizes only the training samples and fails at generalizing to novel inputs. To the contrary, an overfitted storing system recognizes everything and does not discriminate anymore between stored and nonstored data.

2.2.2 Willshaw Networks

Willshaw networks are another model of associative memories in which information is carried by the existence or absence of connections [10,34]. Its material is made of a set of N neurons and N^2 potential connections between them. A message is then a fixed size subset of the N neurons, and can be represented by a sparse vector of length N with ones at these neurons' positions and zeros everywhere else. The connection weights are binary, and the active units in a message get fully interconnected as soon as it is memorized, thus forming a clique. Figure 2.1 gives an example of such a network. The performances of Willshaw networks are way superior to those of Hopfield memories, given that stored messages are sparse (i.e., they contain a small proportion of nonzero elements). Further theoretical and numerical comparison between Hopfield and Willshaw networks can be found in [35–38].

2.2.3 Clustered Cliques Networks

Recently, a novel type of associative memories was proposed by Gripon et al., called Gripon-Berrou Neural Networks (GBNNs) or clustered cliques networks (CCNs) [14, 39]. These associative memories make use of powerful yet simple error correcting codes. These networks consider input messages to be nonbinary, and more precisely to be words in a finite alphabet of size l. This specific structure allows the separation of nodes into different clusters, each being constituted of the same number l of nodes. Connections between nodes inside a given cluster are forbidden, only the connections between nodes in two different clusters are allowed. There again, this model brings a significantly improved performance as compared to the former state-of-the-art of associative memories, namely Willshaw networks [40–42]. For instance, it can be found experimentally that with 2,048 nodes and 10,000 stored messages of order 4 and 2-erasures queries, a Willshaw network will have an error rate close to 80%, while a clique-based neural network will only make 20% of wrong retrievals.

In Hopfield networks, the number of messages one can store and retrieve successfully is linearly proportional to the number of nodes [36]. In clique-based neural networks however, storage capacity grows quadratically as a function of the number of units.

One of the objectives of the present work is to explain the performance improvement brought by the separation of the network into clusters. Therefore, we study a network that can be considered as an intermediate between the Willshaw and Gripon-Berrou models. More precisely, our proposed model adds a locally exclusive rule for nodes to be active in the network.

2.3 Willshaw Networks and biological considerations

Willshaw networks are models of associative memories constituted of a given number of neurons. A stored message, or memory, is a combination of nodes taken in this set. The storage of this information element corresponds to the creation of connections with unitary weights between every two neurons in this message. The graphical pattern thus formed is termed "clique". The storing process of n binary vectors x^μ of length N, is equivalent to the modification of elements of the network's connection matrix W, according to the formula:

$$w_{ij} = \max_{\mu} x_i^\mu x_j^\mu \tag{2.3}$$

Note that here, the max operator is performed coefficient-wise. Equivalently, the connection weight between nodes i and j is equal to 1 if, and only if, those two nodes are both part of one of the n stored messages.

The network's density d is defined as the expected ratio of the number of ones in the

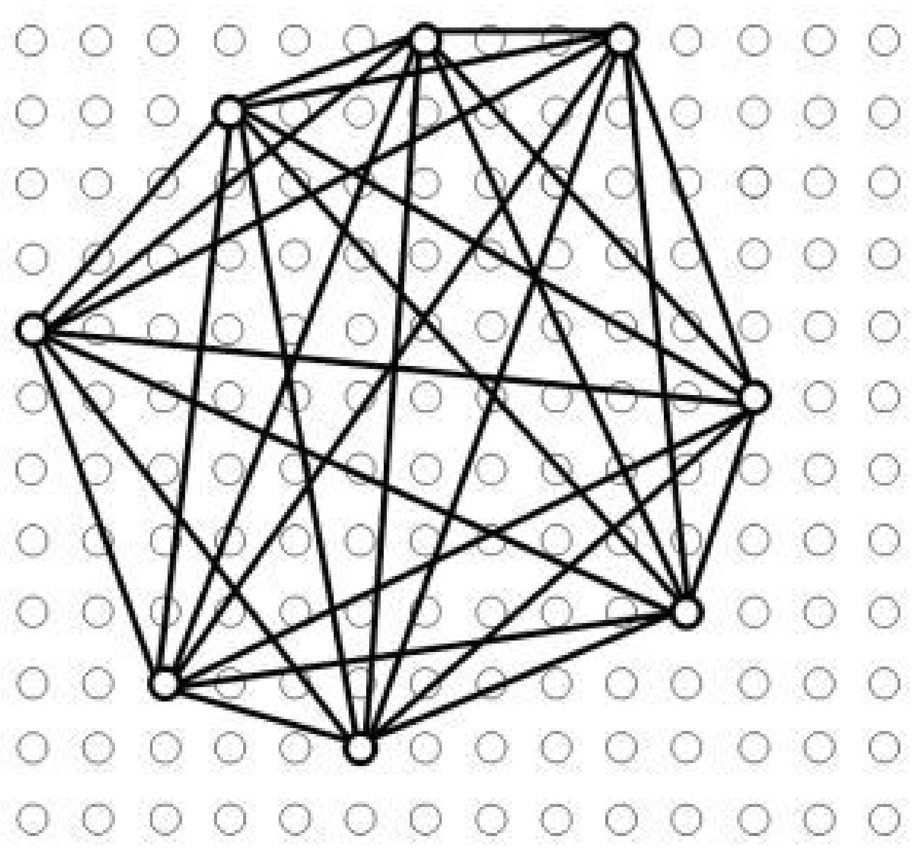
Figure 2.1: Willshaw network. A message composed of 8 nodes is displayed, the interconnections being the means of its storage in the network.

matrix W to the number of ones it would contain if every possible message was stored. For cliques of order c, the number of connections they contain is $\binom{c}{2}$. Despite the correlation between these edges, the probability of a given connection to be picked when forming a message can be estimated as $\frac{\binom{c}{2}}{\binom{N}{2}}$. Provided that the n stored messages are uniformly distributed and independent, the density of the network equates to the probability for any given connection to belong to at least one of these messages. This leads to the formula:

$$d = 1 - \left(1 - \frac{\binom{c}{2}}{\binom{N}{2}}\right)^n. \tag{2.4}$$

The efficiency of a connectionist associative memory is defined as the ratio of the maximal amount of information carried by the messages it is capable of storing then retrieving with high probability, over the total information quantity represented by its set of connection weights. For a Willshaw network with N nodes, the number of potential connections, or binary resource, is

$$Q = \frac{N(N-1)}{2}[bits]. \tag{2.5}$$

After M messages have been stored in the network, the amount of information it contains is

$$B = M\left(log_2\left(\binom{N}{c}\right)\right)[bits]. \tag{2.6}$$

Hence the efficiency of a Willshaw network is

$$\eta = \frac{2M\left(log_2\left(\binom{N}{c}\right)\right)}{N(N-1)}. \tag{2.7}$$

The maximal attainable efficiency is ln(2) [43].

The stimulation of a Willshaw network with an input request can be performed as the product of the sparse input vector by the network's connection matrix. The resulting vector then contains the output scores of the network's neurons. The score of a neuron is thus the sum of unitary stimulations it receives from the request elements it is connected to. Neurons must then be selected based on their score. Algorithm 1 defines a procedure that can be used for the recovery of a complete message from a subpart of its content. The Global Winner-Takes-All step consists in discarding all active neurons with a score below the maximum.

Algorithm 1: Message retrieval procedure in a classic Willshaw network.

```
Data: Subpart $x$ of a stored message
Result: Set of nodes $z$ active after treatment
$z = x$
Repeat
    $y = Wz$
    $z = \text{GlobalWinnerTakesAll}(y)$
while (convergence not reached
    and max. nb. of iterations not reached)
Return $z$
```

The probability of error in the retrieval of a message from the Willshaw network can be calculated when the process uses a single iteration. If only one vertex of the clique to complete has been erased, the probability of completing the message accurately after one iteration is the probability that no unit is connected to all elements of the query, other than the missing one:

$$P_{retrieve} = (1 - d^{c-1})^{N-c}. \tag{2.8}$$

The probability of error is then:

$$P_e = 1 - P_{retrieve} = 1 - (1 - d^{c-1})^{N-c}. \tag{2.9}$$

Knowing (2.4), this gives:

$$P_e = 1 - \left(1 - \left[1 - \left(1 - \frac{\binom{c}{2}}{\binom{N}{2}}\right)^{n}\right]^{c-1}\right)^{N-c}. \tag{2.10}$$

This holds for numbers of erasures c_e superior to 1, bringing:

$$P_e = 1 - \left(1 - \left[1 - \left(1 - \frac{\binom{c}{2}}{\binom{N}{2}}\right)^{n}\right]^{c-c_e}\right)^{N-c}. \tag{2.11}$$

We aim to modify classic Willshaw networks in a way that is relevant in regard to biological observations. Emphasis is put on lateral inhibition, a phenomenon that has been observed in several areas of the brain. It is notably present in sensory channels. For vision, it operates at the level of retinal cells and allows an increase in contrast and sharpness of signals relayed to the upper parts of the visual cortex [31] [44]. In the primary somatosensory area of the parietal cortex, neurons receive influx coming from overlapping receptive fields. The Winner-Takes-All operation resulting from the action of inhibitory lateral connections allows localizing precisely tactile stimuli, despite the redundancy present in the received information [45]. The same scheme of redundancy among sensory channels, and filtering via lateral inhibition, is present in the auditory system [30]. WTA is observed in the inferior colliculus and in the upper levels of the auditory processing channel.

2.4 Preventing connections between neighbor neurons

Classic Willshaw networks have no topology. Their material is constituted with a list of neurons each having an index as sole referent. There is neither a notion of spatial position in these networks, nor, a fortiori, of spatial distance. We get closer here to a real neural network, by arranging them on a two-dimensional map. In the model we propose, the respective positions of two neurons impact the possibility for them to get connected together. The considered network is composed of a number N of nodes evenly distributed along a square grid, of side $S = \sqrt{N}$. Stored messages are of constant order, meaning they are all constituted of the same number of neurons. We forbid connections between nearby neurons. To this end, we apply a threshold σ on the spatial length of a connection. Stored messages must necessarily be conform to this constraint. Each message is formed in a random manner, units being chosen iteratively. Each new element of the message is picked from the positions left available after the removal of the neighbors of the formerly selected nodes, as indicated in Figure 2.2. One can consider the introduced constraint as applied on the network's material, as the weights of a predetermined set of short-range connections will be enforced to stay null all along the network's life. During the formation of a message, it is practical to pick neurons to satisfy this constraint in a sequential manner, with a local inhibition applied on a neuron's neighborhood from the moment it is selected until the message generation is complete.

A link can be drawn between this approach and Kohonen Self-Organizing Maps, where close-by neurons encode more similar information [46]. Therefore, long-range distance separates information elements that are different in nature, whereas shorter-range distance depicts a difference in degree. Local competition is particularly relevant in this scheme.

During retrieval, the network is stimulated iteratively with a request that will most often change from one iteration to the next. Each node of the request will first stimulate every other element it is connected to. Scores are initialized with zero at the start of every iteration, and each stimulation is a unitary increment to the score of the receiver unit. For the first iteration, after the stimulation we apply a global Winner-Takes-All rule, which consists in excluding from the research scope all units that do not achieve the maximal score observed in the network. We know indeed that the neurons from the searched message will all have the maximum possible score, equal to the number of elements in the request. Once non-maximum elements are put to zero, we only pay interest in the remaining neurons during the rest of the retrieval process. Moreover, for every iteration after the first one, neurons in the new request are the only ones that can receive stimulation as the algorithm proceeds to only discard neurons from then on.

Thereafter, we can keep using the global WTA principle iteratively, but other algorithms such as Global Winners-Take-All (GWsTA) or Global Losers-Kicked-Out (GLsKO) [47] are

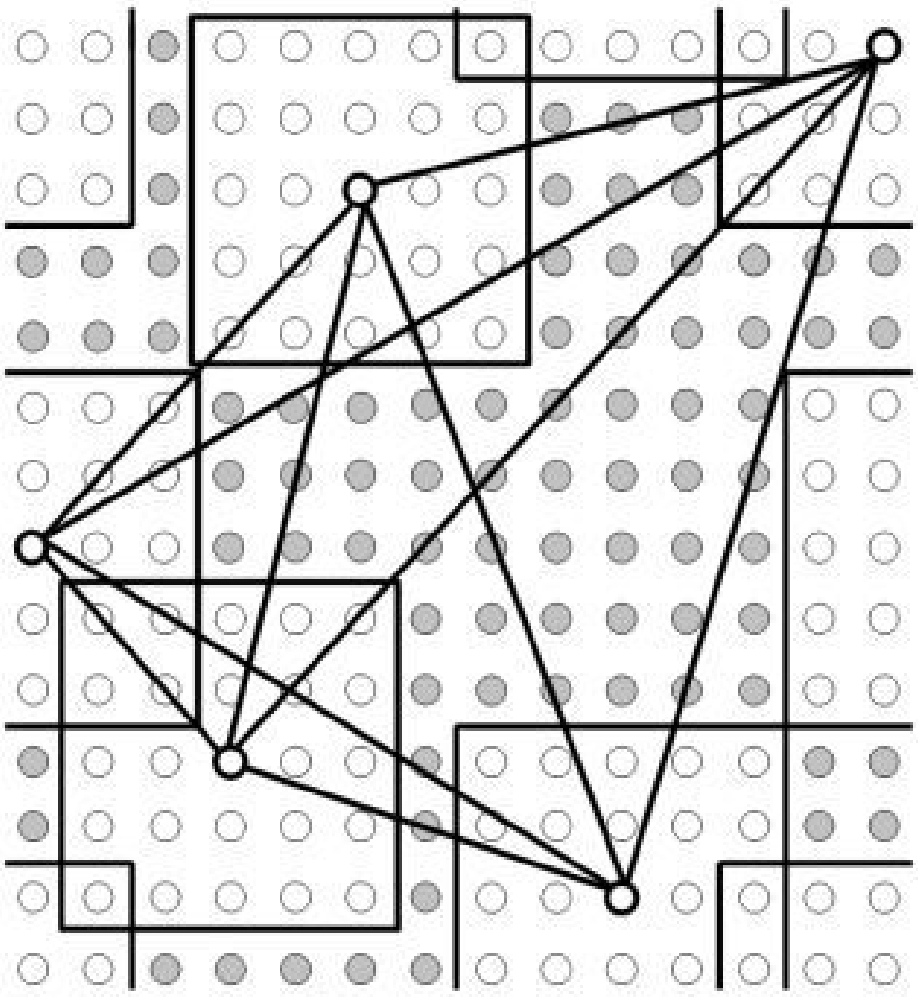

Figure 2.2: Cyclic Willshaw network with a constraint on local connections.

more efficient in discriminating the right nodes from the spurious ones that can appear during retrieval.

GWsTA relies on the calculation of a threshold score to select winner neurons. This threshold is chosen such that neurons with an activity above it are in number at least as large as the order of stored messages.

GLsKO consists in putting off, at each iteration, all the units that do not have the highest score, or a subgroup sampled randomly in this ensemble.

These two algorithmic techniques allow getting rid of an important proportion of false-positives. In the clique-based CCN, clusters play a similar role.

The iterative nature of the process means that a message retrieved as output from the network is typically reinjected in it until input and output no longer differ. A limited number of iterations is applied in the case where the network would not converge to a stable solution, an observable case in which it can oscillate between two states.

In addition to these two stopping criteria that are the maximum number of iterations and convergence, comes a third one which is the identification of a clique. Indeed, if we observe that the units still active after an iteration are in number equal to the order of stored messages, and that they all have the same score, this means it is a clique. This ensemble is then retained as the response given by the network for the current request.

Algorithm 2 shows the message retrieval procedure used in the results we present. Phase II uses GLsKO.

We experiment the storage of messages of order c in the connection matrix of the network. Messages are formed with the constraint of a minimal space between connected nodes. Two units in a message must be spaced apart at a distance superior to a minimum σ. In order to ease computations and avoid edge effects, we choose to use the L_1 distance, even though we believe this method should work using any distance. This way, when picking a node x for a message, all nodes located in a square grid centered on x, of side $2\sigma{+}1$, are excluded from the possible choices for the elements of the message remaining to be filled.

Algorithm 2: Message retrieval procedure in the modified Willshaw network with spacing constraint.

```
Data: Subpart $x$ of a stored message
Result: Set of nodes $z$ active after treatment
Phase I
    $y = Wx$
    $z$ = GlobalWinnerTakesAll($y$)
Phase II
    Repeat
        $y = Wz$
        $a$ = active nodes in $y$
        $m$ = nodes in $a$ with minimal score
        $z = a - m$
    while (convergence not reached
        and max. nb. of iterations not reached)
Return $z$
```

Moreover, this distance is applied in a cyclic way, meaning a node located on the right edge of the grid will be considered a direct neighbor of the element located at the crossing between the same line and the left edge of the grid. All four corners of the grid will also be neighbors to one another. We call the network so described a torus.

2.5 Efficient use of the memory resource

When applying a constraint σ on the minimal spacing between connected neurons, the number of potential connections in the modified Willshaw network becomes

$$Q = \frac{N(N - (2\sigma + 1)^2)}{2} [bits]. \tag{2.12}$$

Let the total number of messages one can form in it under the spatial constraint, be denoted $\overline{M}$. The entropy per message b is given by

$$b = \left(log_2\left(\overline{M}\right)\right) [bits]. \tag{2.13}$$

The amount of information contained in the network after the storage of M messages is then

$$B = bM = M\left(log_2\left(\overline{M}\right)\right)[bits]. \tag{2.14}$$

Hence the efficiency of the network with lateral inhibition is

$$\eta = \frac{2M\left(log_2\left(\overline{M}\right)\right)}{N(N-(2\sigma+1)^2)}. \tag{2.15}$$

Predicting the diversity of the spatially constrained Willshaw network given N, c and σ, is not trivial for most values of these parameters. Indeed, once one neuron has been picked among N, the remaining choice for the second element of a message is naturally $N-(2\sigma+1)^2$. However, there are then several possibilities for the number of remaining allowed components, as the inhibition areas around the first two elements can overlap. Figure 2.3 shows this behavior with $\sigma = 1$. In this case, two overlapping inhibition areas can either share one, two or three neurons. Given a fixed position for the first neuron, there are four positions for the second one that will give an intersection of one, eight that will give an intersection of two, and four positions will give intersections of three neurons.

For $N-(4\sigma+1)^2$ neurons, the two inhibition areas do not overlap, and there are $N-2\left(2\sigma+1\right)^2$ remaining possible choices for the third node.

We denote $\mathcal{A}_\sigma(N,C)$ the set of all possible ordered arrangements of C nodes among N. The number of ordered arrangements of $c=3$ nodes respecting the spacing constraint $\sigma = 1$ is thus

$$\begin{aligned} |\mathcal{A}_{\sigma=1}(N,3)| = N((N-25)(N-18) + 4(N-17) \\ +8(N-16)+4(N-15)). \end{aligned} \tag{2.16}$$

Finally, as the considered messages are unordered, we divide this formula by $c!$, here

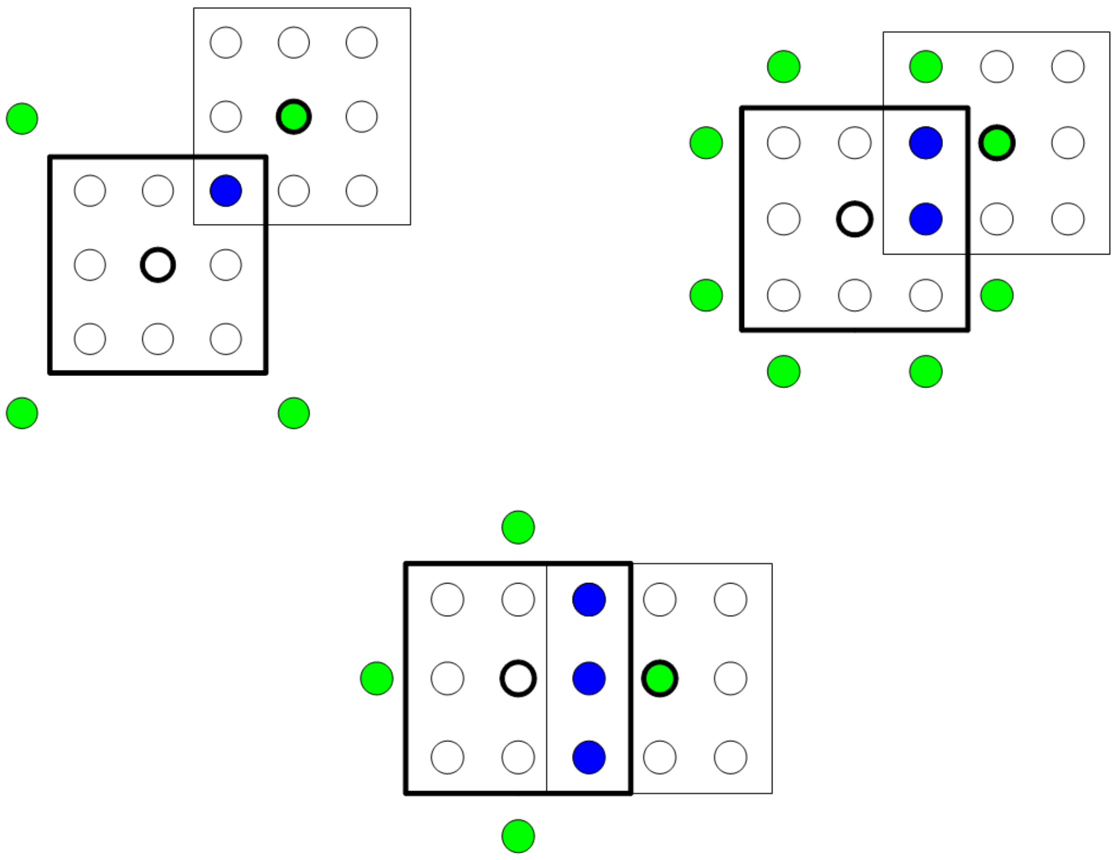

Figure 2.3: Different overlapping configurations between the inhibition areas around two neurons. Here with $\sigma = 1$, the union of the two overlapping areas can contain 17, 16, or 15 neurons. The green dots show the locations of nodes whose inhibition areas have the same number of intersecting elements with that of the central node.

six, to get the number of allowed messages:

$$\overline{M} = \frac{N^3 - 27N^2 + 194N}{6}. \tag{2.17}$$

Similarly, for other sets of parameters, the diversity of the spatially constrained network will be obtained by sum-product compositions [48] of polynomials with integer roots. As c grows, the number of overlapping configurations between inhibition areas for neurons in a message is multiplied, and finding a formula to predict the diversity becomes increasingly complex.

Indeed, a general formula for the number of possible ordered arrangements of $c = 3$ nodes among N under the spatiality constraint σ would be:

$$|\mathcal{A}_\sigma(N,3)| = \sum_{n_1} \left(\sum_{n_2 \notin V_1} (N - |V_1 \cup V_2|) \right) = N \left(\sum_{n_2 \notin V_1} (N - |V_1 \cup V_2|) \right), \tag{2.18}$$

where n_i is the i^{th} node in the arrangement and V_i is its surrounding inhibition area.

With $c = 4$ this formula becomes:

$$|\mathcal{A}_\sigma(N,4)| = N \left(\sum_{n_2 \notin V_1} \left((N - |V_1 \cup V_2|) \sum_{n_3 \notin V_1 \cup V_2} (N - |V_1 \cup V_2 \cup V_3|) \right) \right), \tag{2.19}$$

and so forth for higher values of c.

Figure 2.4 shows a possible configuration for the intersections of the inhibition areas of four neurons that can be part of the same message, when $\sigma = 2$. This illustrates the variety of intersection configurations that arise as c and σ grow larger.

For some values of σ and c, the total number of legitimate messages can be predicted by polynomial formulas that can be easy to find experimentally. Table 2.1 shows a list of such

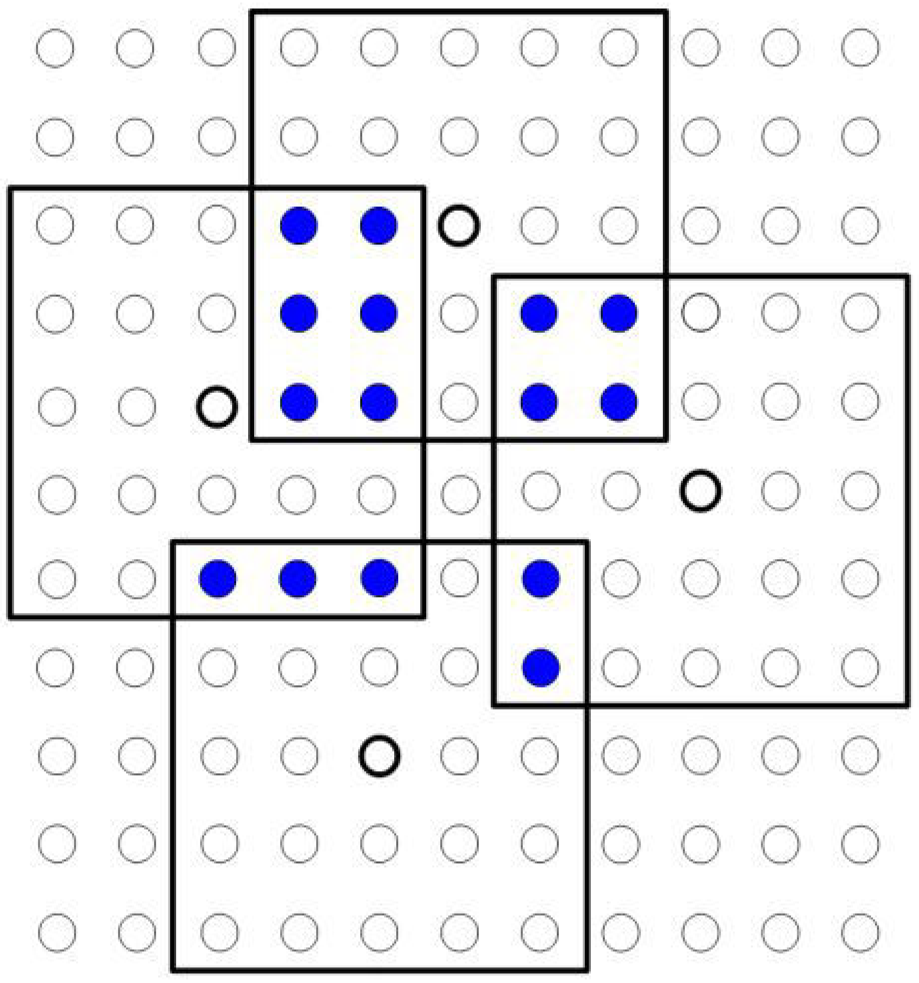

Figure 2.4: Overlapping inhibition areas for a combination of 4 selected neurons satisfying the spatial constraint $\sigma = 2$. Intersections are made of 2, 3, 4 and 6 neurons.

Table 2.1:
POLYNOMIAL FORMULAS FOR THE NUMBERS OF ALLOWED AND FORBIDDEN MESSAGES UNDER THE SPATIALITY CONSTRAINT ON CONNECTIONS, AS A FUNCTION OF THE NUMBER N OF NEURONS, FOR DIFFERENT CLIQUE ORDERS AND INHIBITION RADII. CASES WITH $\sigma = 0$ ARE EQUIVALENT TO WILLSHAW NETWORKS. THE VALIDITY OF THESE FORMULAS HOLDS FOR NETWORK SIZES SUFFICIENTLY LARGE BEFORE σ.

c	σ	**Allowed messages**	**Forbidden messages**
2	0	$\frac{N^2-N}{2}$	0
	1	$\frac{N^2-9N}{2}$	$4N$
	2	$\frac{N^2-25N}{2}$	$12N$
	3	$\frac{N^2-49N}{2}$	$24N$
3	0	$\frac{N^3-3N^2+2N}{6}$	0
	1	$\frac{N^3-27N^2+194N}{6}$	$4N^2-32N$
	2	$\frac{N^3-75N^2+1514N}{6}$	$12N^2-252N$
	3	$\frac{N^3-147N^2+5834N}{6}$	$24N^2-972N$
4	0	$\frac{N^4-6N^3+11N^2-6N}{24}$	0
	1	$\frac{N^4-54N^3+1019N^2-6798N}{24}$	$2N^3-42N^2+283N$
	2	$\frac{N^4-150N^3+7931N^2-149550N}{24}$	$6N^3-330N^2+6231N$
	3	$\frac{N^4-294N^3+30539N^2-1133958N}{24}$	$12N^3-1272N^2+47248N$

formulas, for the diversity of the constrained network as well as its number of prohibited messages, for different values of c and σ. $\sigma = 0$ corresponds to the unconstrained Willshaw network. Figure 2.5 shows the evolution of the number of legitimate messages for different constraints on connection length, for $c = 3$.

With $c = 3$ and $\sigma = 1$, the network's efficiency is given by

$$\eta = \frac{2M\left(\log_2\left(\frac{N^3-27N^2+194N}{6}\right)\right)}{N(N-9)}, \tag{2.20}$$

while the efficiency of the corresponding Willshaw network is

$$\eta = \frac{2M\left(\log_2\left(\frac{N^3-3N^2+2N}{6}\right)\right)}{N(N-1)}. \tag{2.21}$$

The ratio of these two efficiencies is thus given by

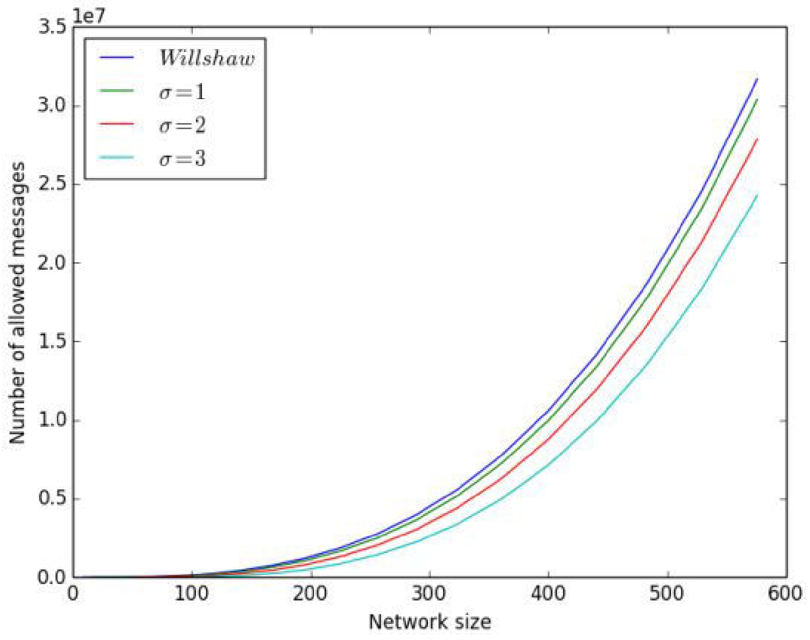

Figure 2.5: Number of allowed messages as a function of the total number of neurons, under constraints $\sigma = 1, 2$, and 3 compared with a Willshaw network, for $c = 3$.

$$\frac{\eta_{\sigma=1}}{\eta_{\sigma=0}} = \frac{(N-1)\left(\log_2\left(\frac{N^3-27N^2+194N}{6}\right)\right)}{(N-9)\left(\log_2\left(\frac{N^3-3N^2+2N}{6}\right)\right)}, \tag{2.22}$$

and is independent of the number M of stored messages. Figure 2.6 displays the evolution of the efficiency ratios between spatially constrained networks with $\sigma = 1$, 2, 3, and the unconstrained Willshaw network, as a function of the number of neurons in the network, for messages made of three neurons. It shows that, although the spacing constraint can not be applied on too small networks, for sufficiently large networks, prohibiting shorter connections comes with an increase in efficiency, as the ratio is superior to one. Larger values of σ are associated with larger gain in efficiency, and for a given σ the best improvement over a Willshaw network comes for the smaller networks where this constraint applies, *i.e.*, where it does not block the vast majority of Willshaw messages. The decay of the ratio when the network size increases is due to the fact that the constraint then prohibits a smaller proportion of connections, making the difference with a Willshaw network less noticeable.

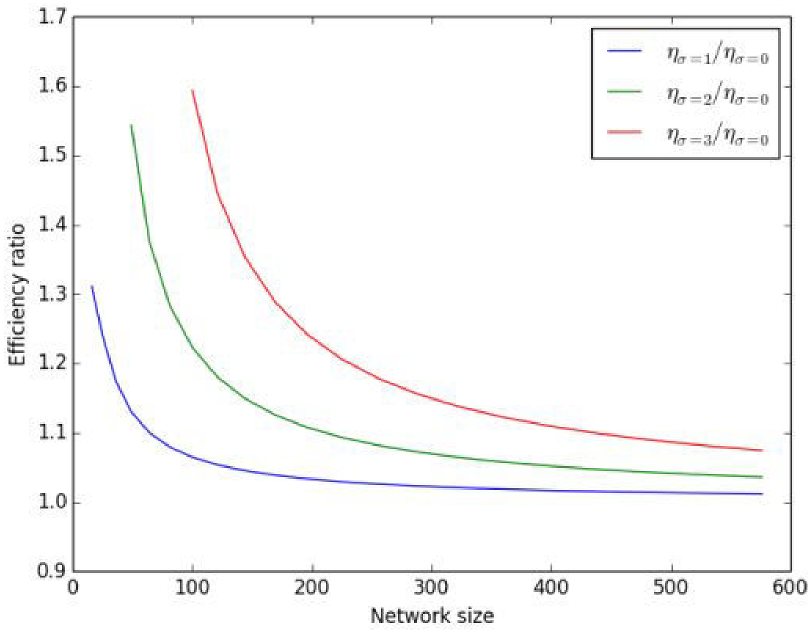

Figure 2.6: Evolution of the ratios of the efficiency of the networks with spatial constraints $\sigma = 1, 2$, and 3 over the efficiency of a Willshaw network, with increasing network size, for $c = 3$.

Figure 2.7 shows the evolution of the same efficiency ratios for cliques of order 4. The improvement is then slightly lower for constant σ and network size, in comparison to the case $c = 3$. The ratio remains superior to one however. This tends to show that the spacing constraint can be more beneficial, in terms of gained efficiency, for shorter messages.

The efficiency is a measure of the amount of information one can store for a given amount of available memory. The improvement in efficiency brought by the spacing constraint on connections means that the reduction of used material is more significant than that of the quantity of information carried by messages.

2.6 Pattern completion

We now pay interest specifically in the pattern retrieval ability of the modified Willshaw network, as compared to the classical model. During retrieval, only a sample from the nodes of the complete message are stimulated, the inputs are subparts of stored messages. Units

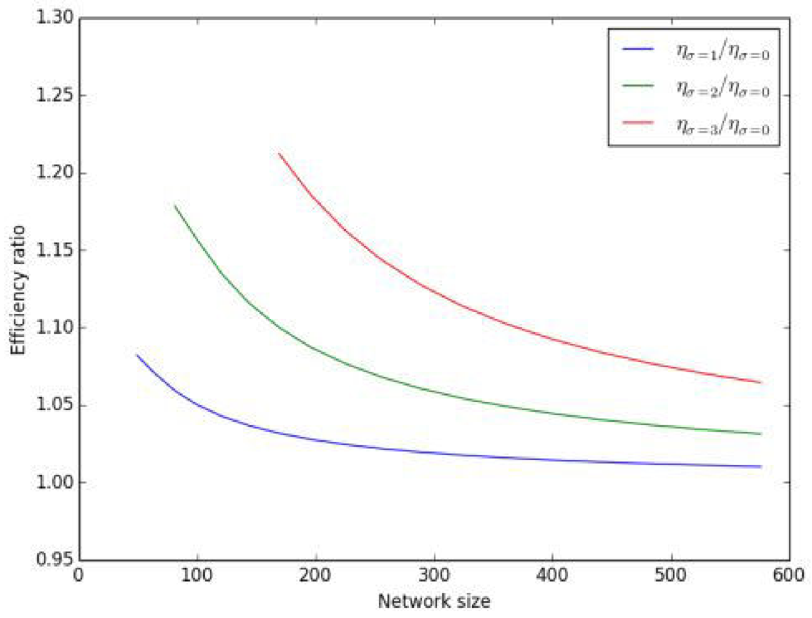

Figure 2.7: Evolution of the efficiency ratios for $\sigma = 1, 2$, and 3 over the Willshaw efficiency as a function of the number of neurons in the network, for $c = 4$.

that are close to elements of an input will not reach the maximum score in the network, and will therefore be ruled out after the first WTA operation. During the second phase of the algorithm, nodes in the vicinity of input neurons will also be more likely to reach a low score if they are activated, and to be discarded. Hence, the local inhibition used initially during the creation of messages impacts the retrieval process as well.

We pay interest in the network's ability to return the exact memory associated with a request. Hence every difference, even marked by a single unit, between the expected pattern and the network's output is counted as an error.

We measure the performance of the network as the ratio of the number of successfully retrieved messages over the total number of requests.

Various parameters can impact this performance, albeit to different degrees:

- the length S of the grid's side

- the number M of stored messages

- the minimal space σ between two elements of a message

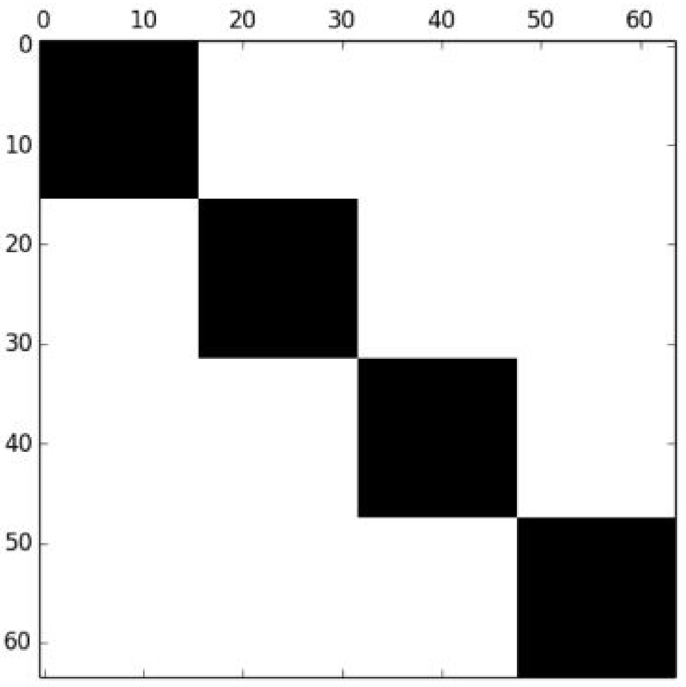

Figure 2.8: Matrix of the potential and forbidden connections in a clustered clique network with 4 clusters of 16 fanals each. The element at the crossing of a line i and a column j represents the connection between neurons n_i and n_j. White cells represent allowed connections, black cells correspond to forbidden ones.

- the order c of stored messages

- the number of erasures c_e applied on stored messages to obtain the corresponding request messages

The behavior of this network is interesting in relation to Willshaw networks and clustered cliques networks, as it is close to a classic Willshaw network and displays the added feature of prohibited connections as observed in CCNs. This modification can be viewed as a form of sliding-window clustering.

Figures 2.8 and 2.9 represent the matrices of allowed and forbidden connections in a clustered clique network with $\chi = 4$ clusters comprising $l = 16$ neurons each, and in a modified Willshaw network of side length $S = 8$ under constraint $\sigma = 1$, respectively. The two networks have the same number of neurons, and comparable numbers of allowed and forbidden connections. Indeed, the number of potential non-oriented connections in the

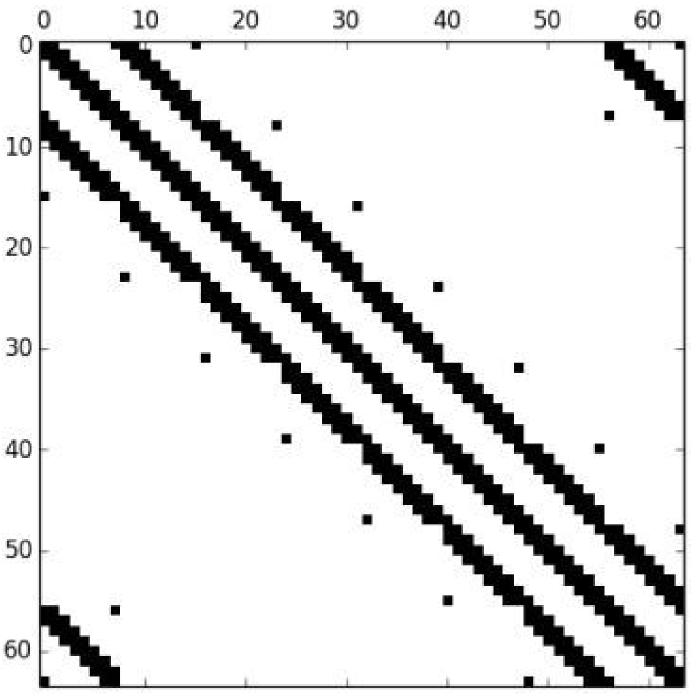

Figure 2.9: Connection matrix for a modified Willshaw network with side length $S = 8$ and $\sigma = 1$.

clustered clique network is given by

$$Q = \frac{\chi(\chi - 1)l^2}{2}[bits], \tag{2.23}$$

which here gives 1,536 allowed connections, against 480 ones forbidden due to the clustering constraint. On the other hand, from (2.12) we have 1,760 allowed connections in the spatially constrained Willshaw networks, and 256 forbidden ones. The major difference is the potential overlapping between the inhibition areas around different neurons in the modified Willshaw network. It follows that in this network, two neurons n_i and n_j can be both prohibited from connecting to a third one n_k, and yet be allowed to connect together. To the contrary, if in a clustered clique network, connection weights w_{ik} and w_{jk} are forced to remain at 0, then necessarily w_{ij} will be as well.

In a first series of experiments, we focus on the ability of the network to first store independent, identically distributed messages, and then complete them properly when probed with partial cues. For every configuration of the network, messages and requests we test, we store a set of thousands of messages in the network. These messages are generated randomly following the local inhibition pattern described in section 2.4. We then request it with the full set of queries associated with stored messages.

For each network size, we observe that there is an optimal value of the minimal distance σ, that lowers the most significantly the error rate, as compared to the corresponding Willshaw network without constraint on local connections. For a given minimal distance, the reduction in error rate depends on the number of stored messages, with an optimal number of messages which is a function of the network size. For cliques of order four and with two erasures, the maximal reachable improvement is close to 15%, and seems to be the same for all network sizes. In this configuration, the minimal distance bringing the best performance is approximately the third of the network side.

The evolution of the retrieval error rate as a function of the number of stored messages is slower with the appropriate constraint on connections than for a classic Willshaw network, as can be seen in Figure 2.10.

Figure 2.11 shows a similar comparison, this time between the modified Willshaw network with constraint $\sigma = 5$ made of 400 neurons, and an unconstrained Willshaw network with 335 neurons. Because of the reduction in the number of connections when $\sigma = 5$, the two networks have almost the same binary resource. Indeed, the Willshaw network has 55.945 connections while the constrained one has 55.800 possible connections, despite having more neurons. Even though the modified network has a slightly lower footprint, the improvement is even more noticeable than for the comparison with equal size of the neurons sets. In fact for $1,500$ stored messages, the modified network gains around 50% in error rate over the Willshaw network. Like the CCNs approach, this shows the interest of spreading the binary resource across a larger set of units with constrained connectivity, as opposed to allowing any two neurons to link.

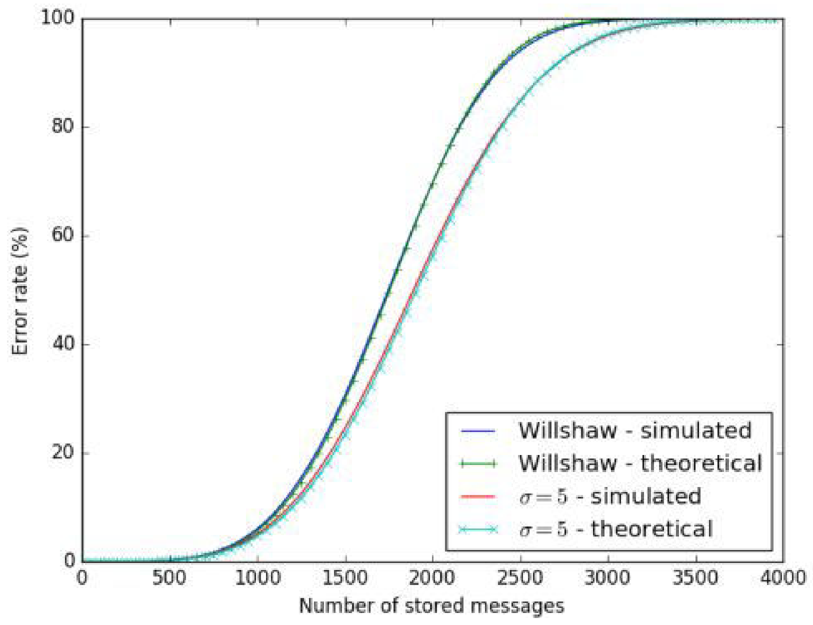

Figure 2.10: Evolution of the retrieval error rate with and without constraint $\sigma = 5$ in a network of side length 20 with 400 neurons, stored messages of order 6 and 1 erasure applied to form corresponding requests, with 1 iteration.

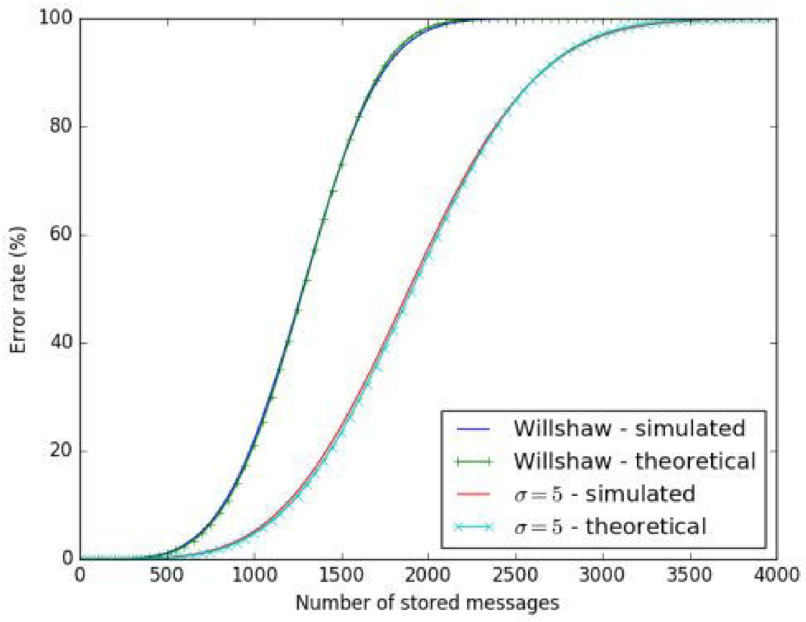

Figure 2.11: Evolution of the retrieval error rate after one iteration as a function of the number of stored messages, in a classic Willshaw network with 335 neurons, and in a modified Willshaw network of side length 20 with 400 neurons, with constraint $\sigma = 5$. Stored messages have order $c = 6$ and associated queries are obtained by erasing one vertex. The two networks have close numbers of possible connections.

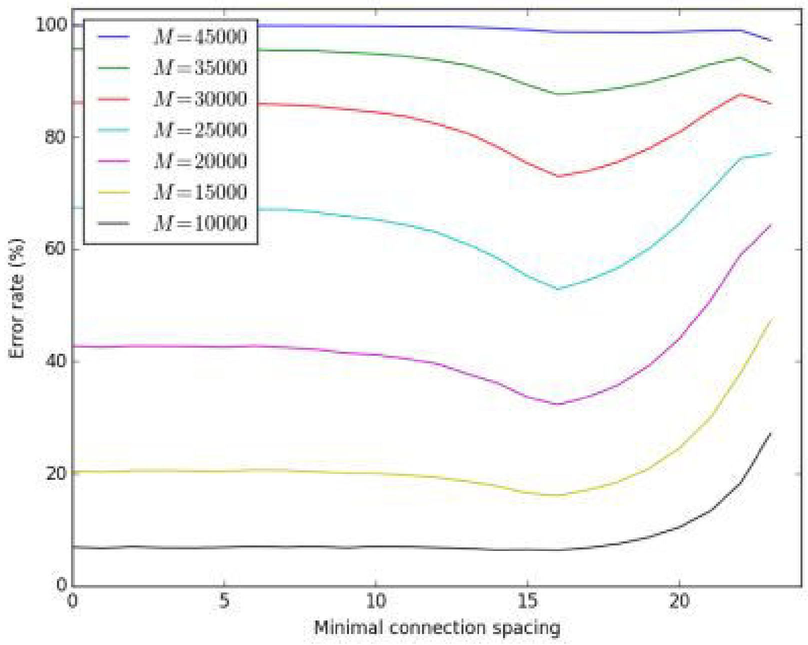

Figure 2.12: Minimal connection distance effect on performance in a modified Willshaw network with local inhibition, made of 2500 neurons. Stored messages are of order $c = 4$, and $c_e = 2$ erasures are applied to form corresponding requests. Seven different numbers of stored messages are tested. The case where minimal spacing $\sigma = 0$ corresponds to a classic Willshaw network.

For a constant number of stored messages, the graph of the error rate as a function of σ is characterized by a progressive decay down to a minimum, followed by a rapid growth for upper values of σ, as shown in Figure 2.12.

This can be explained by two phenomena. On the one hand, the prohibition of a growing part of the possible connections gradually decreases the probability of a "false message", characterized by the intrusion of a spurious node in the output. The existence of a node that is connected to all elements in a request, yet is not part of the corresponding message, will potentially cause an error. In fact, forbidding some connections has the effect of reducing the number of concurrent nodes susceptible to cause errors. We can estimate the mean number of concurrent nodes remaining after the choice of k neurons of a message:

$$N_c(\sigma, k) = N \left(1 - \left(\frac{(2\sigma + 1)^2}{N}\right)\right)^k . \tag{2.24}$$

Indeed, $\frac{(2\sigma+1)^2}{N}$ represents the probability for a random neuron to be in the inhibition neighborhood of another previously picked neuron, thus $1-\left(\frac{(2\sigma+1)^2}{N}\right)$ is the probability for not being in it. $\left(1-\left(\frac{(2\sigma+1)^2}{N}\right)\right)^k$ is the probability for a neuron not to be in any of the inhibition neighborhoods of k previously selected neurons.

The corresponding number of nodes blocked by the constraint on connections is thus, on average:

$$N_b(\sigma,k) = N\left(1-\left(1-\left(\frac{(2\sigma+1)^2}{N}\right)\right)^k\right). \tag{2.25}$$

This explains the decay phase in error rate observed for the first values of σ. Let us note that it comes with a decrease in the diversity of messages, namely the total number of different messages that can be stored in the network. Following this decay, the decrease in the number of concurrent nodes has another effect: the reuse of connections by different messages becomes more frequent as the choice for possible connections gets reduced. This comes to counteract the former phenomenon and raises the error rate.

The density of the modified network after the storage of n messages can be calculated by

$$d = 1-\left(1-\frac{\binom{c}{2}}{Q}\right)^n, \tag{2.26}$$

that is

$$d = 1-\left(1-\frac{c(c-1)}{N\left(N-(2\sigma+1)^2\right)}\right)^n. \tag{2.27}$$

As for the Willshaw network, we can calculate the probability of error of the modified network after one iteration of decoding. Given queries where c_e nodes have been removed, the probability of retrieval after one iteration can be estimated by:

$$P_{retrieve} = (1-d^{c-c_e})^{N_c(\sigma,c-c_e)-c_e}. \tag{2.28}$$

One can then deduce the probability of error:

$$P_e = 1 - (1 - d^{c-c_e})^{N_c(\sigma, c-c_e)-ce}. \tag{2.29}$$

Now referring to (2.26), this leads to:

$$P_e = 1 - \left(1 - \left[1 - \left(1 - \frac{\binom{c}{2}}{Q}\right)^n\right]^{c-c_e}\right)^{N_c(\sigma, c-c_e)-c_e}, \tag{2.30}$$

which holds for lower values of σ. For higher values, however, the global density is no longer a proper estimator of the probability of spurious connections. Given $c - c_e$ message elements, the local connection density between these nodes and the restricted ensemble of allowed neighbors they can all be potentially connected to, is then higher than the average density over the whole network.

Figure 2.13 shows how the network density grows faster as messages are stored in the network, than for a classic Willshaw network with equal number of neurons. This is because of the decrease in number of possible connections due to the spacing constraint. When the number of connections of the compared classic and modified networks is close, the two densities evolve at a similar rate however.

Besides, we observe that the maximal improvement in performance, for given values of c and c_e, does not considerably vary as a function of the network size. This can be explained by the fact that the minimal distance giving the best performance is approximately proportional to the side of the network. Consequently, the proportion of neurons in the network that cannot be connected to the $c - c_e$ neurons in the request remains more or less the same for different network sizes, with the optimal minimal distance.

The benefit brought by the constraint on connections tends to be stronger for smaller numbers of erasures. For erasures of about half the units of the messages, the maximum gain will be lower, yet for a high amount of erasures the performance may be more noticeably enhanced by the added constraint. The performance improvement over a classic Willshaw

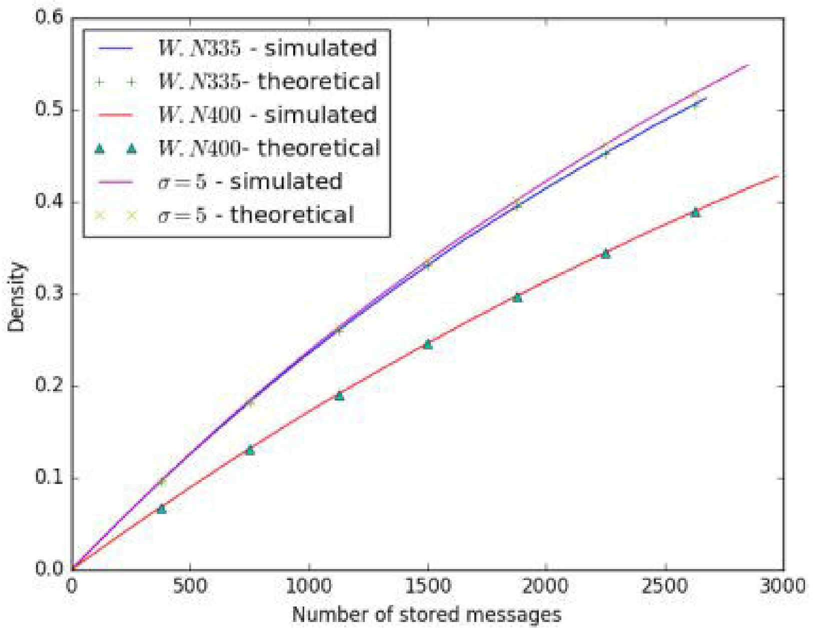

Figure 2.13: Evolution of the density for ordinary Willshaw networks with 335 and 400 neurons, and for a modified Willshaw network of side length 20 with 400 neurons and $\sigma = 5$, with stored messages of order 6.

network also depends on the number of messages stored in the network. It reaches a peak for a certain number of stored messages, and then decays when additional messages get stored. The maximal improvement tends to be reached earlier during storage for higher numbers of erasures, as illustrated by Figures 2.14 and 2.15 for a network with 400 neurons, messages of order $c = 6$, and numbers of erasures c_e ranging from one to five. Figure 2.16 shows that for a larger network with 900 neurons, this arrangement is respected for the most part, with the exception of the case where $c_e = 1$, for which the peak in performance improvement occurs for a lower number of stored messages than for $c_e = 2$ or $c_e = 3$. Figures 2.14 and 2.15 also show that the maximum number of iterations applied during retrieval has a varying effect on performance improvement, depending on the number of erasures applied to form requests. With 400 neurons, increasing the number of iterations has a clear effect on performance for $c_e = 3$ and $c_e = 4$, more so than for $c_e = 1$ and $c_e = 2$.

When comparing networks with the same number of neurons, the greatest performance

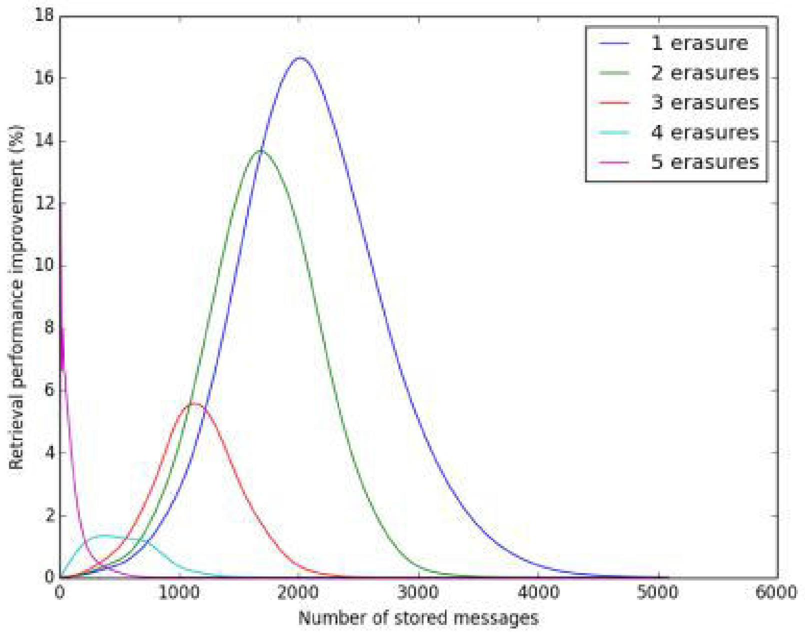

Figure 2.14: Maximal improvement obtained over a classic Willshaw network made of 400 units with messages of order 6, using a single iteration of decoding.

improvement is most often observed over a classic Willshaw network and a number of stored messages originally giving an error rate ranging from about 40%, up to 70%. The performance gain is then often close to 15%.

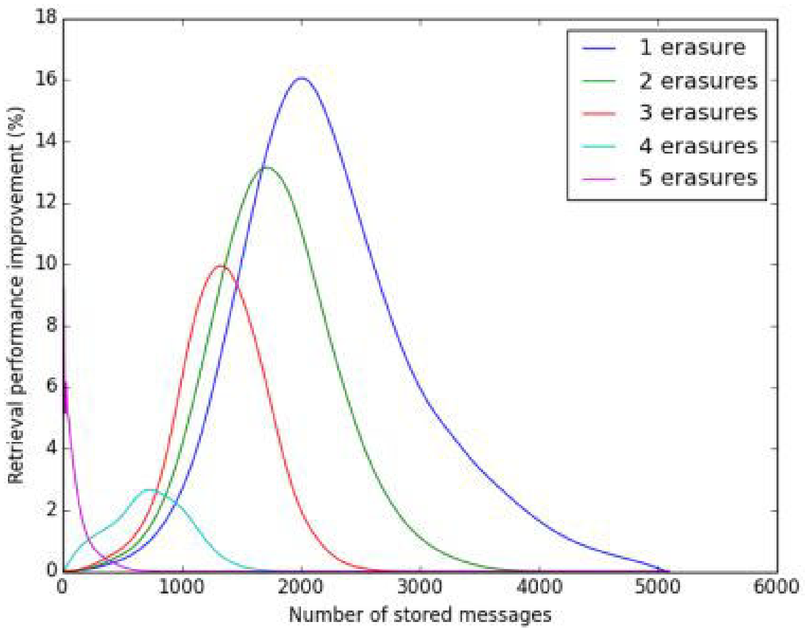

Figure 2.15: Maximal improvement obtained over a classic Willshaw network of 400 neurons with messages of order 6, using a maximum of 3 iterations.

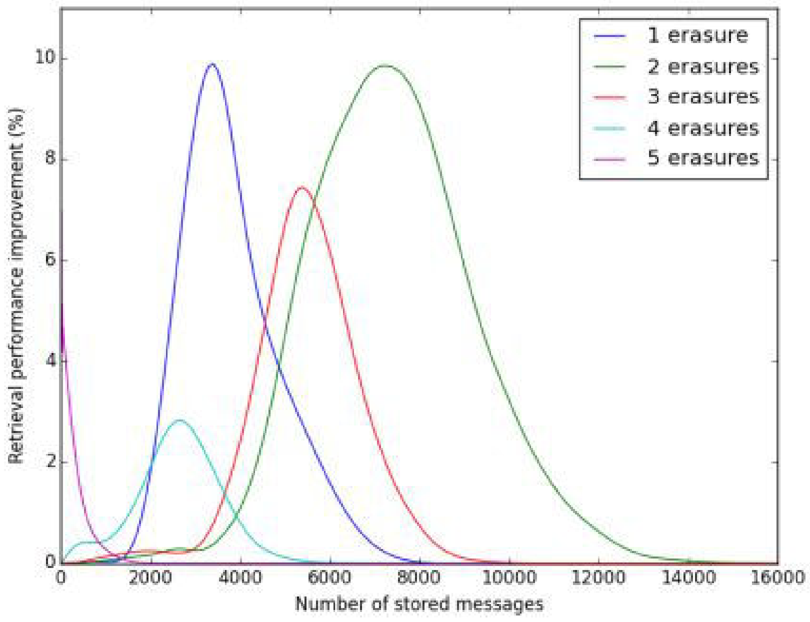

Figure 2.16: Maximal improvement obtained over a classic Willshaw network of 900 neurons with messages of order 6, using a maximum of 3 iterations.

Chapter 3

Robustness of Deep Neural Networks to Erasures in a Reading Task

3.1 Introduction

The attempts to process visual input using artificial neural networks with a serial, hierarchical architecture, date back to the seminal work of Hubel and Wiesel that revealed the visual cortex of mammals was organized in such a way [49,50]. These early findings also led to new models of reading based on overlapping, compositional neural structures supporting letters, syllables, words and sentences. A few decades later, the amount of training data and computing power has raised to a level sufficient to allow training of layered neural networks on high-dimensional visual data. In particular, the translation-invariant Convolutional Neural Networks have proven very efficient on a vast panel of tasks. Moreover, they have been found to mimic the visual cortex in their level of performance, and internal representation geometry. We experiment with such a network on a word reading task where only a small percentage of the pixels of a word are displayed, using the same testing conditions as in a

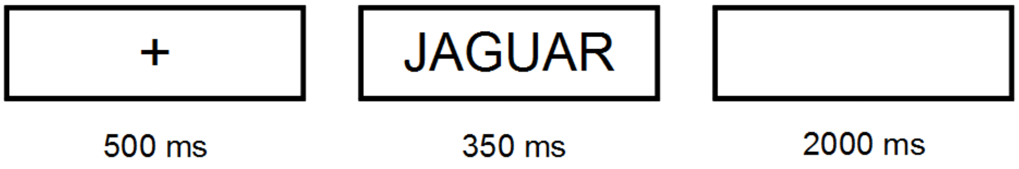

Figure 3.1: Display sequence used for each word during the reading experiment.

previous psycholinguistic study.

3.1.1 Visual recognition of partially displayed words

An experiment was formerly run on 23 human subjects confronted with partial displays of words [51, 52]. The volunteers were asked to signal through a computer keyboard whenever they thought they had recognized a word, at display rates always inferior to 30% of the pixels. They were then to type in the word they believed to have seen. The display rates came in ascending order, starting at 0.25% and were incremented each time all words in a sequence had been seen. The first increment was of 0.25%, and the following increments were of 0.5%. Participants signaled their recognition of all words, accurately or not, before reaching the 10% mark. The order of the sequence of words was randomly reset at each display rate increment, with correctly recognized words getting removed from the display list right away. A fixation cross was displayed for 500 milliseconds prior to the display of a word, centered on this location, for 350 milliseconds. A white screen was then displayed for 2000 milliseconds, before repetition of this tri-fold sequence with the next word. Figure 3.1 illustrates this protocol. Word length ranged from 4 to 9 letters, all letters were in upper-case and the used font was Courier.

The words were chosen to be sufficiently well balanced in terms of length as well as frequency in the French lexicon. The results of this study showed the conjunct role of several parameters in word recognition, which can be arranged in three categories. The first category is constituted by bottom-up factors, *i.e.*, characteristics of the display triggering recognition. Other studied parameters are linguistic factors, such as the frequency of a word and the size of its orthographic neighborhood in the lexicon, and subject-related factors, *e.g.*, age and level of education. The prominent factors in the specifics of recognized visual displays happened to be the display rate of the whole word, followed by the specific display rates of lines and curves in the letters. Regarding lexical factors, the frequency of a word had a significant impact, with high-frequency words being recognized at lower display rates than lower-frequency words, on average. Longer words (8-9 letters) were also usually recognized at lower display rates than mid-length words (6-7 letters), themselves being recognized faster than short words (4-5 letters). As for the human subjects, age and level of education were the most determinant factor affecting word recognition performances.

3.1.2 Pre-existing models of reading

One well-known phenomenon involved in the recognition of written letters is the Word Superiority Effect (WSE), first evidenced by the work of Reicher [53] and Wheeler [54], in which human subjects were shown to recognize briefly displayed words more easily than random sequences of letters. This suggests that images of words may stimulate higher-level, conceptual feature detector units in the brain, when inconsistent strings of letters trigger more bottom-up processing. Indeed, when presented with a known word, the brain can exploit stored visual memory of this word and its sub-parts, *e.g.*, syllables, while it has to perform some additional low-level exploration of the input when it is either unknown or simply irrelevant. Several models of visual text recognition have been proposed to account for the WSE. Among these, the Template Matching Model [55] postulates that several versions of each object are stored in memory, such as different possible variations of a letter. An incoming pattern would then be recognized as a known object if it matched precisely with one of its stored templates. The Prototype Matching Model [56] brings a more flexible approach, which no longer relies on exact matching between input and memory elements. One prototype is stored for each object class, and pattern recognition here occurs when an input object displays sufficient similarity with one of the prototypes. As a result, such a system can be robust to changes in font, size and angle, as well as the variations found in handwriting. Some of the proposed models also use hierarchical treatment to emulate the combination of bottom-up and top-down processing at work during reading. The Feature Analysis Model [57] assumes that words-like patterns are detected, then decomposed recursively. The set of sub-parts are then compared with memory elements, and global recognition happens through distributed classification of components paired with structural rule matching. The Interactive Activation Model (IAM) [58] of McClelland and Rumelhart promotes a hierarchical treatment of visual text through successive levels associated with strokes, letters, and words. The process is interactive in that top-down retroaction, both excitatory and inhibitory, operates concurrently with bottom-up flow. For instance, input strokes activate letters that feature them, and inhibit letters that do not. In turn, letters propagate activation and inhibitory signal to the words layer. Strongly activated words then stimulate back the letters they contain and inhibit the other ones. Lateral inhibition also occurs between elements at the same level. The IAM thus gives a plausible explanation to the Word Superiority Effect, in that recognizing a specific letter in a word is helped by the retroaction from the activated word, which is lacking in the case of non-words displays. Recent variations of this model, such as the Bimodal Interactive Activation Model (BIAM) [59], the Bilingual Interactive Activation (BIA) [60] and its extension BIA+ [61] add language-related levels on top of the original chain, *e.g.*, lexical and phonological.

3.1.3 Deep Convolutional Neural Networks

As hierarchical processing has become a key aspect in the cognitive models of reading, it has been the object of lots of interest in the computer science communities. Several attempts have been made to compute complex mathematical functions through the combination of multiple layers of neurons, such as the Multi-Layer Perceptron [62] and Fukushima's Neocognitron [63]. This decades long endeavour has recently given rise to the field of Deep Learning. Once inefficient on most high-dimensional Machine Learning problems, it has now established state-of-the-art performances on a vast array of applications and data-sets, including the majority of visual pattern recognition problems. This late success is due in part to the raise in computing power available, with the advent of modern Graphical Processing Units (GPUs) and machine clouds. From the algorithmic point of view, it has been demonstrated that even shallow feed-forward neural networks could approximate any possible mapping between sets of inputs and outputs, with a precision depending on the number of units in the network. This is known as the Universal Approximation Theorem [64].

Deep learners rely on successive layers of formal neurons. The parameters of the network are arranged into matrices of inter-layer connection weights, and vectors of additive activation biases. Activity propagates from one layer to the next by the product of an input vector with the local matrix of synaptic weights, the result of which is added to the biases vector of the output layer before passing through an activation function, typically linear or sigmoïdal. In the end of the chain, a scalar cost is computed based on the difference between the output of the network and its assigned objective. This cost is then differentiated with respect to each parameter. The resulting gradient is used to concurrently adjust the whole set of parameters with respect to their individual contributions to the loss.

Convolutional Neural Networks are a particular type of deep learners, which rely in part on classic formal neurons in the upper layers, but use maps of convolutional filters in the first levels of treatment. These filters are used in order to reproduce the characteristics of translation invariance as observed in human vision. A visual feature detector, once learned, will indeed fire regardless of the position of the object in the visual field.

It is also a practical solution for learning features in large images, as it can drastically reduce the number of parameters necessary in the first layers of the chain of treatment, hence lowering the amounts of required memory and calculations. In the images data-set used in this work, the number of pixels per image exceeds 48.000, and renders irrelevant other solutions sometimes used to develop invariance to translation, such as providing a classic Multi-Layer Perceptron with several laterally shifted versions of each training image.

3.1.4 Previous study of the representation geometry and biological plausibility of CNNs

Since their introduction with LeNet5 on the toy data-set MNIST [11], Convolutional Neural Networks have set a long track of record breaking performances on various Machine Learning challenges. Notably, the AlexNet [2] model of Krizhevsky *et al* drew much attention by overcoming its competitors by a large margin in the ILSVRC contest, on the high-dimensional ImageNet data-set of natural images comprising 1000 categories. Shortly thereafter, Zeiler and Fergus presented an effort to visualize and understand the deep feature detectors developed by AlexNet, through the use of deconvolution filters [65]. This visualization put in evidence that Krizhevsky's model developped primarily high-frequency and low-frequency filters, and neglected mid frequencies. Higher-level features also displayed aliasing. By raising the convolution sampling rate and using smaller kernels, Zeiler et Fergus found an architecture whose features were exempt of these artefacts, and which performed better on ImageNet. In this line of work, several attempts have been made to understand the inner working of a learning CNN and enable visualization of the feature detectors developped in higher layers. This often involves the gradient ascent search for a bounded input bias, or image prior, that maximizes the activity of a chosen hidden unit in a trained deep network whose all other parameters remain fixed [66–68].

Apart from visualizing features to get insights on the inner working of deep networks, another research approach has been used that compares the activity patterns of deep layers in CNNs with those of the human visual cortex, in reaction to the same input images. This method has revealed the relatively high level of functional biomimetism of certain deep convolutional architectures, and, therefore, has given some credit to the biological plausibility of these systems. Among these efforts, Cadieu *et al* notably compared the inner activations of AlexNet and its updated version by Zeiler et Fergus, with activity patterns in the different levels of the ventral stream in the visual cortex of human subjects, when both groups were confronted with the same images of different types of objects, such as car, animal, or face [69]. In their study, object pose varies among images of the same category, and objects are cropped from their original image background to be superposed on a randomly selected one, to ensure that the task would focus on the core recognition of the object [70]. The representations are typically sampled at the level of the penultimate layers of the studied models, right before the output level associated with the class representation. Depending on the model considered, said output would be a grandmother cell layer topped with a softmax or logistic regression operator [71, 72]. A linear Support Vector Machine is used to assess the degree of separability of the groups of feature activation vectors associated with the different classes, using hyperplanes. The study demonstrates that the network of Zeiler and Fergus has high-level representations that are almost exactly as easy to separate class-wise as those of the inferior temporal (IT) cortex, where AlexNet is just slightly below. The inner representations of both models are significantly more easy to part than in-vivo measurements in V4, the major area preceding IT in the ventral stream.

The study also involves artificial systems with fixed, hand-engineered features, whose behavior do not appear to resemble as much that of higher levels of the visual ventral stream on such a complex, high-dimensional images data-base. This underlines the interest of using a learning process such as gradient descent, that adapts the parameters finely to the data and task at hand. Several authors have argued in favor of the biological plausibility of gradient descent, among which [73] showed its mathematical equivalence with a combination of homeostatic plasticity and Spike Timing Dependent Plasticity (STDP), a corollary of Hebb's update rules. Other biologically plausible weights optimization methods have been proposed, such as Contrastive Hebbian Learning [74] and eXtended Contrastive Attractor Learning (XCAL) [75], to name but a few.

As a potential, supplementary indicator of biological relevance, it was put in evidence by Yamins *et al* that the performance of a given deep convolutional architecture is correlated with its level of IT prediction, *i.e.*, the better a specific CNN architecture performs, the more the activity patterns of its deeper layers resemble those of the infero-temporal lobe of human subjects on the same task [76]. To the contrary, the level of performance of other types of classifiers does not appear to correlate as significantly with their behavioral similarity to IT. This suggests that the visual cortex has been shaped by a dual constraint, having to ensure good recognition performance using a hierarchical cascade of representations. This hierarchical organization is likely itself the result of an evolutionary optimization that combines the low computational cost, and efficient reuse of material, resulting from a factorization of modular visual functions [77].

Following this line of ideas, Khaligh-Razavi *et al* have shown the importance of supervision to emulate IT behavior using gradient descent learning [78]. Supervision here designates the method that uses the category of each image to define the output vector that the network is trained to produce given that image. It is the typical method employed in most comparative studies of deep networks and IT. A deep network can also be trained in unsupervised fashion, and is then typically led to reproduce its input image on its output layer. Such a network is called a deep auto-encoder and, provided the network's structure and training algorithm are adequate, can extract features relevant with the structure of the training data. Because it is not assigned to categorize inputs, however, it tends to learn the specific details of individual samples in place of the low-frequency patterns that help define categories, and discriminate between them. As a result, it turns out that weakly supervised models do not have an activation behavior as category-specific as that of IT, and have a significantly different representational geometry.

Here one of the brain areas whose function the artificial network is trained to emulate, is the visual word form area (VWFA), located in the occipitotemporal cortex [79]. However, the focus is put on comparing human subjects and our network on a functional level, that is, determining which variations of the task are associated with a high correlation between the performances of the two groups, and to the contrary, which specific display settings reveal a difference in behavior.

3.2 Methods

3.2.1 Data-set pre-processing

Knowing pixels could only be displayed inside one of up to 9 letter bounding boxes of 77*70 pixels, the inter-letter spacing is ignored and each image is represented as a sparse binary vector of length 9*77*70 = 48.510. For words shorter than 9 letters, a padding is applied at the right end of the vector, as if the word was 9 letters long but with strictly no pixel displayed in the bounding boxes corresponding to the last letters. Hence, unlike what was done during the experiments with human subjects, no centering is applied, the first element of each image vector always represents the value of the top-left pixel of the displayed word's first letter. Knowing that the learning process of the first layers of a Convolutional Neural Network is designed to develop invariance to translation of input images, it is not altered by this padding choice. On the other hand, the centering of words displayed to human subjects was aimed at optimizing conditions relative to visual attention, ensuring that each subject would have the least ocular effort to do to get a global picture of the shortly-displayed word. A fixation cross was used before each display to recall the central position of the word.

3.2.2 The convolutional architecture employed

The Convolutional Neural Network used in the experiment contains 3 convolution layers made of 10, 20 and 40 convolutional feature maps, respectively. Feature kernels are square grids of dimension 11*11 pixels for all 3 layers. Each of these convolution layers is followed by a Rectified Linear Unit (ReLU) activation function, which zeroes out negative output values [80]. The third layer is followed by a 2*2 max-pooling operator which retains the

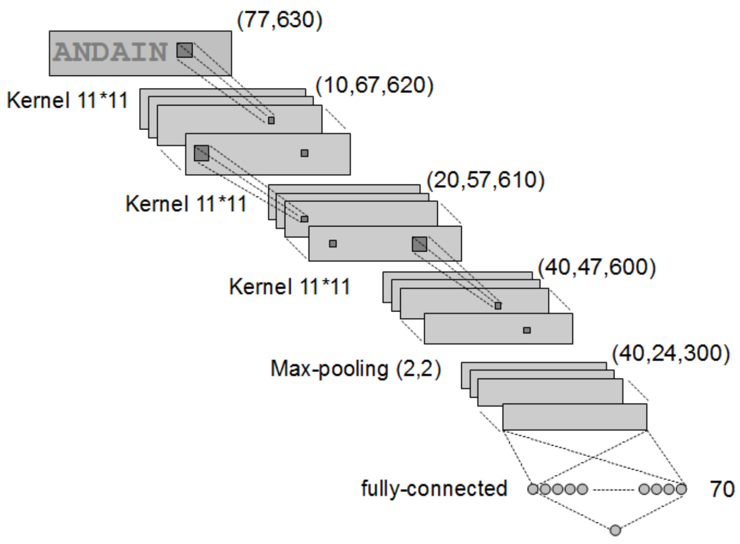

Figure 3.2: Architecture of the Convolutional Neural Network used.

maximum value out of every four units, and reduces the vector size accordingly. A dropout process is applied on the output of the last max-pooling, with an exclusion probability of 0,25 [81]. This means that for each image, an ever changing, random sample of 25% of the units are put to zero in the activation vector resulting from the max-pooling. The 3-layer convolutional stack is followed by a perceptron with one hidden layer of size 70 with a sigmoïd activation, on which a dropout of 50% is applied. The last fully-connected layer finally connects to the output. A soft-max activation is applied on top of it, to bring output vectors closer to $1 - of - K$ codewords. The output size is equal to the number of classes considered, which is increased progressively during learning. Figure 3.2 shows the architecture of the Convolutional Neural Network used.

Sigmoïdal functions limit activation to one, and are practical in the top layers to match with one-zero objective vectors. ReLU, by allowing single-unit activations up to infinity, favorize the activity to be borne by a small sub-set of the units of a layer, for each input.

This promotes sparsity of activation patterns. Both max-pooling and dropout also promote sparse activations at different levels in the network. Although the output of ReLU and max-pooling operators are deterministic while dropout is stochastic, they all help concentrate the energy on a sub-set of the units at various levels, for all images. As a result, each image is processed by a sub-part of the network's parameters, and in the training phase only this set of active parameters gets updated in reaction to said image. In the end the trained network is an assembly of processing channels sharing weight values [82].

These operations are part of a set of methods called regularizations, and have been retained by the community of Deep Learning through trial and error. They all favor the performances of deep convolutional classifiers, as they are prone to combat over-fitting, the propensity of a learning system to become overly specialized on its training data to the point of failing to generalize to previously unseen samples. To the extreme, an over-fitted system would have learned a definition of a class as the very set of representative examples it would have been trained with, and would fail to recognize even a very slightly distorted version of one of these. To the contrary, one would want a classifier to develop an abstracted canvas, and be able to extrapolate a category under different kinds of variations, such as distorsion, scale, rotation, translation, and lighting, but also the differences that are found between two indivuals of a species or two models of cars. In statistics, cross-validation is a commonly used method aimed at checking how a model's parameters inferred using a part of the available data translate to other parts [83]. In Deep Learning, regularization methods based on sparsity emulate assemblies of classifiers inside of a single network, exploiting the same logic as cross-validation. Instead of having a global learner that sees the whole set of samples and learns a holistic, reductive view of it, every step of learning is performed by a unique channel, which in the end has only been confronted with a reduced portion of the data. The different channels are averaged smoothly through their weight-sharing. This approach ensures the extracted features hold valuable information as to the common traits shared by a category, as well as those that more specifically distinguish it from other classes.

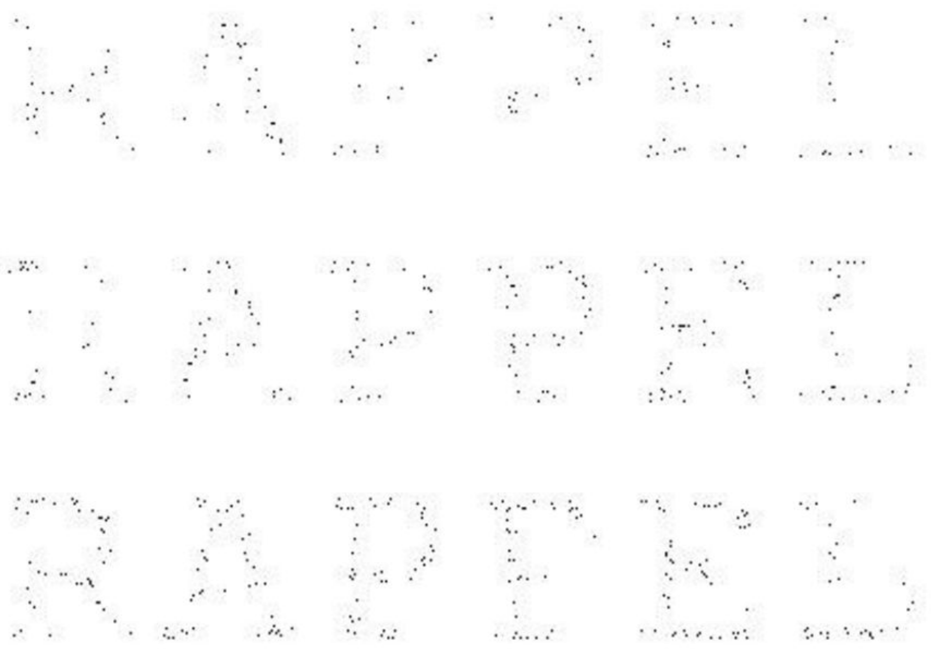

Figure 3.3: Displays of the word 'RAPPEL' at 1%, 2% and 3% of the pixels.

The network was programmed using Python along with libraries Theano [84] and Keras [85]. Training involves newly generated images obtained through random erasures of pixels from the corresponding full displays of the words. For every given iteration, each training sample is assigned an erasure probability p chosen randomly between 1% and 99%, all values being equally probable. Once this target rate is fixed, each pixel of the fully displayed word image gets erased following a Bernoulli distribution of probability p. Figure 3.3 shows an example of a word plotted at low display rates.

3.2.3 Training procedure

To ensure convergence towards a satisfyingly low classification error rate, the network was trained incrementally, starting with 100 words. Every time the error rate on a random test set got under 15%, 100 new words were added. This is similar to human learning, in that the vocabulary is expanded step-by-step. The procedure was repeated until the number of words reached 3000. The output layer was made large enough from the start to handle these 3000 classes.

To explain the necessity for training the network incrementally, let us consider a classification problem with only one class. The trained neural network would then only be able to output the single right answer, thus all possible configurations of the synaptic weights would be considered solutions. Add in a second class, and then only a region in the space of weights configurations would contain satisfying solutions to the classification problem. Add a third class, and the solutions are now a smaller region of the parameters space, contained in the previous region of solutions to the two-classes problem. Add in hundreds of other classes to the problem, and the right configurations of the weights are now localized in a considerably smaller region of the space. This explains why it can be practically difficult to bring a network to converge to a solution to a problem with a very large number of categories, starting out from a random initialization of the weights. Incremental augmentation of the number of classes allows to guide the network progressively towards a solution by refining the search area, knowing that a weights configuration solving the 3000-classes problem would necessarily also be a solution to the partial problems with 2000, 1000, and 100 classes.

The frequencies of words in the lexicon were used to create the training sequence. The number of occurrences of each word in the sequence was, thus, calculated as the floor integer value of the ratio of its lexical frequency over that of the least frequent word present. This way, even the word with the lowest frequency would be presented at least once, while more frequent ones would be seen several times. The resulting sequence would be repeatedly fed to the network, each time in a newly shuffled order, until the required performance would be attained. The average display rate of each word occurrence would be chosen as a random integer percentage ranging between 1% and 99%. At each augmentation of the list of words, the training error rate would temporarily raise back to a high level, then slowly decay again. We found that, over the long haul, the decay in error rate after each update tends to be slower and require more iterations, when the network has already been trained on numerous words.

3.2.4 Testings to study the effect of the different letter feature types on recognition

In the former psycholinguistic study on the same type of written data, the impact of certain component features of letters on recognition was analyzed. The six feature types considered were straight lines, curves, terminators of essential lines, terminators of serif segments, crossings of essential lines, and crossings between essential lines and serifs. As every single random display presented to subjects in the experiments had been recorded, the capacity of the individual display rates of the different feature types to explain word recognition was studied using logistic regression. Here we use another approach as our trained network can be queried with new data without restraint. Hence, we observe the evolution of the recognition performance starting with a blank screen, and adding pixels randomly across the word pixel map. In addition to displaying pixels randomly in the word, we also experiment displaying random pixels in locations associated with one type of features, specifically. We repeat this process separately for all six types of features. Given that the different types of features typically represent highly varying proportions of the pixels of a word, we compare the average recognition for the different feature types at equal number of shown pixels, equal feature type display rate, and equal word display rate. To mitigate the effect of word frequency in this study, tests are performed on a sample of 200 words, of which 20 have very low frequency (<10), 20 are very frequent (>50) and the remainder 160 have intermediate frequency.

3.2.5 Study of the learned feature detectors and associated activation patterns

Measuring the mutual information between intermediate layer neurons and classes

Referring to the concepts from Shannon's theory of information [86], the entropy of a discrete random variable X is the expected surprisal associated with it, given by:

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{3.1}$$

where x represents a possible value that X can hold, and $p(x)$ is the probability of the event $(X = x)$. Entropy is expressed in bits of information provided the logarithm in (3.1) is in base 2. It is maximal if all possible values x of X are equally probable.

The conditional entropy of a given random variable X knowing the value of another variable Y, is defined by:

$$H(X \mid Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y) \tag{3.2}$$

$H(X|Y)$ quantifies the amount of uncertainty remaining as to the outcome of X once knowing that of Y.

The mutual information between two variables X and Y is, to the contrary, what uncertainty is removed from the entropy of one variable when the other is known:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x \in X} \sum_{y \in Y} p(xy) \log \frac{p(xy)}{p(x)p(y)}. \tag{3.3}$$

Basically, it represents the quantity of information that X holds about Y and that Y holds about X [87]. It is worth zero if X and Y are independent. Its maximum value is $min(H(X), H(Y))$ and is reached when one of the variables is totally redundant given the other.

The mutual information between the activation of feature detectors in a given layer of a neural network, and the class of the input, indicates how categorical this layer is. Here we assess the mutual information between the activation patterns of the hidden layers of the network and its output layer representing the class, using empirical measures of p based on observed frequency. Estimators for these magnitudes are calculated using the formula:

$$\hat{I}(X;Y) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{\hat{p}(x^i y^i)}{\hat{p}(x^i)\hat{p}(y^i)}. \tag{3.4}$$

where X is the variable associated with the binarized value of a single neuron randomly sampled from the layer of interest, and the Y variable represents the class, whose value is integer and ranges from 0 to 2999. The values x are obtained by binarizing the activity of neurons using a 0,5 threshold. The value of Î is calculated by producing ten random displays of each of the 3000 words, and recording the activations of each layer in reaction to these images. Denoting n the number of neurons in a layer, this results in $30000 * n$ measurements for the pair $(X;Y)$, as X represents the value of any given neuron in the layer. That is, 100.800.000 measures for the first convolutional layer, 165.000.000 for the second one, 259.200.000 for the third one, and 2.100.000 for the last, fully-connected hidden layer with 70 neurons. Given these numbers, the calculation of (3.4) can be tedious, hence for all convolutional layers it is made on a random subsample of $N = 2.000.000$ pairs of neuron activation and class value. For the fully-connected hidden layer it is made on the whole set of 2.100.000 measurement pairs, without sub-sampling.

Visualization of the internal representation geometry using t-SNE embedding

To study the geometry of representation spaces at the different levels of the deep network, visualization tools can be useful to get an idea about how the density of samples is spread across these high-dimensional systems. Recently developed embedding methods are very interesting tools in order to display complex, high-dimensional data-sets across two or three dimensions. Among these, t-SNE [89] is particularly efficient at producing maps where the groups of points associated with the different classes are easy to identify, without recourse to supervision. It relies on the approximate reproduction, in the low-dimensional map space, of the distance relationships between data points in the original, high-dimensional referential. To this end, original distances are first converted into probabilities of the points to be neighbors of one another, or similarities.

The similarity of point j to point i in the space of the data-set is expressed as:

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}, \tag{3.5}$$

Similarities between map points are formulated the same way. Here the variance is set to $\frac{1}{\sqrt{2}}$, giving:

$$q_{j|i} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i} exp(-\|y_i - y_k\|^2)}, \tag{3.6}$$

In both referentials, only similarities between different points are considered , and $p_{i|i} = q_{i|i} = 0$ for all i. Relying on these two definitions, a cost function is calculated as the sum of Kullback-Leibler divergences between distributions $p_{j|i}$ and $q_{j|i}$ over all points i, j of the data:

$$C = \sum_i KL(P_i\|Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \tag{3.7}$$

where P_i and Q_i are conditional probabilities defined over all other points given x_i and y_i, respectively.

Map points coordinate vectors y are updated iteratively in a gradient descent procedure aimed at minimizing this cost function, and thus at making neighborhood probability distributions Q_i in the map space as close as possible to distributions P_i from the data

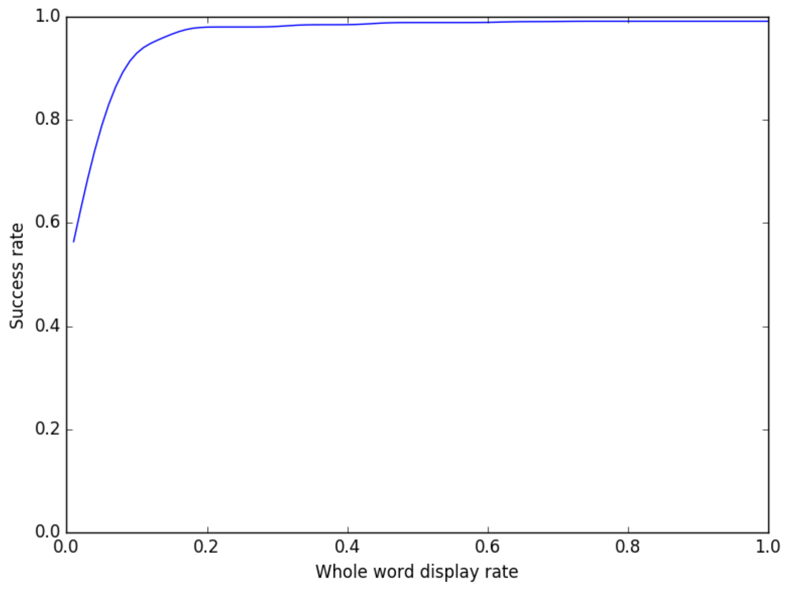

Figure 3.4: Evolution of the average word recognition rate as a function of the display rate.

space.

In P_i, values of the standard deviations σ_i are adapted to the local densities around data points x_i, where values of σ should be smaller in denser regions of the input space and larger in sparser regions. A given value of σ_i defines a specific neighborhood probability distribution over all other data points given x_i.

3.3 Results

3.3.1 Display rate effects

Similarly to human subjects, the trained network recognizes the majority of words correctly for display rates superior to 5%. Figure 3.4 shows the evolution of the average success rate as a function of the display rate.

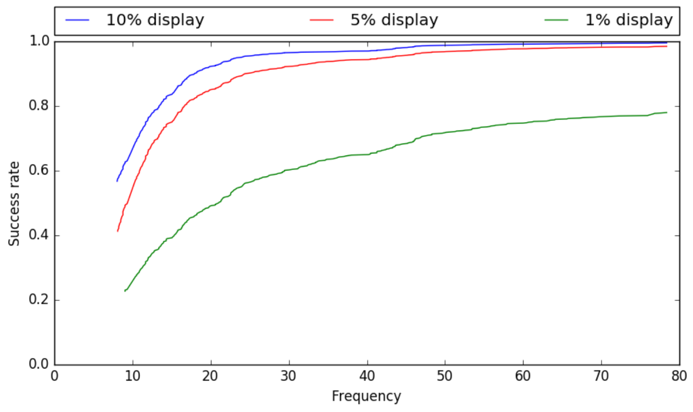

Figure 3.5: Average word recognition rate as a function of lexical frequency for three display rates.

3.3.2 Relationship with lexical factors

frequency

Word frequency has a very notable impact on recognition. Frequency values correspond to the average number of occurrences of each word per 1.000.000 words measured in a corpus of 50.000.000 words [88]. We study the relationship between word frequency and recognition using three display rates, namely 1%, 5% and 10%. Each word is first presented 10 times for each display rate to calculate an average success rate. Figure 3.5 shows the three regression curves obtained by averaging on $(frequency,\ success\ rate)$ pair samples.

Success rate seems approximately proportional to the exponential of the word frequency. Very frequent words (f>50) are very well recognized even at 5% of pixels. For 1% displays, rare words (f<10) are recognized less than 20% of the time on average, when very frequent words are recognized about 70% of the time.

length

Word lengths in experiments range from 3 to 9. We find the error rate to be higher on average for longer words. Also, the average length difference between the accurate word and the response of the network is higher for longer words.

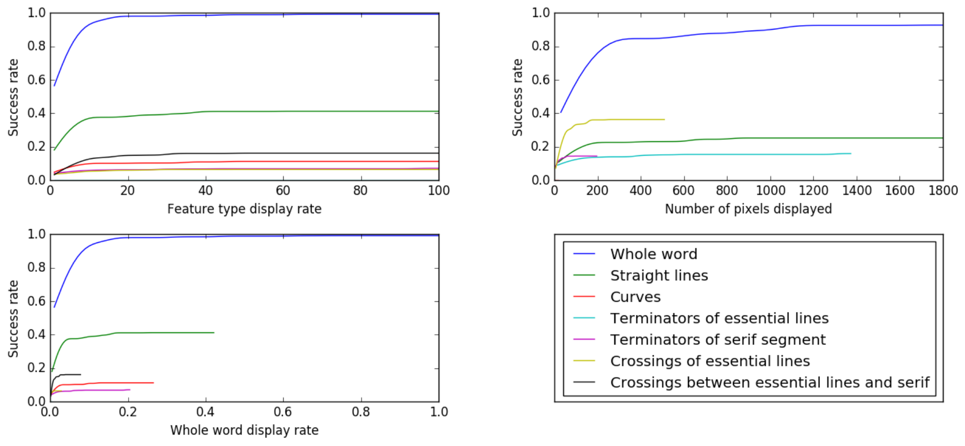

Figure 3.6: Average word recognition rates for feature-type specific displays, as a function of the specific display rate (top-left), overall display rate (bottom-left), and number of pixels displayed (top-right).

Table 3.1:
AVERAGE RESPECTIVE PROPORTIONS OF THE SIX FEATURE TYPES IN THE PIXELS OF WORDS.

Feature type	**Proportion**
Straight lines	42.12%
Curves	26.54%
Terminators of essential lines	0.25%
Terminators of serif segments	20.47%
Crossings of essential lines	2.85%
Crossings between essential lines and serifs	7.78%

3.3.3 Effects of feature types

Figure 3.6 shows the network's success rate when pixels are displayed randomly in the whole word or in specific feature types, as a function of the number of shown pixels, the feature type display rate, and the word display rate, respectively. This representation is motivated by the highly-varying proportions of pixels in each feature type. Table (3.1) gives the average proportions of pixels in the six feature types in the data-set.

On Figure 3.6, curves associated with the six feature types are limited to the right by their respective proportions in words, and thus their maximum number of pixels. All curves are characterized by a quick increase in success rate with the first steps of display augmentation, soon followed by a slow progression, or stagnation in the case of the overall word displays. Interestingly, this is particularly evident in Figure 3.6 where the progression rate of good recognition is the most similar between the graphs of the different feature types, whereas the number of pixels associated with a given abscissa varies widely from one feature type to another. Indeed, all curves enter the slower progression phase around the 5% mark of feature-specific display rate. Displaying a specific feature type at 10% is usually enough to get the information it holds about the word, as further increasing this display rate does not increase performance significantly. This remains true whether the feature type accounts for 40% of the pixels of the word, or 10%.

As is the case for human subjects, lines are hereagain the feature type that is the most important for good recognition [52]. Crossings between essential lines and serif come in second, followed closely by curves. The previous study of human recognition on this task evidenced that curves were the second most determinant feature type in triggering recognition, with crossings between essential lines and serif coming in third. In both studies, though, curves and crossings between essential lines and serif and curves appear to have very comparable effects.

3.3.4 Observations on the geometry of representations and the information carried by their activation patterns

Mutual information

The calculation of the mutual information between the activity of single randomly sampled neurons in the hidden layers and the index of the class, using formula (3.4), gives the values in table (3.2).

This shows how the activation of a single neuron in the hidden layer of the network brings more and more information about the class of the image as we ascend the layer

Table 3.2:
AVERAGE MUTUAL INFORMATION BETWEEN NEURONS IN THE HIDDEN LAYERS AND THE CLASS INDEX.

Layer	**Mutual information per neuron with the class layer**
Convolution 1	0.02396 bits
Convolution 2	0.11995 bits
Convolution 3	0.27970 bits
Fully-connected layer	0.34551 bits

hierarchy. The activity of neurons in a given layer is more class-specific than that in the layer under it.

Embedding

Figure 3.7 shows a two-dimensional embedding obtained using, as input data, last feature layer activation patterns for words of length seven, eight and nine. These activations result from stochastic partial displays of the pixels of words, where several different displays are generated for each word. Words are grouped into five clusters based on their orthographic proximity, so that words in the same cluster share at least four common letters. The letters in common are often a suffix, like -VRIER or -ATION, but can also be present in varying orders among words of a group, as the letters E,E,I,R,S in GRISETTES and BRASSERIE. The group of words containing letters V,R,I,E,R is concentrated in the down-left corner, whereas groups M,E,N,T and E,R,I,E,S are more spread across the center of the graph. Groups A,T,I,O,N and especially I,L,L,O,N are split in several parts. Some regions contain close-by words belonging to different groups, like COTERIES, RILLONS and OUVRIERS at the right of the graph, that all share letter O and pair RI but in different parts of the word, and the final S. Also, AGACERIES and SERREMENT, in the left part of the figure, share the partially overlapping sequence ER*E, with an I or R in between that have a vertical line in common. This leads to think that the learned feature detectors at the network's penultimate level of representation, respond to associations between specific

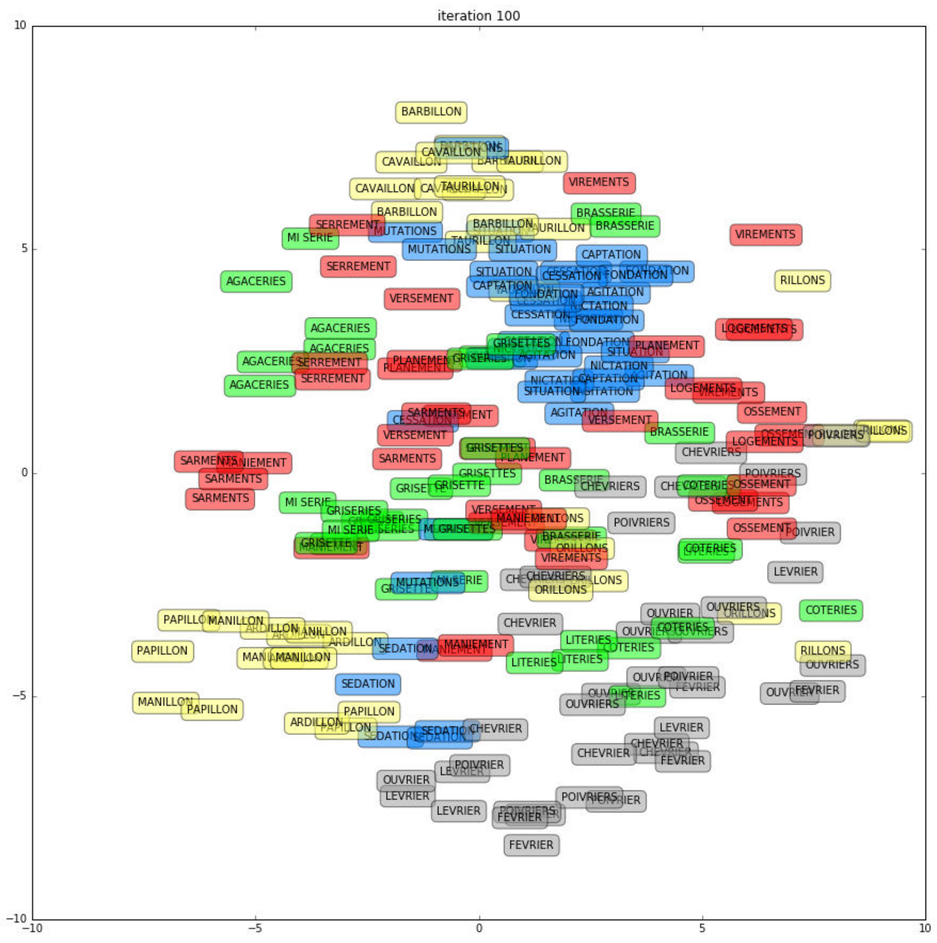

Figure 3.7: t-SNE visualization of the last hidden layer activation patterns resulting from input words grouped in 5 clusters. Each cluster contains words with at least 5 letters in common, e.g., PAPILLON and MANILLON.

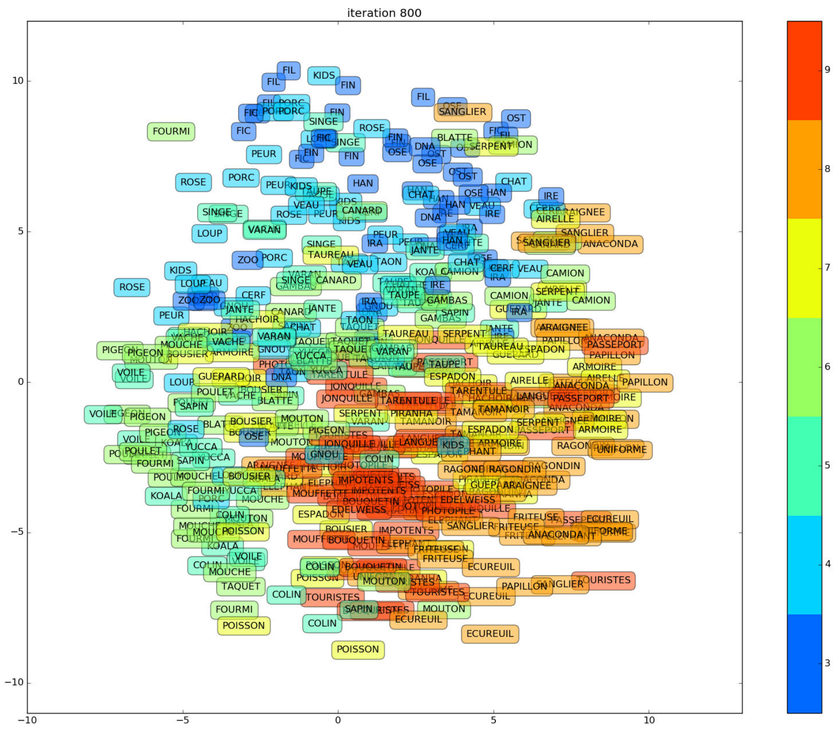

Figure 3.8: 2-dimensional visualization of the network's last hidden layer activation patterns associated with words of variable lengths, obtained after learning a t-SNE embedding for 100 iterations. Seven groups of words of lengths three to nine are displayed in different colors.

letters and approximate positions in the word.

Figure 3.8 shows an embedding where words are colored by length from three to nine. Words of length three and four are clearly concentrated in the upper part of the graph, and long words of length eight and nine are more present in the lower part. This shows that length information is maintained in the penultimate level of the network and plays a role in the classification.

3.4 Discussion

3.4.1 Reproducibility of lexical effects relative to word frequency

The method used to reproduce the effect of lexical frequency during learning is equivalent to presenting a learner with a text where word order is random, but where word occurences follow a distribution close to that found in real text. As such the learning process holds reasonable cognitive plausibility. As is seen in human subjects, the display rate necessary to trigger word recognition is highly dependent on word frequency. Very frequent words can be accurately classified at lower display rates than rare words. This brings evidence that the frequency effects found in psycholinguistic experiments can be reproduced without taking other aspects into account, like semantic relationships between words. Indeed, the size of the semantic neighborhood of words is known to be correlated with their frequency, and could therefore be suspected to play a role in the ease of reading frequent words. However, presented experiments reveal that such semantic information is not necessary to reproduce this effect.

3.4.2 Role of lexical factors

Experiments reveal that the relative impacts of the different feature types on the performances of the deep convolutional network and of human subjects are comparable. One may argue that feature types that have a strong impact, namely lines and curves, do so because they account for more surface in the letters. Pixels randomly displayed in these letter parts therefore tend to be spread apart more, and give more indication to the viewer, human or artificial. However, crossings between essential lines and serifs represent a much smaller part of the letters, and still have an influence comparable to that of curves. This can be due to the simple fact that these parts are what make letters most distinguishable from one another, *i.e.*, it can be a factor inherent to the data-set. Still, this shows that deep convolutional nets are sensitive to this information, like humans are.

To the contrary of human subjects, the deep net shows a preference for shorter words. We note that, even though the first convolutional layers of the network detect small and mid-sized shapes independently of their location in the image, the fully-connected layer captures the global organization of the image. Since words pixel vectors are arranged so as to start with the bounding box of the first letter, the network is trained more for the first letter positions, and only sees the rightmost part of its input vectors activate for longer words. It is thus less trained on the right part of the 9 letter spots, which explains its lesser performances on longer words.

3.4.3 Comparison with other models

CNN is a model of vision and does not contain, in its current form, specific processing strategies for written text. Other models of reading, listed in Section 3.1.2, explain how full letters are processed, and can provide robustness to shifted letter positions or elastic distortions. No feedback processing occurs from upper levels to lower ones, as in the FAM and IAM models, during the recognition phase. Top-down interaction only happens during the error gradient back-propagation phase. No lateral interaction occurs between detectors at the same level, unlike IAM. The principal type of distortion handled by the presented neural network is partial erasures of pixelated letters. This is because of the training data-set used, and it has been shown that convolutional networks can perform well on distorted text [90]. Moreover, our study of the internal geometry of the hidden layer representations using embeddings reveals that the position of letters in a word is less important than their simple presence. This suggests that the model could have some degree of robustness to typoglycemia and be able to recognize words with scrambled letters [91].

3.4.4 Major differences with human recognition behavior

It is crucial to note how deep networks that perform image classification at near human-level performance on a given data-set can still behave very differently on new image variations.

Adversarial examples are a notorious instance of this phenomenon [92]. Such examples can be obtained using gradient descent on the pixels space, with a cost function that promotes misclassification while penalizing large modifications of pixel values. In the end, the original and modified images are barely distinguishable to humans, yet the neural network that could classify the original ones accurately is fooled by the new examples. To the contrary, by performing gradient ascent on the pixels in order to maximize the activation of a class, one can create a synthetic image that resembles nothing like the actual instances of the class. These inputs, that can look very much like random noise, still activate strongly the targeted class neuron, meaning they are classified with very high confidence by the network [93]. These phenomena are sometimes interpreted as proof that the structure of deep learning networks and their training process are fundamentally different from the working of the brain. One may also argue that current neural networks are simply not trained on enough data. When trained on ImageNet, artificial classifiers only get to see hundreds of discrete samples of a given class, where human subjects have typically been confronted multiple times with continuous streams of images from instances of this category, under varying lighting, pose and background. Of course, humans can create a new mental category out of a few discrete examples [94]. Even in this case they would not associate noise-like inputs with this category, the way deep networks do.

So maybe, the same way currently available data-sets have permitted to reach near human-level performance, much larger data-sets consisting of multiple videos for each object category could bring fully human-like behavior on most images, to the level of the degree of confidence in one's decisions. Such data-sets could obviously be gathered nowadays, but using them will require considerable memory and computation resources. For the moment, these odd false positives and false negatives offer room for investigation and trial of new training techniques that would bring more human-like behavior. Hubel and Wiesel determined the approximate shape of the preferred input of the orientation filters at the low levels of the visual cortex. For the most part, precise estimations of the visual inputs maximizing the reaction of neurons is lacking for the deeper levels. Such data would bring a possibility to diminish the gap between deep networks and the brain at the intermediate representation levels.

3.5 Conclusions

This study has corroborated findings from the former psycholinguistic studies where human subjects were confronted with the same data-set of partially displayed pixelated words. Namely, many of the display and lexical factors correlated with good recognition in the previous study were found again to play a significant role here. The presented artificial neural network indeed performs better on average at higher display rates, and necessitates significantly less pixel information to recognize frequent words than rare words. Moreover we find lines, curves and crossings between essential lines and serif to be the most important letter parts for recognition, as was the case for human subjects. To the contrary, some previous work found terminators and line junctions to be more important. We conclude that the three prominent feature types here may result from the nature of the data-set, and particularly from the pixelation process. Secondly, the fact that the deep network reacts more strongly to the same feature types as humans supports the hypothesis that its working has at least some similarity with that of the brain. The initial quest for performing computations using layered neural networks was motivated by the same neuroscientific discoveries that inspired the main models of reading. As a result, although some of their key features vary, *e.g.*, top-down retroaction and lateral interactions, the principal Deep Learning architectures are not in opposition to these models. Integrating these top-down and same-level interactions in the current model would be an interesting area of investigation. Deep networks do not recognize good images altered by a slight shift in pixel values, but they recognize synthetic images with the highest possible certainty, that look nothing like any real-world category instances. It is not clear yet whether these differences are due to the training procedure and amount of training data, or to the structure of these neural networks. Near-human performance is now achieved by Deep Learning networks for many data-sets. The Universal Approximation Theorem states that with a high enough number of neurons, any mapping can be achieved between pairs of vectors. Consequently, one could expect to reproduce the same associations between visual objects and categories as in a human mind using current algorithms, provided the right data, memory and computing power were available. This does not mean the visual input would then be processed and stored as efficiently as it is by the brain. Also, it is not certain that such an artificial learner would generalize sufficiently well to new data. We can legitimately assume that the working of current deep networks is different in nature to that of the visual cortex, although some aspects of it are more questionable than others. Gradient descent may not be the definitive tool for training, no more than convolutional filters and the commonly used regularization techniques. Hierarchy, though, may be crucial. It is indeed a key characteristic of mammal sensory systems. Its computational and material efficacy has been proven, and is likely one of the reasons why evolution shaped our brains this way [77].

Chapter 4

Assembly Output Codes for Learning Neural Networks

Neural network-based classifiers usually encode the class labels of input data via a completely disjoint code, *i.e.* a binary vector with only one bit associated with each category. We use coding theory to propose assembly codes where each element is associated with several classes, making for better target vectors. These codes emulate the combination of several classifiers, which is a well-known method to improve decision accuracy. Our experiments on data-sets such as MNIST with a multi-layer neural network show that assembly output codes, which are characterized by a higher minimum Hamming distance, result in better classification performance. These codes are also well suited to the use of clustered clique-based networks in category representation.

4.1 Introduction

Automatic learning systems are different from storing systems in that they aim at generalizing to unknown inputs. This happens through the extraction of the core features of learned data, and works as long as the unknown data to extrapolate to follows a similar distribution. The system thus learns a dictionary of features that is targeted to be well suited for the task at hand, *e.g.* classification. These features are meant to correspond to the most relevant building blocks underlying the training inputs, that unseen data samples would likely also be made of. In supervised Deep Learning networks, an output is calculated through a pipeline of vector-matrix products between input data and connection weights, intertwined with non-linear mathematical operators. Each level of the network is associated with a set of features that is more and more abstract as one moves towards the upper layers. During learning, an error is calculated from the difference between the resulting output and an objective vector specific to the class of the input. A gradient is then calculated from this error for the whole set of connection weights, and the hierarchy of features thus gets optimized through gradient descent for the task of classifying the data-set examples. Combining classifiers has been an extensive area of research for a few decades [95] and several algorithms have been shown to bring improved decision by leveraging the diversity brought by an assembly of systems [96–98]. Among these methods, boosting is a way to combine opinions of experts by weighting them based on their respective estimated accuracy. These base classifiers can be differentiated using various strategies, like training them on various subsets of the data or by providing them with different sub-parts of an ensemble of learned feature detectors. Another way to combine classifiers is to split the problem into a set of binary problems. A base classifier will then focus on classifying inputs between two of the initial classes, as in One-Vs-One (OVO), or between one class and the rest as in One-Vs-All (OVA). A way to implement these strategies is to provide a classifier with objective vectors that are not always specific of a single class but can be associated with a set of classes. Much attention is paid here to the Error-Correcting Output Coding (ECOC) method, which splits a multi-class problem into several two-way classification problems between meta-classes. It is shown in [99] that ECOC can reach a better classification performance on an image data-set as compared to other multi-class methods. The approaches presented here are derived from ECOC for multi-class problems. These methods allocate assemblies of output neurons to the different input classes, with potential overlap between the codes of two classes. A clustering of the output layer is also applied, with only one active neuron per cluster for each target vector, and a local soft-max [100] process applied in each cluster at test time. The soft-max operator has the effect of normalizing to 1 the sum of energies of output neurons in each cluster. Experimental results suggest that the number of classes sharing an output node impacts performance, and so does the minimal distance between class codes. This finding is maintained when output codes are repeated so as to ensure that the different tested networks have very similar numbers of parameters. The outline of this chapter is as follows. Section 4.2 provides theoretical considerations on the advantages of different output codes. Section 4.3 explains the methodology used in training the networks. Section 4.4 presents experimental results.

4.2 Coding theory

Prior to experimenting with assembly codes as output for neural networks, a theoretical analysis can provide insights about which assemblies should perform better. Consider a classifier with P classes to identify. The simplest way to make the classifier express its decision is to assign a single output node (the so-called grandmother cell) to each class. We propose to replace these P nodes with $n = \binom{P}{m}$ nodes representing all the combinations (or assemblies) of m classes among P. Let us define the coding rate R of the corresponding code as the ratio between $\log_2(P)$ and n:

$$R = \frac{\log_2(P)}{\binom{P}{m}}. \tag{4.1}$$

To calculate the minimum Hamming distance, let us consider any two classes among P. The number of assemblies that contain neither one nor the other is $\binom{P-2}{m}$ and the number of assemblies that contain both is $\binom{P-2}{m-2}$.

The minimum Hamming distance of the code is given by n minus the latter two terms:

$$d_{min} = \binom{P}{m} - \binom{P-2}{m} - \binom{P-2}{m-2}. \tag{4.2}$$

The product of R and d_{min}, called the merit factor F, is deduced as:

$$F = Rd_{min} = \frac{2m(P-m)\log_2(P)}{(P-1)P}. \tag{4.3}$$

and its maximal value is obtained for $m = \frac{P}{2}$:

$$F_{max} = \frac{P\log_2(P)}{2(P-1)}. \tag{4.4}$$

The corresponding minimum distance may be expressed as:

$$(d_{min})_{max} = \frac{P}{(P-1)}\frac{n}{2}. \tag{4.5}$$

For instance, with $P = 10$ (*e.g.* for MNIST classification), the best code involves quintuplet assemblies and offers a minimum distance of 140 and a merit factor of 1.84, with $n = 252$. If quintuplets are replaced with couples, the parameters become: $d_{min} = 16$ and $n = 45$. Note that the classical output code (one node per class), still with $P = 10$, has a minimum distance of 2 with $n = 10$.

Now we propose that the n assemblies be distributed among c clusters such that each class appears once and only once in each cluster. Therefore, there are $l = \frac{P}{m}$ nodes in each cluster, assuming that P is a multiple of m (Fig. 1). This structure has two advantages. Firstly, it complies exactly with the clustered clique-based associative memory proposed in [14] which offers the possibility to store a number of patterns proportional to l^2 (for larger sought diversities, the sparse scheme proposed in [42] may be contemplated). However, the optimal value I = 2 (as deemed by the optimization of F) is too low for a clique-based implementation and a trade-off has then to be found. For instance, $l = 5$ (that is, $m = 2$) seems a good choice for $P = 10$. The second advantage is that a cluster containing

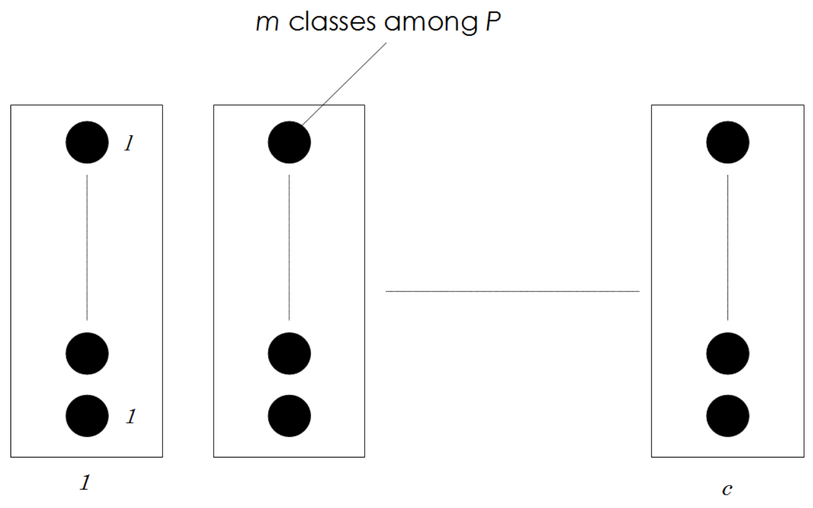

Figure 4.1: The output layer of the classifier is organized in c clusters, each one having l nodes. These l nodes represent disjoint combinations of m classes among P with $l = \frac{P}{m}$.

disjoint assemblies and therefore probabilities, through the soft-max principle [101], may be rigorously used for both learning and testing.

4.3 Methodology

4.3.1 MNIST

MNIST [102] is a data-set of grey-scale images of handwritten digits that is widely used in Machine Learning as a benchmark. The data-set is made of 60.000 images targeted for training and 10.000 test examples. Many published works use the first 50.000 examples from the training set to actually train the network, saving the last 10.000 examples to perform cross-validation. Some papers however present networks trained on the whole set of 60.000 training images, as is the case in [103]. Here the 50.000/10.000 split of the training set is used for the experiments of section 4.4.1, whereas in sections 4.4.2 and 4.4.3 the networks are trained with all 60.000 examples. Figure 4.2 shows a sample of images from MNIST.

Figure 4.2: Sample of images from the MNIST data-set.

4.3.2 SVHN

SVHN [104] is a data-set made of color images of digits captured in real-world situations, *e.g.* house numberings. As for MNIST, the number of classes is 10. It contains 73.257 training images and 26.032 test images. We use 60.000 examples from the train set to actually train our networks, and the remaining 13.257 serve for cross-validation. Figure 4.3 shows a sample of images from SVHN.

4.3.3 Assembly codes

Couple cells are a method we introduce for encoding the class using a distributed code. Each cell no longer reacts to a single class but is specific of a couple of classes. Hence since the experiments are performed on data-sets with 10 classes, there are 45 possible couplings a given output neuron can be associated with. The whole set of 45 couplings

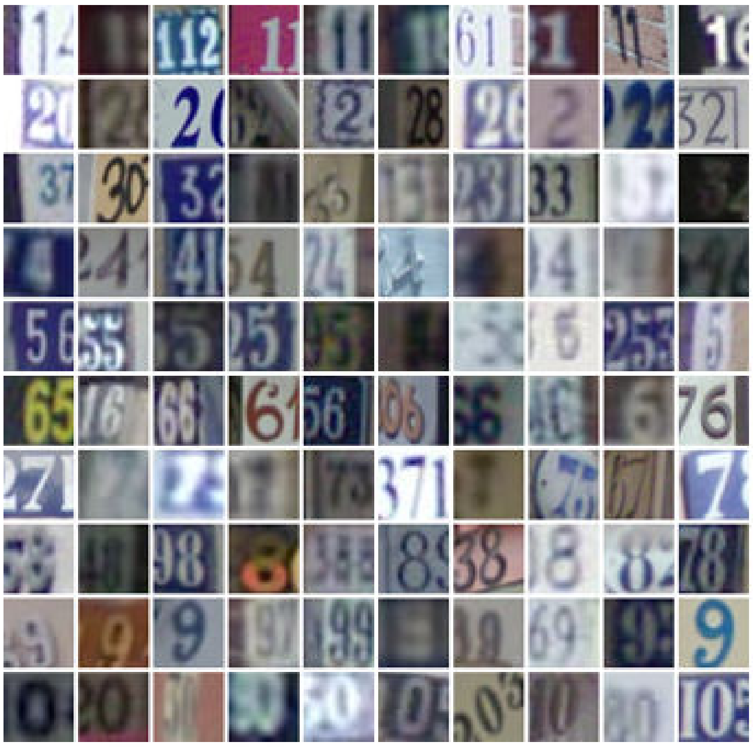

Figure 4.3: Sample of images from the SVHN data-set.

is used here. Moreover, it is possible to partition these 45 couplings into 9 clusters of 5 couple cells where each individual class is represented exactly once in each cluster. This way the soft-max methodology can be applied inside of every cluster, between 5 competing hypotheses that are mutually exclusive. Quintuplets are yet another code where each neuron is associated with a combination of classes, this time 5 among 10. The same approach as for the couples is used, by using all 252 possible quintuplets and partitioning them into 126 clusters each containing 2 complementary quintuplet cells. The first two important factors considered to choose assembly codes are usability in a clique-based architecture and the merit factor. Alongside with the grandmother cell (our baseline), couples of classes (best trade-off cliques/merit factor) and quintuplets of classes (best merit factor, but larger output network) are tested. But only comparing these parameters is not enough, because one could argue that couples and quintuplets work better because they have a larger number of parameters for the last layer of the neural network. To avoid this we also train networks

Table 4.1:
Summary of the tested assembly codes. Assembly code * number: Code repeated number times. [1]: 252 non-repeated clusters

Assembly code	n	m	l	c	F	d_{min}
Grandmother cell (1GM)	10	1	10	1	0.66	2
Couples (1C)	45	2	5	9	1.18	16
Quintuplets (1Q)	252	5	2	126	1.84	140
Grandmother cell * 25 (25GM)	250	1	10	25	0.66	50
Couples * 6 (6C)	270	2	5	54	1.18	96
Grandmother cell * 126 (126GM)	1260	1	10	126	0.66	252
Couples * 28 (28C)	1260	2	5	252	1.18	448
Quintuplets * 5 (5Q)	1260	5	2	630	1.84	700
Special Couples (SC)[1]	1260	2	5	252	1.18	448

using repeated codes to reach output length of equal or comparable size. This repetition allows us to make a fair comparison between networks with virtually the same number of parameters in spite of using different output codes. Couples of classes have an interesting characteristic, in that there are a very large number of different possible ways to partition 45 couple cells into 9 clusters, each featuring the 10 classes. Therefore it is possible to design an output layer of 1.260 couple cells parted in 252 clusters where no cluster configuration is repeated twice. During the training phase the categorical cross-entropy is used as the loss function. For classification a majority voting is done where each active output node votes for its associated set of classes. The assembly codes are summarized in table 4.1.

4.3.4 Neural network settings

For tests, the neural networks used are multi-layer perceptrons. Two architectures of network are used, one that is shallow and the other deep. The shallow network has only one hidden layer, while the deep network has five hidden layers. The results are presented by the mean and standard deviation over ten executions with different weight initialization, noise and image order. In sections 4.4.1 and 4.4.2 the neural network is shallow and its only hidden layer is composed of 2000 feature units. As it is applied on MNIST, its number of parameters is 1.568.000+2000*n. Training lasts for 200 epochs with a constant learning rate of 0.1. The model "baseline + noise" from [103] is chosen as a base for the deep network used in sections 4.4.3 and 4.4.4. This is a network that gets close to the current state-of-the-art in the task of permutation-invariant MNIST. It has 6 layers, where the first 5 of them are fully connected layers (with sizes 1000-500-250-250-250) and the last one is an output layer. At each connection layer, between the input and the activation, a batch normalization [105] and a Gaussian noise with mean 0 and standard deviation 0.3 are used. To finish the connection layer, a rectifier activation is used. The output layer has the length of the output code. A batch normalization is applied between the input and the activation and as in the case of the shallow network it uses a per-cluster soft-max activation. The number of parameters of each network is given by the formula: 1.000*(input length) + 750.000 + 250*n. This means: 1.534.000+250*n for MNIST and 3.822.000 + 250*n for SVHN. This deep network has 150 epochs to learn where it optimizes the weights with the ADAM optimizer [106], using an initial learning rate of 0.002 that has an annealing phase of 50 epochs where the learning rate decays linearly to 0.

4.4 Results

4.4.1 Experimenting with random codes

Assigning the same number of output nodes to the different classes may not be ideal for a real-world data-set. Following this idea, it may be interesting to allocate different amounts of the output material to the different combinations of classes, as for instance the distributions of examples can be more correlated in a subset of classes than on average in the whole data-set. An experimental scheme inspired from the quintuplet configuration is tested, with 252 output neurons parted in 126 clusters of size 2 where a soft-max is applied. Instead of rigorously associating an output cell with each possible quintuplet however, ten binary

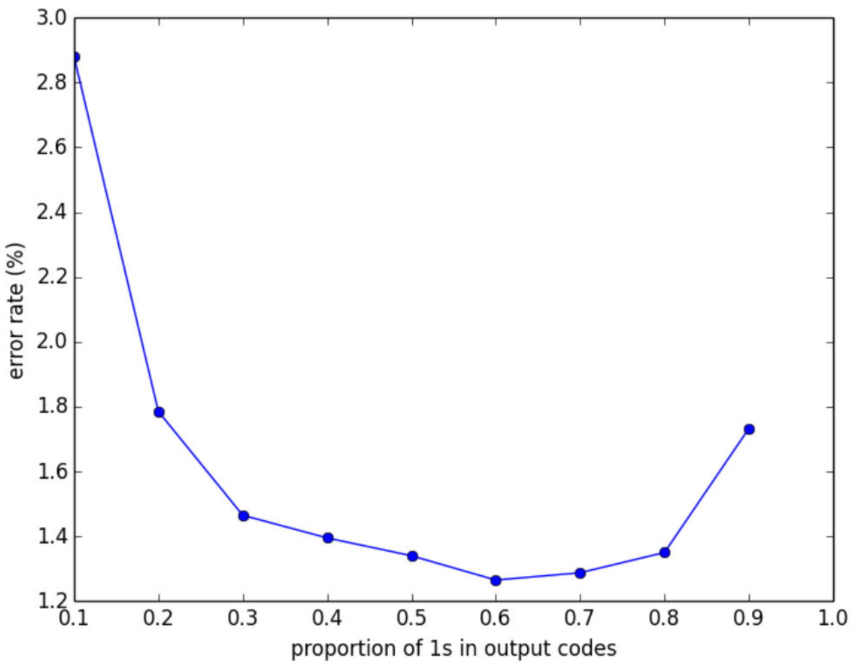

Figure 4.4: Influence of the proportion of ones in the output target vectors on classification error rates.

output codes are generated following a binomial distribution. This distribution is modulated to make vary the average number of ones in the output codes. Used as output for a shallow network, the average number of ones has an impact on classification performance, as shown by Figure 4.4.

We see here that the error rate is maximal when there are less than 10% of ones in the output codes. It decays with higher values down to a minimum reached when there are around 60% of ones. Above that proportion, the error rate raises again. The minimal distance is obtained when generating as many ones as zeros on output as shown on Figure 4.5, whereas the classifier performs better with output codes made of 60% up to 80% of ones. The error rate is also about 1.7 times lower for 90% of ones as compared to the case with 10% of ones, while the minimal distance is the same in both cases. This asymmetry is due to the way the class label is selected at test time, where a majority voting procedure is applied in which each output unit getting a value of 1 increments the score of all classes it

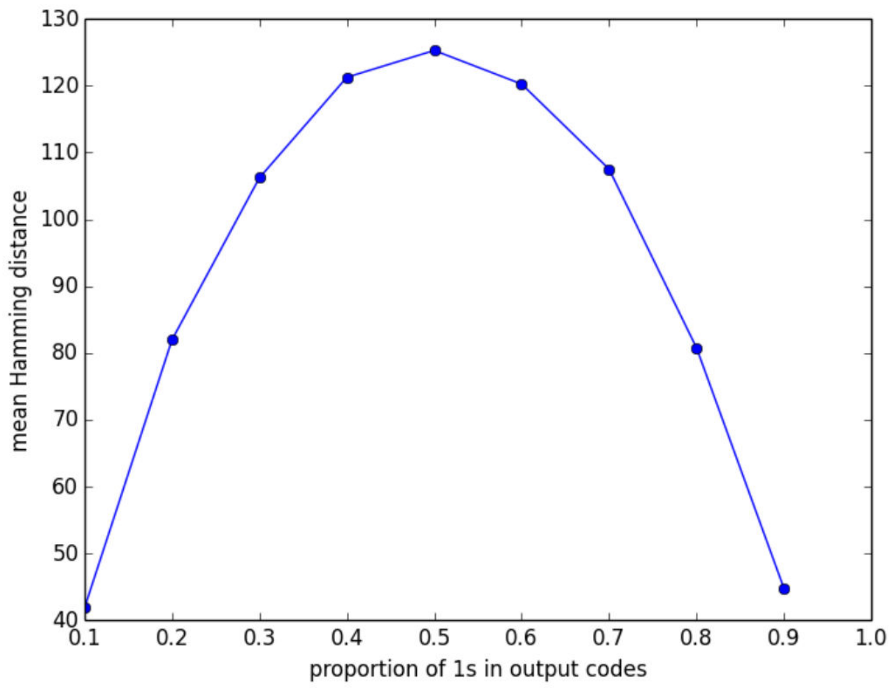

Figure 4.5: Measured mean Hamming distance between generated class codes depending on the proportion of ones.

is associated with. In this setting, the higher the number of ones in the output targets, the more connection weights end up being involved in the decision process at test time. The classifier thus makes use of a finer features-output mapping in this case. With too many ones however, *e.g.* 90%, the distance between codes becomes too low which affects performance.

4.4.2 MNIST - Shallow network

The first test with the assembly codes defined in section 4.3.3, applies the shallow network to the MNIST data-set. The goal of this test is to compare the assemblies in similar settings, rather than achieving a competitive result to the state-of-the-art. The results indicate that Quintuplets are better than Grandmother cells and also that Couples are better than Grandmother cells, but it is inconclusive in the comparison between Couples and Quintuplets (less than two misclassified images of difference on average between the best networks of the two assemblies).

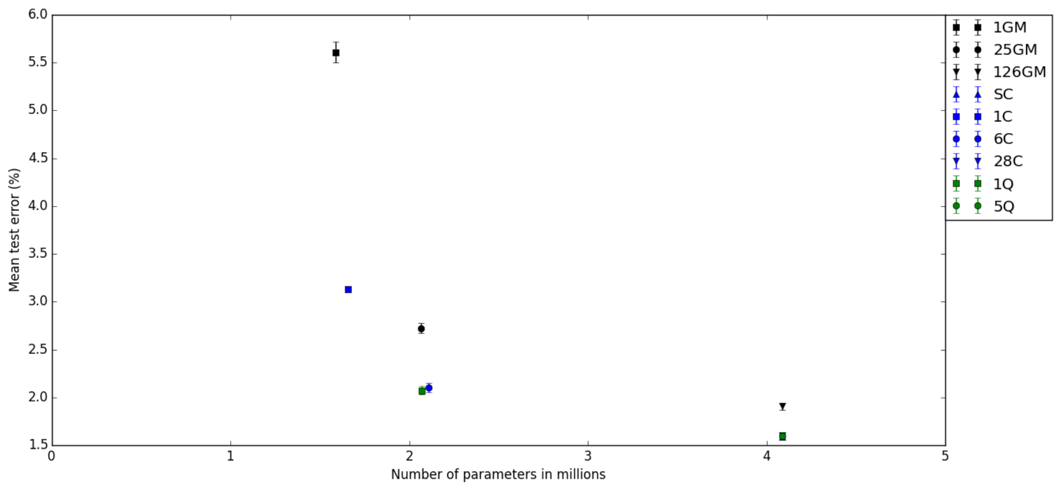

Figure 4.6: Results summary for a shallow network on the MNIST data-set. GM, C and Q respectively stand for the Grandmother Cell, Couple and Quintuplet codes. The numbers beforehand indicate the number of repetitions of the code to adjust output lengths. 25GM, 6C and 1Q all have approximately 252 output units, while 126GM, 28C, SC and 5Q have exactly 1260 output units.

4.4.3 MNIST - Deep network

Another test is conducted on the MNIST data-set, now trying to emulate the results from [103]. It is summarized in Figure 4.7. Despite respecting the hierarchy of Quintuplets $\geq$ Couples $\geq$ Grandmother cells, the results are too close to take any conclusions (less than one image on average between 5Q and 126G).

4.4.4 SVHN - Deep Network

Finally, applying the deep network to the SVHN data-set allows us to obtain more significant results than the ones obtained over MNIST. SVHN is more difficult to classify and has been less extensively studied. The results respect the hierarchy drawn from coding theory in section 2 (Quintuplets $\geq$ Couples $\geq$ Grandmother Cells), with an average distance of 0.11% ($\sim$ 26 images) between the worst quintuplets network and the best couples network and 0.33($\sim$ 85 images) between the worst quintuplets network and the best grandmother cells.

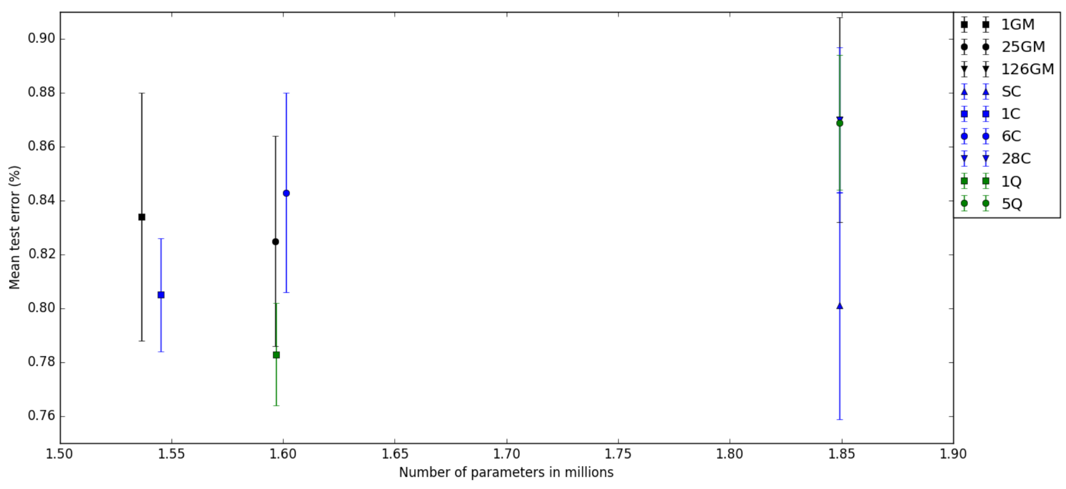

Figure 4.7: Results summary for a deep network on the MNIST data-set.

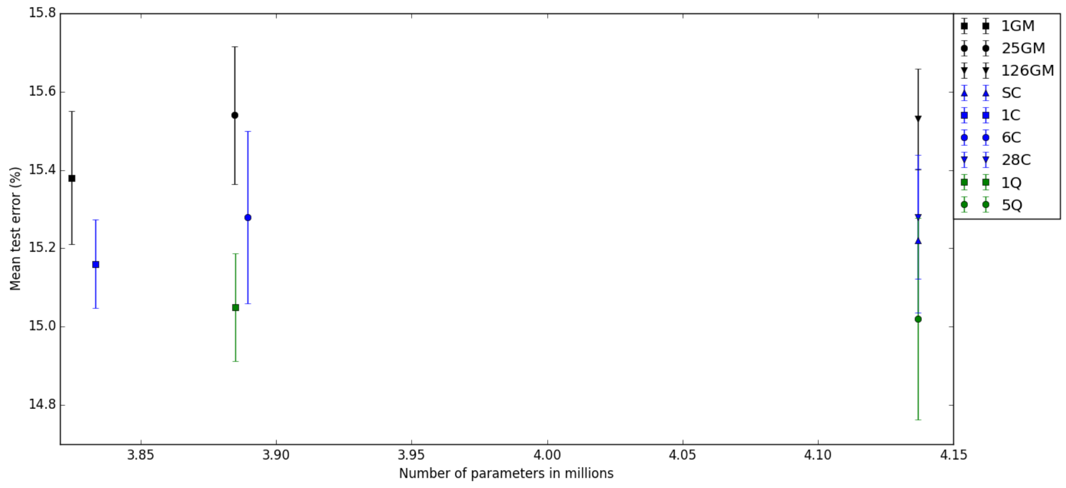

Figure 4.8: Results summary for a deep network on the SVHN data-set.

On both MNIST and SVHN, these results show the interest of the quintuplets code which gives better performance than most other output codes for a comparable number of parameters. The only case where a code outperforms the quintuplets is the "special couples"(SC) code of non-repeated couple cells clusters, which beats 5Q on MNIST. It is also worth noting that the non-repeated couples perform better than the repeated 28C on both data-sets.

4.5 Problems with a large number of classes

4.5.1 Memory constraints

The number of classes in Machine Learning problems is today usually inferior to 1000. With the augmentation in volume of data available and the progress of classification systems, data-sets with large number of categories are bound to become common-place in the span of a few years. In this frame, the way that classes are represented is critical. Indeed, in a problem with 1.000.000 classes, a learning neural network with a last hidden neurons layer comprising 5000 units would have 5.000.000.000 synaptic weights only in its last matrix. Using four bytes per weight value, this would bring the memory requirements up to 20 GigaBytes for this sole matrix, beyond the capacity of most current commercially available GPUs. Assembly codes are especially interesting in this setting, as they can drastically reduce the size of the output layer. In the above example, one may chose to represent the different classes using a constant-weight code with three units active in a set of only 200 output neurons [109]. The number of distinct codewords would be sufficient to handle the 1.000.000 classes, and with 200 output units, the top layer would contain 1.000.000 synaptic weights. This would represent only 4 MegaBytes, which is easily handled with affordable graphic cards. Finally, we know that the cortex has evolved a complex folding of its outer-layer grey matter, in order to store more neuron somas in a skull of reasonable size [110]. Similarly, it is reasonable to suppose that evolution has found and retained efficient coding strategies for the neural representation of concepts. In such a scheme, a given unit, *e.g.*, a micro-column, may well often be used to support different, partially overlapping categories.

4.5.2 Error-correcting decoding

Another advantage of using assemblies to represent categories is the possibility to combine the deep neural network with an error-correcting code. A rather simple example of error-correcting code is the Hamming code, which uses parity check bits to detect the presence and location of errors [111]. The Hamming (7,4,3) code uses seven bits per codeword, of which four represent useful information and three bear added redundancy, i.e., the parity checks. In this code the bits in the first, second and fourth positions are the parity checks. Let us denote by d_1, d_2, d_3 and d_4 the four bits of data. They are positioned in the third, fifth, sixth and seventh spots in the word, respectively. The first bit of a word represents the parity of the sum $d_1 + d_2 + d_4$, the second encodes the parity of $d_1 + d_3 + d_4$ and the fourth bit is the parity of $d_2 + d_3 + d_4$. A parity bit is put to 0 if the sum is even, and 1 if the sum is odd.

At the reception of a codeword, one can detect transmission errors by recomputing the parity bits based on the received four core bits. Summing together the positions of the parity bits found to be erronated gives the position of an erroneous bit of data. The Hamming (7,4,3) code can repair up to two bit errors.

The product code associated with the Hamming (7,4,3) scheme consists in a 7*7 matrix, whose first four rows and four columns contain the data [112]. In this embedded 4*4 matrix, each row and each column is treated as the data in a Hamming (7,4,3) codeword, and is thus used to compute three parity check bits. These bit triplets are appended at the end of said row or column in the 7*7 matrix. Finally, the four top bits in the fifth, sixth and seventh columns are used to compute parity check bit triplets, that are put in the three lower spots of these columns. Computing parity bits from the four leftmost bits in the three last rows yields the same nine values. This principle is illustrated by Figure 4.9.

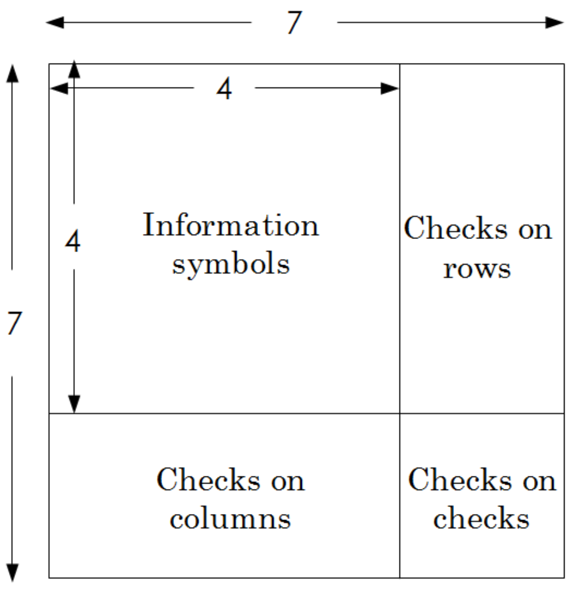

Figure 4.9: Product code structure obtained with the Hamming (7,4,3) code.

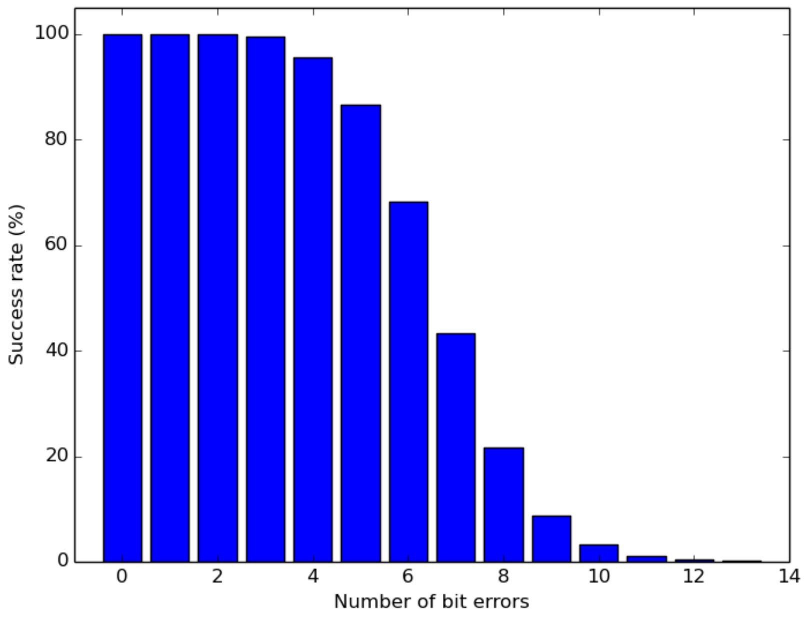

Figure 4.10: Hard decoding performances of the Hamming (7,4,3) product code per number of randomly inserted bit errors, obtained by iterating twice over all lines and columns.

The decoding process of this code simply consists in performing the Hamming (7,4,3) decoding procedure iteratively on all rows and then all columns, alternately. Figure 4.10 shows the average decoding performances of this product code as a function of the number of bit errors.

The capacity to decode a corrupted codeword directly depends on the corruption pattern, and not on the codeword itself.

4.5.3 Experiments

A typical problem involving a large number of categories is the classification of written words. We experiment the combination of a deep neural network with a Hamming product code on a part of the synthetic data-set introduced in [113]. This set is made of synthetic images of words written in random fonts, in a randomly selected grey-level, over a random grey-level background. The words are also rotated and distorted randomly. Figure 4.11

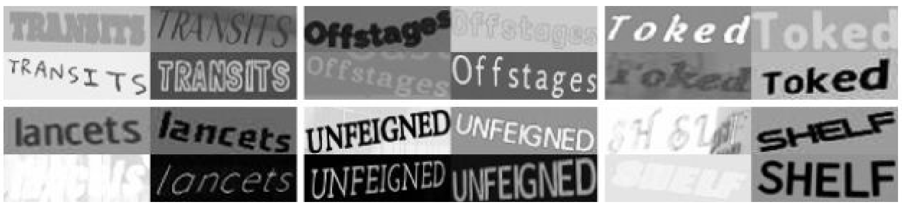

Figure 4.11: Sub-samples of the available images in the Synth data-set for words 'transits', 'offstages', 'toked', 'lancets', 'unfeigned' and 'shelf'.

shows samples of images from this data-set, representing 6 different words.

We extract all images corresponding to the first 1.000 words in the data-set, and obtain a training set made of 28.593 images and a test set with 3.436 images. Our network architecture is based on that in [113]. It has five convolution layers containing respectively 64, 128, 256, 512 and 512 kernels in ascending order. Kernel sizes in the five layers are 5, 5, 3, 3 and 3. Two-dimensional max-pooling layers are added on top of the first, second and fourth convolution layers. This convolutional stack is followed by two successive fully-connected layers of 4.096 neurons each. Both of these layers are followed by a dropout regularization operator with probability 0.5 [81]. All of these convolutional and fully-connected layers have a ReLU activation function [80]. Finally, a fully-connected output layer with 49 units and a

Table 4.2: Classification performances with and without error-correcting decoding.

	Initial classification performance	Classification after decoding	Initial average number of output bit errors	Post-decoding average number of bit errors
Training set	93.71%	94.78%	1.64	0.58
Test set	83.00%	84.17%	2.6	1.46

soft-max activation function completes the architecture. All layers are initialized using the Glorot uniform method [114].

The output matrix contains $4.096 * 49 = 200.704$ synaptic weights, instead of 4.096.000 if a $1 - of - K$ code was used to represent classes. The compression ratio obtained for this part of the network is thus 20.4.

Experimentally we find convergence to be more difficult when using variable-weight objective codes. To encode 1.000 classes using 16 bits, we thus pick 1.000 binary words out of the 1.820 possible code-words of length 16 and weight 4. Convergence is achieved without problems on these constant-weight 16-bits words in 2.000 iterations. The parity bits are then appended to these objective vectors, first with the six parity checks of the first two rows, and then the six parity checks of the third and fourth rows. Both times, training is pursued for 2.000 iterations and achieves convergence again after an initial divergence phase. The bits in the fifth, sixth and seventh rows of the matrix are then successively added to the target vectors and training is each time proceeded with. The reasons justifying this incremental approach are similar to those described in Subsection 3.2.3.

At test time, the ouputs resulting from each sample are binarized using a threshold value that is calculated so as to keep four active bits among the 16 core bits of the product code. The resulting binary vectors are decoded using the product code procedure.

4.5.4 Results

Table 4.2 shows classification results before and after error-correcting decoding on the training and test data-sets.

It appears that the error-correcting procedure effectively corrects some bit errors, although the average number of bit errors per output vector is already inferior to three on both sets. As a result, there is a noticeable improvement in classification thanks to the product code operation, which accounts for 304 training images and 40 test images.

Finally, the reduction in output size due to the use of a population code reduces the complexity of the classification. When an output vector is computed, it is multiplied by all class codes in order to find the highest dot-product. For each image, these 1.000 dot-products involve 1.999.000 single operations with the $1-of-K$ code. Using output vectors of size 49, they only take 97.000 operations, which is 20.6 times less. As a comparison, the decoding of a 49-bit codeword using the Hamming (7,4,3) product code hard decoding procedure with 2 iterations represents less than 336 operations, which is virtually negligible.Even larger gaps in complexity may be obtained for classification problems with tens of thousands of categories. Also, much larger compression ratios could be achieved on such problems.

Chapter 5

Combination of Unsupervised Learning and Associative Memory

As it was shown in Chapter 2, neural associative memories are especially efficient for completing partially erased pieces of information. This property may prove itself useful when dealing with occlusions in natural images, for example. However, like most data in the real-world, the pixel values in pictures usually feature a highlevel of correlation, when neural associative memories deal more easily with random independent, identically distributed patterns.

Gradient-descent based learning neural networks bring the possibility to pre-process data and convert it into binary codes. In Chapter 4, we introduced a way to train a learning neural network to output sparse distributed binary codes associated with categories. Unsupervised training is an open problem that consists in extracting relevant features from a data-set without access to any category information.

We hereby introduce a way to combine the feature extraction capabilities of hierarchical neural networks trained through gradient descent, with the pattern completion ability of neural associative memories, in order to deal with corruption in real-world data. Also, as most available programming frameworks for training deep networks are adapted to run models transparently on massively parallel Graphic Processing Units, we detail ways to run two decoding schemes for Sparse Clustered Associative Memories on this type of device, thus avoiding cumbersome data transfers and speed bottlenecks.

5.1 Introduction

The neural system is trained throughout our lifetime to detect all kinds of patterns, associated with all the sensory channels but also on the level of mental concepts. These patterns, be they sensory or abstract, can cover very wide ranges of scales and complexities. In addition to being able to recognize patterns when we are confronted with them, we have the capacity to evoke them in our minds, and to complete them given partial clues. Good examples of this are our natural propensity to mentally complete familiar melodies when hearing the first notes, or written words where a few letters are missing.

It appears that learning patterns, and then being able to recall them, are two fundamental components of human cognition. Nowadays, layered neural networks trained via gradient descent are the preponderant model for feature extraction from high-dimensional data. Sparse Clustered Cliques Networks are the state-of-the-art among connectionist associative memories in terms of pattern completion and storage capacity. Using these two models, we propose a framework that combines feature extraction by Deep Learning, and association between cliques of pattern detectors using the sparse CCN scheme.

5.2 Gradient descent learning

The term Deep Learning designates a group of neural network algorithms relying on hierarchical processing and synaptic weight optimization by gradient descent. A deep network is typically composed with a series of layers of formal neurons, interconnected by synaptic weights matrices. Input vectors are fed to the entry layer and activity is propagated towards the output by a succession of vector-matrix products, with additive biases and

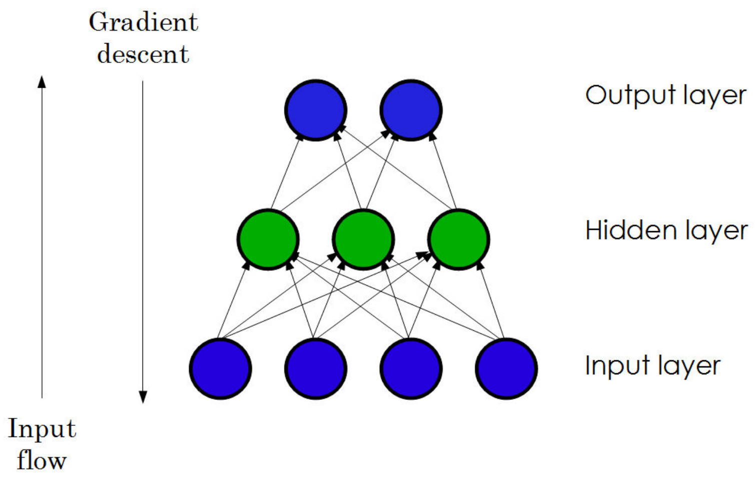

Figure 5.1: Principle of a mutli-layer perceptron.

specific activation functions applied on the output of each product.

Figure 5.1 shows a toy example of deep network with one hidden layer of neurons.

Such a network is called a perceptron [120].

The gradient descent procedure consists in differentiating this loss with respect to every parameter in the network, and then modifying parameters proportionally to their individual contributions.

Through multiple iterations of this process, one can train the network to associate together a large number of high-dimensional input and output vectors, with only little error overall.

The most typical use case of this system is to train it to map inputs to categories they belong to. This method is called supervised learning for classification, and necessitates access to the categorical information associated with each input sample. In the case where no categorical information is available about the data, one can nonetheless extract a relevant dictionary of features using unsupervised learning. The common neural network architecture to perform this is the auto-encoder, a perceptron trained to reproduce its input vector on the output layer.

5.3 Sparse Clustered Associative Memories

5.3.1 CCNs

Clustered Clique Networks (CCN) are composed of a set of N neurons, divided in clusters all containing the same number of neurons. A connection can be drawn between two of these neurons only if they belong to different clusters. A memory element is then supported by a choice of neurons all located in different clusters. The storage of such a message in the network consists in connecting all of its neurons together, forming a clique.

By the separation of the network into clusters, it makes use of sparseness to a higher extent than former states-of-the-art neural associative memories, thus lessening the amount of overlapping between stored messages and easing the search of the unknown elements of patterns to complete. During retrieval, the commonly used procedure consists in propagating activation from the neurons of a request through all their existing connections to other neurons, followed by a selection of neurons based on the resulting activity scores. A local Winner-Takes-All rule is generally used to perform the selection. It consists in keeping active only the neurons possessing the highest activity score in each cluster. This is an iterative process, and it is repeated as long as the stopping criterion is not reached. Activation is computed again with the selected neurons, and the rule is applied once again.

5.3.2 Sparse messages

Sparse Clustered Clique Networks as introduced by Aliabadi *et al* [42] have the same principle as classic CCNs. The main difference is that stored messages no longer need to possess a neuron in every cluster of the network, but only in a selected subset of the clusters. This characteristic raises the diversity of messages, that is the total number of different messages one can store in the network. However, it brings the added difficulty of not knowing which

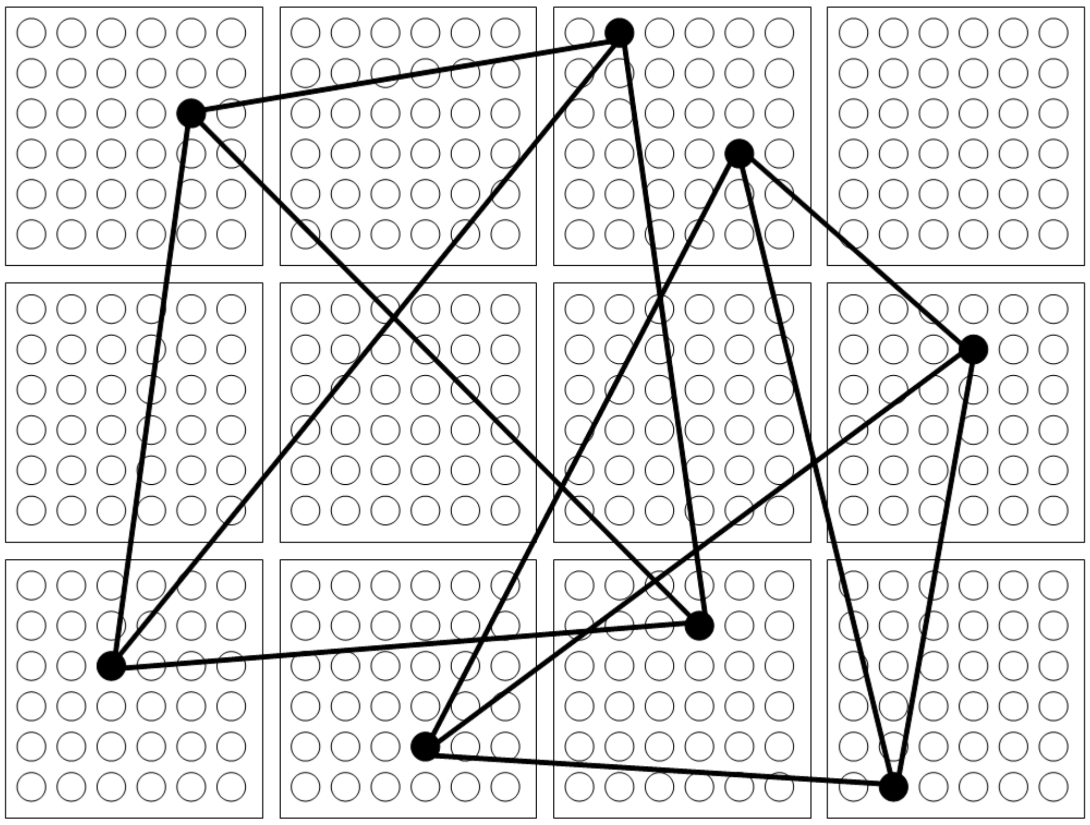

Figure 5.2: Example of a sparse Clustered Clique Network with 12 clusters and messages of order 4.

clusters support the searched message. The high level of sparsity usually makes up for this, and the retrieval error rates are much lower in general as compared to full messages CCNs. Figure 5.2 shows an example of such a network.

This specific case can benefit from an adaptation of the retrieval procedure used in classic CCNs. Aboudib *et al* [47] introduce two such variants of the Winner-Take-All algorithm. These are the Global Winners-Take-All (GWsTA) and the Global Losers-Kicked-Out (GLsKO), and come with a significant improvement in retrieval success rate.

The present work focuses on the acceleration of these algorithm variants by means of Graphical Processing Units (GPUs). These chipsets allow the massively parallel computation of operations on data, and are currently widely used to accelerate neural network implementations. To achieve this, one must ensure a sufficient amount of operations to execute can actually be performed in parallel without altering the accuracy of the whole computation. The main requirement for this is the independence of the parallelly processed data. We aim at making clear here how we adapt the former serial implementation under this constraint. We use the Compute Unified Device Architecture (CUDA) which is a coding framework specifically designed to program GPUs, supported by C/C++. Yao [116,117] formerly achieved acceleration of the classic CCN algorithm using CUDA.

5.3.3 Specific retrieval algorithms for sparse messages

When retrieving sparse messages, the mere selection of the most active neuron in each cluster can no longer be relied upon as not all clusters contain an element of a message of interest. Not knowing which clusters support a given message therefore brings up an additional difficulty. The retrieval of sparse messages can be performed more efficiently by using strategies specifically tailored for this case. These will typically make use of the set of activity scores observed after stimulation instead of solely focusing on local winners. Two such strategies can be found in [47] namely the Global Winners-Take-All and the Global Losers-Kicked-Out. These two algorithms were originally developed to improve retrieval of sparse messages in CCN. However, they are unsurprisingly practical for the modified Willshaw network as well, where no $1 - of - K$ rule applies either, and the use of the GLsKO is documented in Chapter I.

In Global Winners-Take-All, one wants to retain a number of neurons equal or superior to a predefined minimum. The minimal threshold score for a fanal to be retained is adapted in order to fulfill this requirement. After the first activity computation step, we iteratively select a number of fanals which are then used to compute activity for the next iteration. For instance, if after a stimulation of the network, there are 7 activated fanals with scores {2,3,1,4,3,4,4} and we know at least 4 neurons must be retained, a lower threshold of 3 will be applied on the scores of neurons to remain active. This way the 5 neurons with an activity of 3 or more are kept as output for the ongoing iteration, while the 2 neurons with scores 1 and 2 are discarded.

In Global Losers-Kicked-Out, focus is put on active neurons with relatively low scores. After the initial activity computation, the aim is to discard a number of fanals iteratively, instead of selecting them. Among the non-zero scores observed after stimulation, a fixed number of lower score values is set so that neurons with these activity levels are put to zero. In the previous example, if we know 3 lower score values must be discarded, this will designate the 4 neurons in the list with scores {2,1,3,3} and only the 3 neurons with an activity of 4 will remain active. Nonetheless, it has been shown that banning only the one minimal score in the list gives the best results.

5.3.4 Accelerations using GPUs

Graphics Processing Units (GPU) are highly parallel processors. They are based on a high number of processing units, which can be used to process data using a Single Instruction Multiple Data (SIMD) paradigm. In this paradigm, the same instruction can be applied simultaneously on all elements of a vector. As a result, GPU can be very efficient when processing algorithm without data dependency. For example, loops in which each iteration can be performed regardless of previous iterations can be very efficiently accelerated. Most Deep Learning libraries are adapted to perform parallel computations on GPU. During the phase of activity propagation towards the output layer, the matrix-vector products are thus accelerated by parallel computing, as well as the application of biases and activation functions. The differentiation of the cost with respect to the different parameters, and their subsequent updating also leverage the parallelism of the GPU. In the perspective of combining Deep Learning with Sparse Clustered Cliques Networks, a parallel implementation of these associative memories can be useful as it allows the fast completion of requests directly on GPU. In addition to accelerating computations in comparison with a sequential execution of the same algorithm, this suppresses the need to copy data to the host and back again to the GPU, which would itself be time-consuming. For this reason, we propose parallel implementations for the GWsTA and the GLsKO algorithms.

5.3.5 GWsTA

Global Winners-Take-All aims at retaining a number of active fanals superior or equal to a predefined minimum. In order to do this, we compute for each possible score value the number of fanals that reach it. In practice, scores counts are first computed in parallel for the different clusters, with one thread computing the counts for the different scores inside of one cluster, as shown on Figure 5.3. In a second step, the scores counts are accumulated by reduction to get their totals over the whole network. A number of threads is launched, equal to half the number of clusters. Each thread sums the vectors of scores counts of two clusters located in different halves of the network. The operation is repeated on the new set of counts lists and so forth, each time halving the number of computed values until one single list remains with the global sums. A final thread then runs a loop over the possible score values in descending order, accumulating scores counts until the sum equals or exceeds the minimum expected number of active neurons. As soon as this criterion is met, looping stops and the last considered score value is kept as the minimum score for nodes to remain active. A thresholding with binarization is then applied on nodes activities, with scores above the threshold being put to one while other neurons are all at zero. This operation is performed in parallel, where each thread computes the output values for one neuron as illustrated by Figure 5.4. The resulting set of active fanals is then returned as output for the current iteration.

5.3.6 GLsKO

For our implementation of the Global Losers-Kicked-Out we set to 1 the number of low scores to be discarded. This is known to give the best results, and it reduces the complexity of the algorithm, since we only need to find the minimum score and discard neurons with

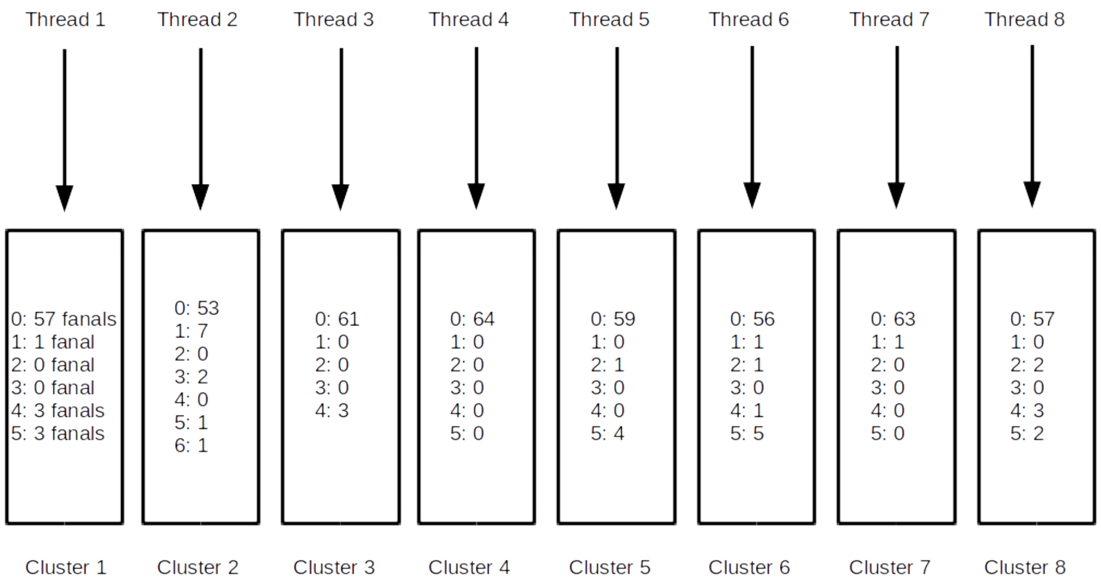

Figure 5.3: Example of the parallel computation of local scores counts in the different clusters. Each thread computes a vector of the numbers of fanals possessing the different possible score values in one cluster.

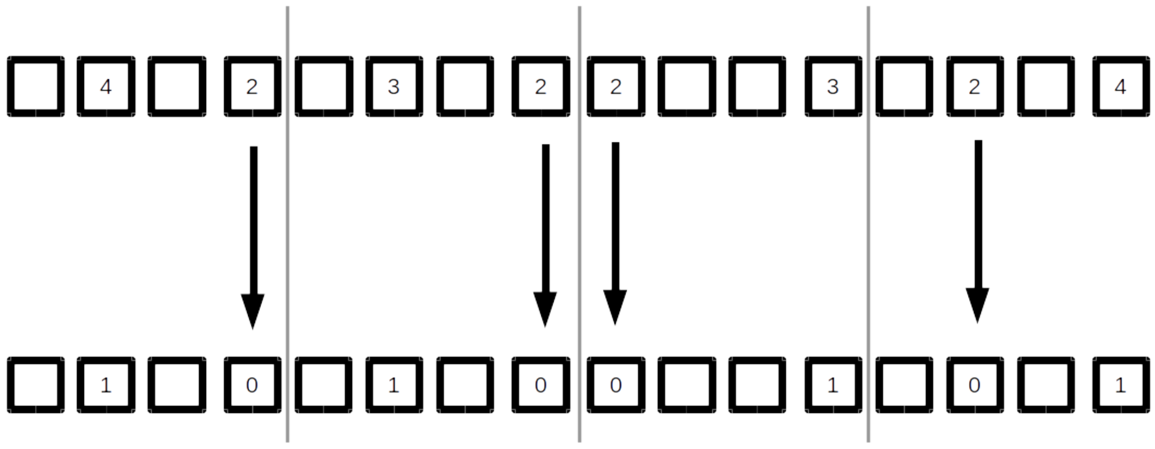

Figure 5.4: Example of the threshold-binarize operation as performed on GPU. Non-zero scores below the threshold are put to 0 while those above it are set to 1. Input and output values are in one-to-one relationships, hence computation can be performed in parallel.

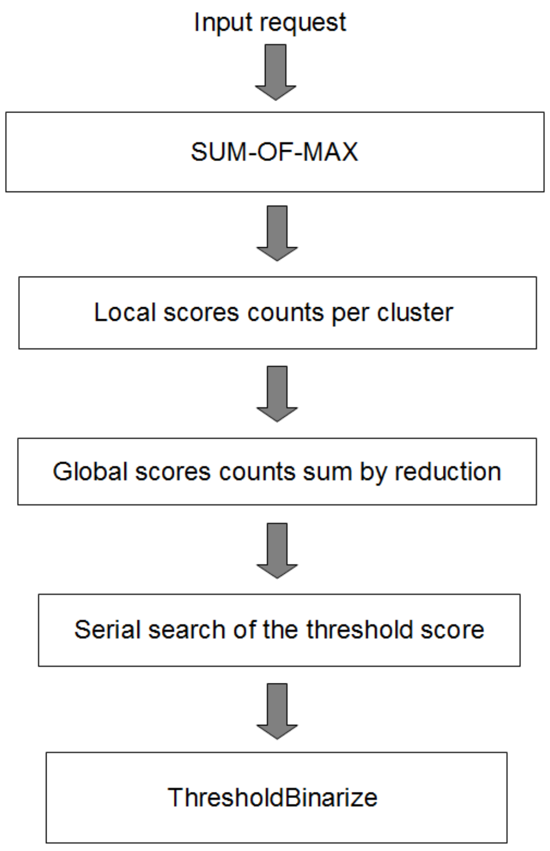
Input request
SUM-OF-MAX
Local scores counts per cluster
Global scores counts sum by reduction
Serial search of the threshold score
ThresholdBinarize

Figure 5.5: Illustration of the GWsTA procedure after the first iteration of retrieval. All operations in rectangular boxes are performed on GPU.

this score [47]. Hence we focus on finding the global minimum among all non-zero scores present after stimulation. To this end the minimum positive scores inside of the different clusters are first computed in parallel. A reduction strategy is employed on the resulting set of local minimums to compute the global one. At every step of the reduction, each thread compares the minimums obtained for two clusters, keeping the lowest of the two values. The set of values is thus halved iteratively to result in a single minimum. Along with the strictly positive minimum, a global boolean value is computed indicating the presence of several different non-zero scores, as described in Figure 5.6. Here the fact that several threads modify the output boolean does not alter its accuracy, as it indicates whether two different non-zero scores are seen at least once. If this boolean is true at the end of the reduction procedure, the ThresholdBinarize operation is applied to compute the iteration output, setting off all nodes with the non-zero minimum score. Otherwise, only one strictly positive score is present and all its representatives are kept active. This case where all stimulated fanals have equal scores is also a good stopping criterion for the GLsKO.

5.3.7 Experiments

Our GPU experiments were performed with an NVIDIA GeForce 780 Ti, with a frequency of 900MHz and 3GB of memory. The CPU used is a 3.3GHz Intel Core i7. Each message is generated randomly by first picking a fixed size subset of the available clusters, and then choosing one fanal in each selected cluster. In the retrieval phase, requests are formed from stored messages by dropping out a fix number of their nodes.

Figure 5.8 shows the execution times in seconds for the retrieval of 500 stored messages as a function of the number of erasures per queries, in networks with 16 clusters and 32 fanals per cluster. Each message is made of 12 neurons and the corresponding query is a subsample from the message with a size going from 11 down to 1.

The execution times are very close at the beginning and differ when the number of erasures exceeds about half the message size, due to the different stopping criteria employed.

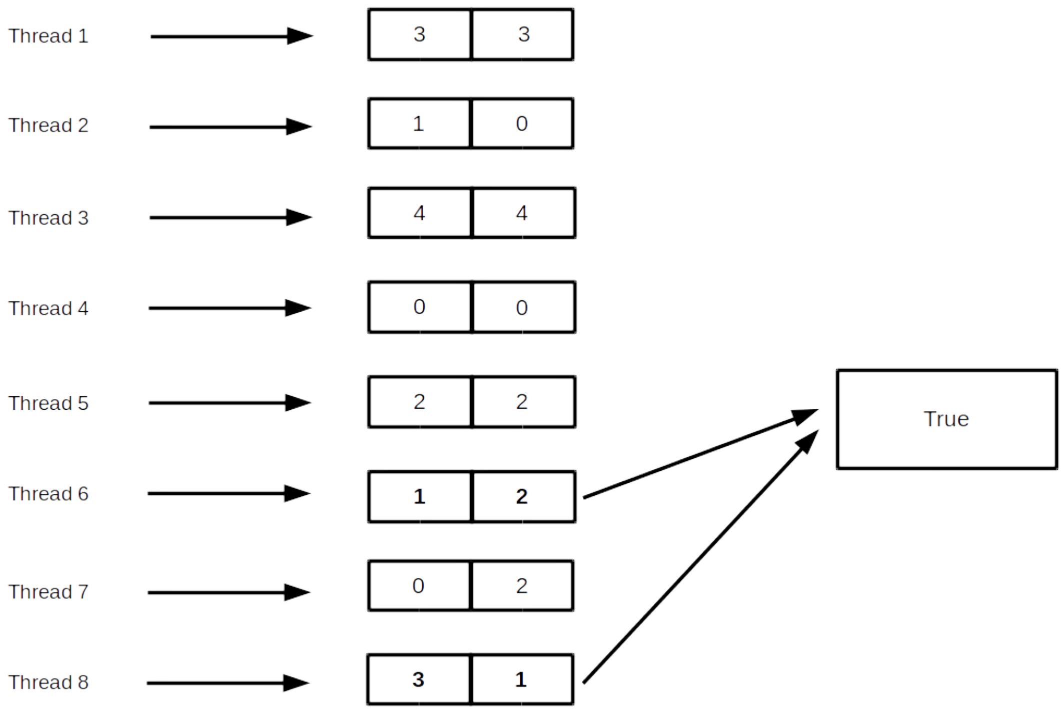

Figure 5.6: Illustration of the operation SeveralScoresPresent performed in the parallel implementation of GLsKO. This operation is performed in the same threads as the search of the non-zero minimum by reduction among local minimums. Each thread compares the minimum scores of two clusters to assess whether they are different and strictly positive. Several threads may access and modify the output boolean concurrently without affecting the result's accuracy.

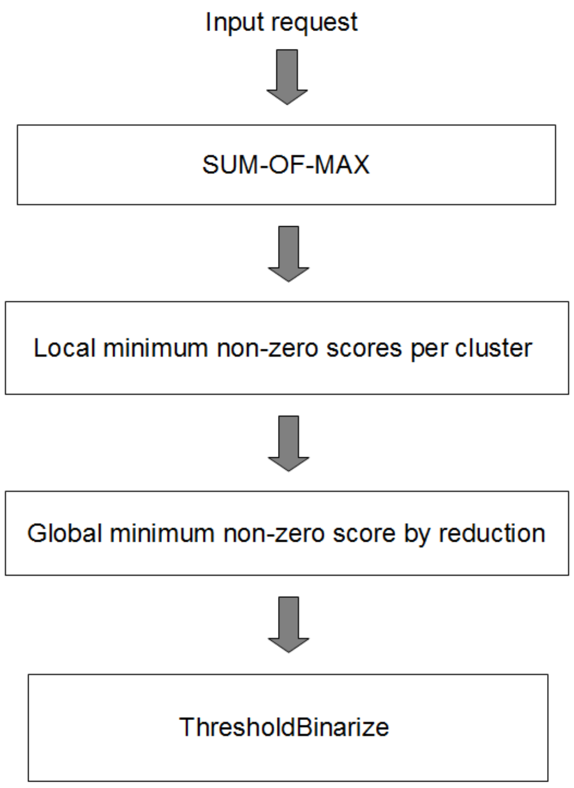
Input request
SUM-OF-MAX
Local minimum non-zero scores per cluster
Global minimum non-zero score by reduction
ThresholdBinarize

Figure 5.7: Illustration of the GLsKO procedure after the first iteration of retrieval. The global minimum non-zero score is computed by reduction, along with a boolean indicating if several non-zero scores are present. The output is computed using the parallel ThresholdBinarize operation, excluding the lowest non-zero score in the latter case.

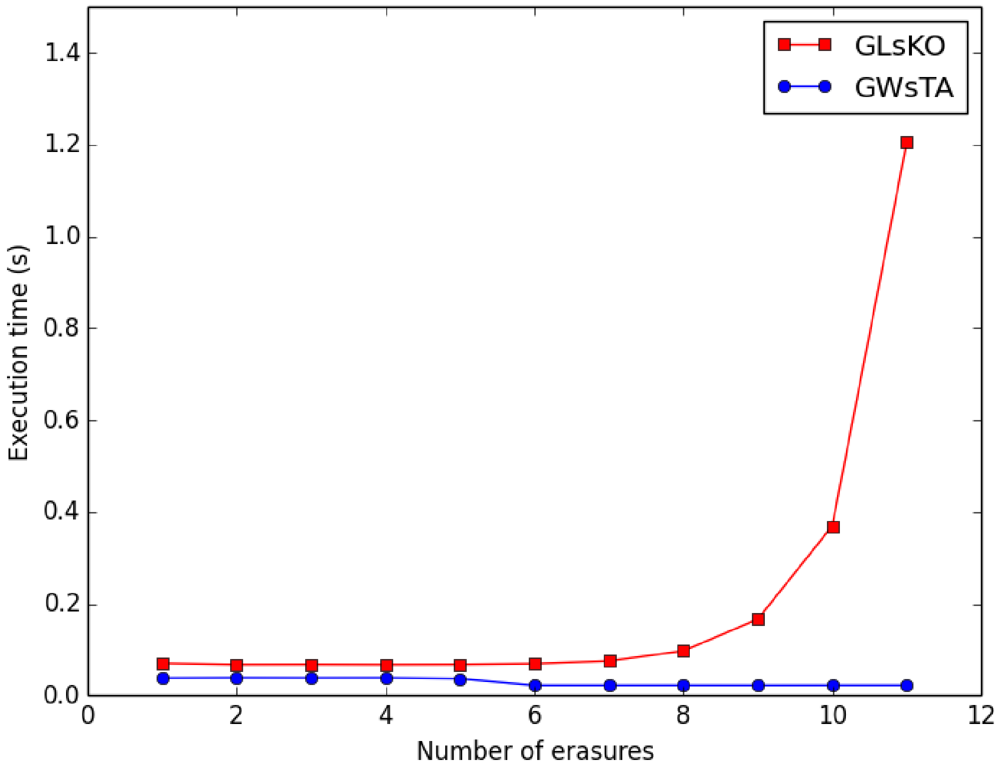

Figure 5.8: Execution time of the parallel versions of GWsTA and GLsKO for the retrieval of 500 sparse messages as a function of the number of erasures per query, in a network of 16 clusters and 32 fanals per cluster. Messages are made of 12 nodes and the size of sub-sampled queries goes from 11 nodes to 1.

For higher number of erasures the GLsKO takes more and more time, with a maximum when the request is made of a single neuron. The stopping criterion used is the equality of scores of all neurons active after stimulation. The GWsTA scheme takes less time for higher numbers of erasures. Its stopping criteria are convergence, *i.e.* equality between input and output at a given iteration, and a maximum number of iterations, hereby 8.

Figure 5.9 shows the execution time of the two schemes for networks of 64 clusters of 724 fanals each and 500 stored messages. Here messages are made of 56 nodes and the number of erasures applied to create the corresponding queries varies from 2 to 54. The two schemes make no retrieval error except for the GLsKO which fails to retrieve one pattern out of 500 when provided with 2-nodes requests. The number of iterations is impacted and alters the speed of retrieval.

With 2 erasures the GLsKO scheme takes 0.093573 seconds to complete retrieval of the whole set of messages, when a serial implementation of the same algorithm takes 4.74341 seconds running on CPU. This brings an acceleration factor of 50.69. Parallel and serial versions of the GWsTA use respectively 0.0246 seconds and 4.16881 seconds in the case with 2-nodes requests, giving an acceleration of 169.319. To be fair, it has to be stated that our serial and parallel implementations are not tailored to leverage a high sparseness of the connectivity graph, which could bring further improvements. To this end, one would want to represent the set of connections as lists of indexes instead of a boolean matrix.

5.4 Combination of learning and memory

Hearing the first notes of a melody or the first words of a proverb often triggers an automatic mental completion, as well as seeing a partially erased picture of an object or a known face is usually enough to activate the mental image present in memory. It is most likely that in these real-world cases, the mental completion of the missing parts of known patterns results from a high-level process that happens remotely from the input sensory neural groups. To

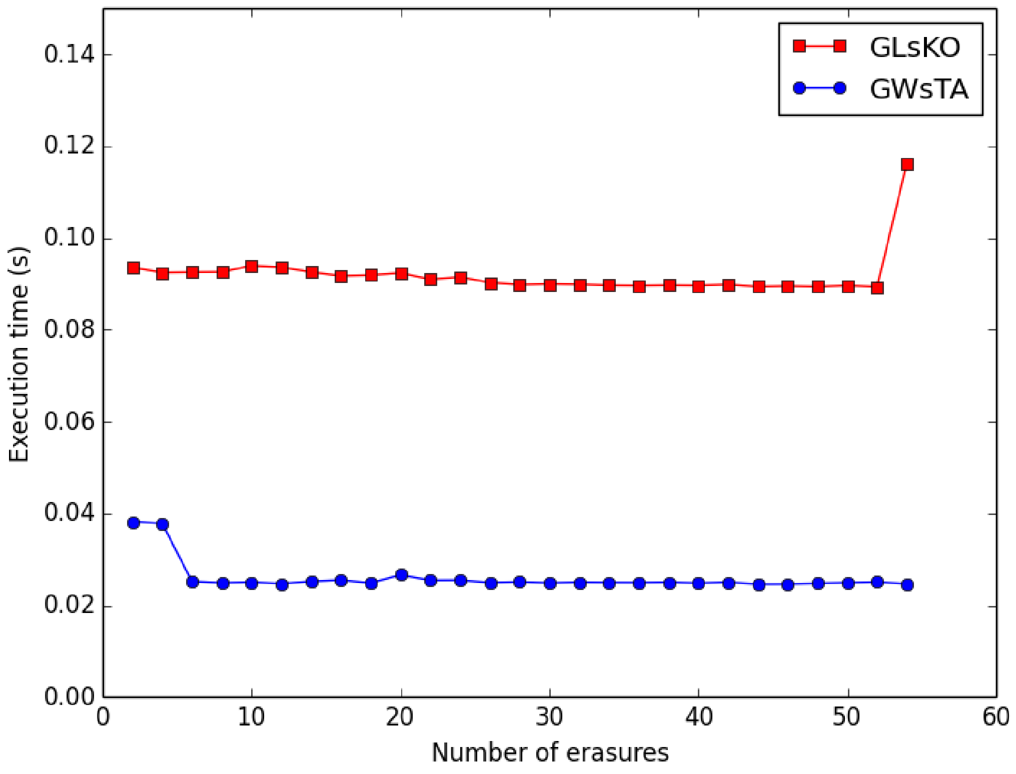

Figure 5.9: Parallel execution times of the two schemes for the retrieval of 500 messages of 56 nodes in a network with 64 clusters and 724 fanals per cluster. Queries size goes from 54 down to 2.

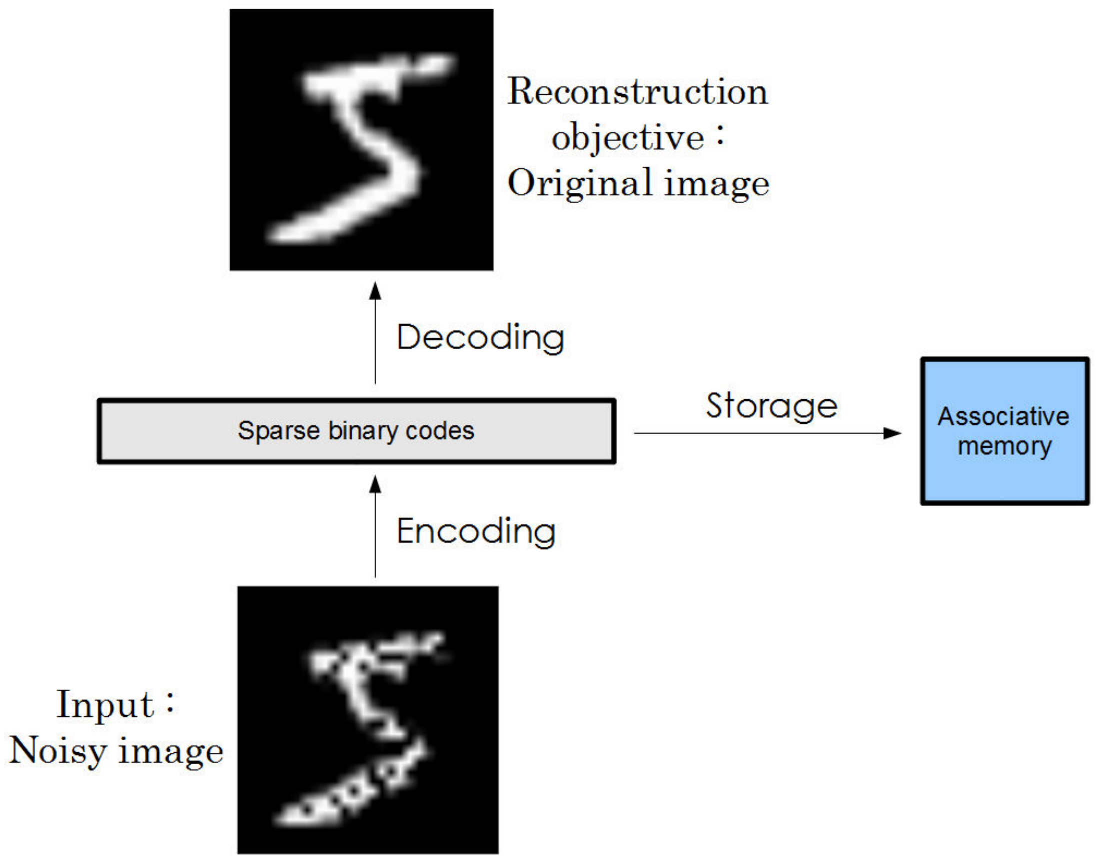

Figure 5.10: Principle of the combination of an auto-encoder with a sparse clustered associative memory. The auto-encoder is first trained for a few iterations, then is used to get sparse binary encodings of input images. These codewords are stored in the Clustered Clique Network and can be used to reconstruct partially occluded images.

reproduce this behavior artificially, it would then be way more sensible to implement these associations between abstract high-level features, rather than between low-level entries such as the pixels in an image. We therefore propose a generic architecture that combines learning and memory, based on a deep learning network and a sparse clustered cliques network.

Figure 5.10 illustrates the principle of this system.

For simplicity, the simulated network is based on a denoising auto-encoder [122] with only one hidden layer, as the data-set used has small dimensions which do not make a deeper architecture necessary. During training, each pixel in an image is put to zero following a Bernoulli law with probability 0.3. A sigmoïd activation is applied on the hidden layer as well as on the output layer. The connection weights in the encoding and decoding matrices are tied, meaning that they are forced to be exact transposes of each other at all time. The hidden layer is made of 5.000 neurons parted in 2.500 clusters of size 2. In order to enforce sparsity and to spread activation patterns across the layer, each image is assigned a subset of 128 clusters. When the network takes an image as input and the hidden layer is activated, the values of the hidden activity vector located in the 128 clusters associated with this particular image are left untouched, while all the other values are forced to zero. A clustered K-sparse activation scheme is then applied, in which the lowest activity among the two neurons of a cluster is put to zero [123,124]. The auto-encoder is implemented using the Theano library, which generates CUDA code in order to execute activity propagation and gradient back-propagation on GPU [121].

When the training phase of the auto-encoder is complete, activity is propagated one last time to the hidden layer using the sigmoïd and K-sparse activations. The resulting sparse activation vectors are binarized, and thus correspond to possible code-words of a Clustered Clique Network with 2500 clusters of size 2, and cliques of order 128. These clique patterns are stored in a binary connection matrix that implements the associative memory. In addition to being binary, this matrix is different from the two weights matrices of the auto-encoder in that its input and output indices correspond to the same set of neurons, which is characteristic of a recurrent neural network [125].

The auto-encoder is trained on a subset of 1.000 of the training images in MNIST for five iterations. These images are then passed through the encoder to get 1.000 distinct sparse binary codes. These patterns are stored in the Clustered Clique Network which, given 64-nodes associated queries, is able to complete all these cliques correctly. Partially occluded images are created by applying a cross-shaped mask on the four central rows and the four central columns of training images. These masked images are given as input to the auto-encoder, with a different activation scheme following the first sigmoïd: the highest 64 activities are retained while the rest is put to zero. Resulting activation vectors are binarized

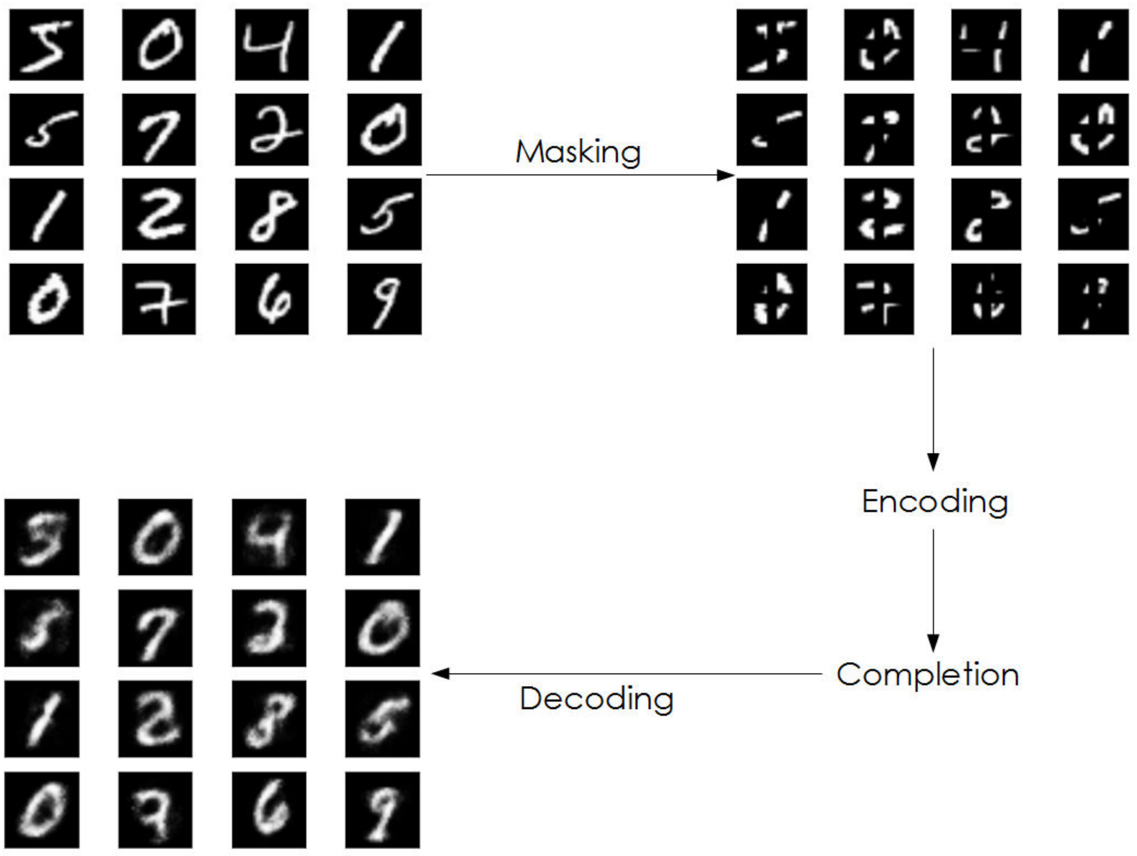

Figure 5.11: Processing chain showing samples of original, masked and reconstructed images after code completion by the associative memory.

and given as queries to the associative memory, which proceeds to complete them. The outputs of the CCN are placed back on the hidden layer of the auto-encoder and multiplied by its decoding matrix. A sample of reconstructed images can be seen on Figure 5.11, which shows that this system is robust to erasures.

The GPU implementation of the sparse Clustered Cliques Networks is useful here to avoid a speed bottleneck. The encoding of a single image only takes around 0.6 milliseconds using Theano, where the completion of 64-nodes queries takes an average of 0.2 seconds using Python. The parallel version of the CCN needs only 0.36 milliseconds to complete a pattern. When processing 1.000 images in a row, the full chain of treatment of occluded images is thus taken from more than three and a half minutes using the sequential implementation of the memory, down to less than two seconds with the parallel scheme.

Chapter 6

Conclusion and Openings

6.1 Conclusion

This work has been driven by the search for implementation methods of learning and memory using artificial neural networks, with computational efficiency and biological plausibility as primary guidelines. These two objectives are inherently connected, as biology is a model of efficient use of material and energy.

We have assumed that a plausible neural network should be robust to alterations of both its input and internal signals. Also, it should perform tasks quickly, by using relatively few units and elementary operations.

The robustness of sensory and mental information must result from redundancy in its supporting material. Clique-based associative memories exemplify this with their redundant connection patterns that directly contribute to their strong pattern completion abilities. Also, a regularization technique like Dropout, employed in training deep networks, enforces redundancy as various random subsets of a pool of feature detectors are trained on each training sample. As any fixed-size, random subsample of the features can be brought to interpret any image, it has to contain a sufficient vocabulary on its own. The resulting feature detectors are thus partly redundant with each other. Also, Dropout prevents single feature detectors from overfitting on very specific elements in the data, and promotes their focusing on redundant parts that better define the data distribution. It thus brings a layer of a deep learning network to behave more like a low-pass filter. Also, it helps fit the Shannon-Weaver model canvas. By representing redundant elements present in the input signal by the activation of single units, it removes unnecessary redundancy from the output code, like a compression algorithm would. By enforcing the ability of subsamples of feature detectors to interpret input vectors autonomously, it promotes inter-features redundancy that likely improves the robustness of the learner to erasures and distortions of the input.

The mid-level representations resulting from the regularized training of deep neural networks tend to be sparse and distributed, like the patterns in clique-based neural associative memories. This convergence is not a complete coincidence as distributed sparse codes are a strategy of choice for efficient use of neural material. The typically large number of possible combinations of neurons brings a high diversity of patterns. Moreover, each unit can be reused in several representations, that can still be distinguished easily even if they partially overlap. Several neuroscientific studies suggest that the brain does indeed use such representations [126–128].

Sparse distributed representations allow to use relatively little material to encode a given amount of information. As such, they can reduce the complexity of training incoming synaptic weights, and the amount of energy needed to do it.

The use of distributed representations is also associated with parallel computing, which is essential to the performance of the brain. When modern central processing units perform at a speed of about one instruction per nanosecond, biological neurons have a maximum firing rate close to 250 Hz. Also, synaptic spikes travel at a very low pace of around 20 meters per second. However, the brain can perform complex tasks quickly thanks to its highly optimized parallel algorithms. This is why GPUs are essential in modern implementations of neural networks.

In Chapter II, we presented an associative memory model using connectivity constraints based on the locations of neurons. This relies on the idea that the known phenomenon of lateral inhibition due to short-range inhibitory connections can be modeled as a prohibition of the co-activation of close-by neurons. It brings increased sparsity and reduced overlap between patterns, and thus enforces the pattern completion performances of the model. Also, it increases the storage efficiency and hence the compression ratio applied on stored information.

Chapter III detailed a study of the biological plausibility of deep convolutional neural networks on a functional level. Several linguistic and graphic parameters were found to favor the recognition performance of the convolutional network, as they do for humans. In addition to the display rate, we found that the positive influence of lexical frequency of words can be reproduced without integrating any semantic treatment. The letter feature types found to favor recognition by the convolutional network were the same overall as for human subjects. Tests involving 2-dimensional embeddings of intermediate representation vectors suggest that the presence of certain letters, bigrams or trigrams in the word can be more determinant in the representation than their specific positions and layout. This opens the question of how well the network would be able to deal with typoglycemia. Convolutional Neural Networks are a model of vision, and making it a more plausible model of reading would probably require substantial adjustments. In particular, integrating characteristics from more high-level models of reading like IAM and FAM would be interesting, such as top-down feedback signal during recognition and lateral interaction between feature detectors at the same level.

Chapter IV introduced a method using sparse distributed codes as training objectives for supervised learning neural networks. This is more biologically plausible than the commonly used $1 - of - K$ code [128]. It also comes with the benefit of strongly reducing the memory footprint of the ultimate layer of the deep network, which can contain the major part of its weights when the number of classes is very large. As a result, the complexity of both training this matrix and classifying its output vector can be importantly reduced. It brings the possibility to combine a supervised deep neural network with a neural associative memory, or with an error-correcting decoder. For this latter option, our experiments showed that the error-correcting decoding could benefit the classification performance. Pursuing this effort on problems with even larger numbers of categories is part of our plans.

Chapter V proposed a simple way to interface an unsupervised learning neural network with a neural associative memory. This revealed the interest of implementing associative memories on GPU, when most programming tools for Deep Learning are already adapted for it. Not only does it bring the execution speed of the memory up in the same range as that of the learning device, it also evacuates the need to copy the memory inputs back to the host machine for CPU processing, and the completed patterns back again to the GPU.

6.2 Openings

In recent years, much attention has been focused on classification problems with a properly crafted training data-set in which label information is fully complete and accurate. In real life, many use cases occur where some samples may be doubtfully tagged, or not labeled at all. Machine Learning researchers are paying more and more interest into these problems today. Based on the findings presented in this work, we foresee at least two lines of future work: one on semi-supervised learning, and another on decision under uncertainty.

6.2.1 Semi-supervised learning

Semi-supervised learning refers to the intermediate case where label information is available for some of the data samples and missing for the other ones. It is a hot area of research nowadays, because it is a common way in which we humans apprehend our environment: we are verbally taught the category of a few items, and we are then able to keep training to categorize new ones without external help, in spite of variations of many kinds from the first examples met [94]. Very recently, new systems based on Deep Learning such as the

Ladder Network [129] and Generative Adversarial Networks [130], specifically designed for semi-supervised learning, have pushed the state-of-the-art further.

GANs in particular have been the object of a lot of attention. They consist of two deep learners: a generator and a discriminator. Like an auto-encoder, the generator is trained to output samples resembling the data, but its input is random noise. The discriminator is alternately trained to classify actual data samples for which label information is available, to dismiss outputs from the generator as being synthetic, and to avoid doing the same for real but unlabeled data. To train the generator, a global network is formed from the two systems, with the output layer of the generator plugged on the input layer of the discriminator. The generator is trained through gradient descent to fool the discriminator, using a loss function based on the probability for generated data to be classified as such by the discriminator. The weights of the discriminator remain fixed during this stage. Over the course of many iterations training both systems in turns, the discriminator becomes increasingly harder to fool and the generator becomes increasingly better at it. For this the generator must capture the underlying distribution of the data, and is then able to generate highly plausible samples that are not simple variations or compositions from actual data samples. As such, Generative Adversarial Networks are a very efficient data augmentation method.

Pseudo-labels are another method for semi-supervised learning, in which unlabeled training samples are assigned estimated labels by the classifier, that are then used as objectives for supervised learning [131].

In experiments, we found that recursively applying the pseudo-labeling method allowed to improve classification performance progressively. On MNIST, starting with only 10 labeled training samples per class, a GAN architecture based on the model in [129] gave an initial test error rate of 1.33% on average. By training this architecture several times starting from different weights initializations, we could produce an assembly of classifiers. Also, the classification step of the GAN is not entirely deterministic, as we keep applying random perturbations on the intermediate layers, such as gaussian noise. As a result, each trained network can be used several times to produce varying sets of pseudo-labels for the same training data. Hence, with a few trained classifiers, we could easily get more than 10 sets of pseudo-labels. Pseudo-labels that would vary little from one set to the next could then be deemed safer. By selecting top-scorers in a majority vote, we could thus increment the supervised part of the training data, from 10 samples per class to 50, then 1000, 2000, 3000, 4000 and 5000. Through this procedure, the error rate on the test set is progressively brought down to 0.87%, and its standard deviation is also significantly reduced.

This method appears very promising for semi-supervised learning. We plan on experimenting with it on other data-sets where the state-of-the-art performance is lower, and to combine it with the assembly output codes technique introduced in Chapter IV.

6.2.2 Decision under uncertainty

Another case of data with partial category information is imprecision of the labels. This can result from data aggregation from many sources, lack of precision of measurement devices, data fluctuation during a measurement time window, etc... Interval-valued data are an instance of this problem that is widely addressed in the literature. However, in some cases such as computation errors, memory alterations or lossy compression, the available information does not come in the form of intervals, but as sets of discrete values [132]. As an example, the suspected inaccuracy of a few bits of high weight in a binary word can result in several possible decimal values spread apart from one another.

Dempster-Shafer theory is an extension of Bayesian theory, in which each event and each combination of events is assigned a mass measuring its degree of belief. The mass of an event or set of events is proportional to the quantity of available information supporting this hypothesis, and the sum of all masses is normalized to one. The plausibility of an hypothesis is equal to one minus the sum of the masses of the sets that do not intersect this hypothesis. For any hypothesis considered, its belief is always inferior to its plausibility.

Table 6.1:
Numbers of classification errors on MNIST test data (10.000 images total), using beliefs and plausibility extracted from three trained networks' outputs and their fusion.

Source	Belief	Plausibility	Belief+Plausibility
Grandmother cell (1GM)	85	85	85
Couples (1C)	79	80	81
Quintuplets (1Q)	82	78	82
Fusion of 3 sources	69	58	60

These two measurements bring several possible decision criteria: the highest belief, the highest plausibility, or the highest sum of belief and plausibility for example.

In experiments, we used the deep network from Chapter IV, trained with assembly output codes, to create set-valued data out of MNIST. Given that each neuron on the output layer is associated with a combination of classes, we used it to derive belief functions over sets of 1, 2, and 5 classes. Applying the Dubois-Prade fusion method to estimate beliefs [133], we find that by combining inputs from the grandmother cells network, the couple cells network and the quintuplets cell network, it is possible to get significantly less classification errors than with single networks. The criterion of the maximum of plausibility gives the best result with only 58 misclassified images, as shown in Table 6.1.

Again, this first result seems promising, and more tests should be run on different datasets. A possible direction of research would be to use this method to calculate pseudo-labels recursively as described in the previous section.

Bibliography

[1] E. Guizzo, “How google’s self-driving car works," In IEEE Spectrum Online, vol. 18, no. 7, 2011, pp. 1132–1141.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks," In Advances in neural information processing systems, 2012, pp. 1097–1105.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, “Generative adversarial nets," In Advances in neural information processing systems, 2014, pp. 2672–2680.

[4] R. High, “The era of cognitive systems: An inside look at IBM Watson and how it works," IBM Corporation, 2012, Redbooks.

[5] F. Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, and L. Yang, “Where does AlphaGo go: From Church-Turing thesis to AlphaGo thesis and beyond," In IEEE/CAA Journal of Automatica Sinica, vol. 3, no. 2, 2016, pp. 113–120.

[6] N. Brown and T. Sandholm, “Libratus: the superhuman AI for no-limit poker," In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, August 2017.

[7] B. Goertzel, “Human-level artificial general intelligence and the possibility of a technological singularity: A reaction to Ray Kurzweil’s The Singularity Is Near, and Mc-

Dermott's critique of Kurzweil," In Artificial Intelligence, vol. 171, no. 18, 2007, pp. 1161–1173.

[8] http://www.lefigaro.fr/secteur/high-tech/2016/06/17/32001-20160617ARTFIG00317-yann-lecun-l-intelligence-artificielle-a-connu-des-succes-et-des-echecs.php

[9] V. C. Muller, and N. Bostrom, "Future progress in artificial intelligence: A survey of expert opinion," In Fundamental issues of artificial intelligence, 2016, pp. 555–572. Springer, Cham.

[10] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," In Nature, 1969.

[11] Y. LeCun *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," In Neural networks: the statistical mechanics perspective, 1995, pp.261–276.

[12] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," In Proceedings of the national academy of sciences, vol. 79, no. 8, 1982, pp. 2554–2558.

[13] I. Kanter, "Potts-glass models of neural networks," In Physical Review A, vol. 37, no. 7, 1988, p. 2739.

[14] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," In IEEE transactions on neural networks, vol. 22, no. 7, pp. 1087–1096.

[15] D.O. Hebb, "The organization of behavior," Wiley, New York, 1949.

[16] R. Sarpeshkar, "Analog versus digital: extrapolating from electronics to neurobiology," In Neural computation, vol. 10, no. 7, 1998, pp. 1601–1638.

[17] C. Berrou, O. Dufor, V. Gripon, and X. Jiang, “Information, noise, coding, modulation: What about the brain?," In Turbo Codes and Iterative Information Processing (ISTC), 2014 8th International Symposium on, IEEE, August 2014, pp. 167–172.

[18] Y. Mochizuki, and S. Shinomoto, “Analog and digital codes in the brain," Physical Review E, vol. 89, no. 2, 2014, p. 022705.

[19] M. P. Van Den Heuvel, and O. Sporns, “Network hubs in the human brain," In Trends in cognitive sciences, vol. 17, no. 12, 2013, pp. 683–696.

[20] K. Hwang, M. A. Bertolero, W. B. Liu, and M. D’Esposito, “The human thalamus is an integrative hub for functional brain networks," In Journal of Neuroscience, vol. 37, no. 23, 2017, pp. 5594–5607.

[21] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland, “A general framework for parallel distributed processing," In Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, 1986, pp. 45–76.

[22] C. Blakemore and G. F. Cooper, “Development of the brain depends on the visual environment,” In Nature, vol. 228, no. 5270, 1970, pp. 477–478.

[23] H. V. Hirsch and D. N. Spinelli, “Visual experience modifies distribution of horizontally and vertically oriented receptive fields in cats,” In Science, vol. 168, no. 3933, 1970, pp. 869–871.

[24] H. Moravec, “When will computer hardware match the human brain," In Journal of evolution and technology, 1998, vol. 1, no. 1, p. 10.

[25] H. Jarollahi *et al.*, “A nonvolatile associative memory-based context-driven search engine using 90 nm CMOS/MTJ-hybrid logic-in-memory architecture," In IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 4, no. 4, 2014, pp. 460-474.

[26] N.Onizawa and W.J. Gross. “Low-power area-efficient large-scale IP lookup engine based on binary-weighted clustered networks," In Proceedings of the 50th Annual Design Automation Conference (ACM), 2013, p. 55.

[27] L. Y. Liu *et al.*, “CAM-based VLSI architectures for dynamic Huffman coding," IEEE Trans. Consum. Electron., vol. 40, no. 3, 1994, pp. 282–289.

[28] M. E. Valle, “A class of sparsely connected autoassociative morphological memories for large color images," IEEE Transactions on Neural Networks, vol. 20, no. 6, 2009, pp. 1045–1050.

[29] E. Gooya, D. Pastor, and V. Gripon, “Automatic face recognition using SIFT and networks of tagged neural cliques," In Proceedings of Cognitive, 2015, pp.57–61.

[30] S. A. Shamma, “Speech processing in the auditory system II: Lateral inhibition and the central processing of speech evoked activity in the auditory nerve," The Journal of the Acoustical Society of America, vol. 78, no. 5, 1985, pp. 1622–1632.

[31] H. Ozeki *et al.*, “Inhibitory stabilization of the cortical network underlies visual surround suppression," Neuron, vol. 62, no. 4, 2009, pp. 578–592.

[32] P. Chossat, and M. Krupa, “Consecutive and non-consecutive heteroclinic cycles in Hopfield networks," Dynamical Systems, 2016, pp. 1–15.

[33] R. J. McEliece *et al*, “The capacity of the Hopfield associative memory," IEEE transactions on information theory, vol. 33, no. 4, 1987, pp. 461–482.

[34] M. Laiho *et al.*, “High-dimensional computing with sparse vectors," Biomedical Circuits and Systems Conference (BioCAS), 2015, pp. 1–4.

[35] H. C. Longuet-Higgins, D. J. Willshaw, and O. P. Buneman, “Theories of associative recall," Quarterly reviews of biophysics, 3(02), 1970, pp. 223-244.

[36] J. D. Keeler, “Comparison between Kanerva’s SDM and Hopfield-type neural networks," Cognitive Science, vol. 12, no. 3, 1988, pp. 299–329.

[37] D. Willshaw and P. Dayan, “Optimal plasticity from matrix memories: What goes up must come down," Neural Computation, vol. 2, no. 1, 1990, pp. 85–93.

[38] A. Knoblauch, G. Palm, and F. T. Sommer “Memory capacities for synaptic and structural plasticity," Neural Computation, vol. 22, no. 2, 2010, pp. 289–341.

[39] A. A. Mofrad *et al.*, “Clique-based neural associative memories with local coding and precoding," Neural Computation, 2016

[40] V. Gripon and C. Berrou, “A simple and efficient way to store many messages using neural cliques," Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on, 2011, pp. 1–5.

[41] ——, “Nearly-optimal associative memories based on distributed constant weight codes," Information Theory and Applications Workshop (ITA), 2012, pp. 269–273.

[42] B. K. Aliabadi, C. Berrou, and V. Gripon, “Storing sparse messages in networks of neural cliques," Proceedings of the national academy of sciences, vol. 25, no. 5, 2014, pp. 461–482.

[43] G. Palm, “Neural associative memories and sparse coding," Neural Networks, vol. 37, 1987, pp. 165–171.

[44] J. R. Cavanaugh, W. Bair, and J. A. Movshon, “Nature and interaction of signals from the receptive field center and surround in macaque v1 neurons," Journal of neurophysiology, vol. 88, no. 5, 2002, pp. 2530–2546.

[45] M. A. Heller and E. Gentaz, “Psychology of touch and blindness," Psychology Press, 2013.

[46] T. Kohonen, “The self-organizing map," Neurocomputing, vol. 21, no. 1, 1998, pp. 1–6.

[47] A. Aboudib, V. Gripon, and X. Jiang “A study of retrieval algorithms of sparse messages in networks of neural cliques," Proceedings of the Sixth International Conference on Advanced Cognitive Technologies and Applications, 2014, pp. 140–146.

[48] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, “Tree-based reparameterization framework for analysis of sum-product and related algorithms," IEEE Transactions on information theory, vol. 49, no. 5, 2003, pp. 1120-1146.

[49] D. H. Hubel, T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex," In The Journal of physiology, vol. 160, no. 1, 1962, pp. 106–154.

[50] D. H. Hubel, T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex," In The Journal of physiology, vol. 195, no.1, 1968, pp. 215–243.

[51] D. H. Kim-Dufor, P. Tigréat, C. Berrou, “Minimum information required for written word recognition," In Groupe de Linguistique Appliquée des Télécommunications (GLAT), June 2014, Brest, France. pp.84–91.

[52] D. H. Kim-Dufor, A. L. Olteanu, P. Tigréat, O. Dufor, C. Berrou, “Visual recognition of partially displayed words," to be submitted.

[53] G. M. Reicher, “Perceptual recognition as a function of meaningfulness of stimulus material," In Journal of experimental psychology, vol. 81, no. 2, 1969, pp. 275–280.

[54] D. D. Wheeler, “Processes in word recognition," In Cognitive Psychology, vol. 1, no. 1, 1970, pp. 59–85.

[55] Y. Hirai, “A template matching model for pattern recognition: Self-organization of templates and template matching by a disinhibitory neural network," In Biological cybernetics, vol. 38, no. 2, 1980, pp. 91–101.

[56] P. Gader *et al.* “Recognition of handwritten digits using template and model matching," In Pattern recognition, vol. 24, no. 5, 1991, pp. 421–431.

[57] Y. Pi, W. Liao, M. Liu, and J. Lu, “Theory of cognitive pattern recognition," In Pattern Recognition Techniques, Technology and Applications. InTech, 2008, pp. 433–463.

[58] J. L. McClelland, D. E. Rumelhart, “An interactive activation model of context effects in letter perception: I. An account of basic findings," In Psychological review, vol. 88, no. 5, 1981, p.375.

[59] K. Diependaele, J. C. Ziegler, J. Grainger, “Fast phonology and the bimodal interactive activation model," In European Journal of Cognitive Psychology, vol. 22, no. 5, 2010, pp. 764–778.

[60] W. J. Van Heuven, T. Dijkstra, J. Grainger, “Orthographic neighborhood effects in bilingual word recognition," In Journal of memory and language, vol. 39, no. 3, 1998, pp. 458–483.

[61] T. Dijkstra and W. J. Van Heuven, “The architecture of the bilingual word recognition system: From identification to decision," In Bilingualism: Language and cognition, vol. 5, no. 3, pp. 175–197.

[62] M. Minsky and S. Papert, “Perceptrons," 1969.

[63] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in posirion," In Biological Cybernetics, vol. 43, 1980, pp. 59–69.

[64] G. Cybenko, “Approximation by superpositions of a sigmoidal function," In Mathematics of Control, Signals, and Systems (MCSS), vol. 2, no. 4, 1989, pp. 303–314.

[65] M. D. Zeiler, R. Fergus, “Visualizing and understanding convolutional networks," In European conference on computer vision, Springer International Publishing, September 2014, pp. 818–833.

[66] D. Erhan, Y. Bengio, A. Courville, P. Vincent, “Visualizing higher-layer features of a deep network," University of Montreal, 1341, 3.

[67] A. Mahendran, A. Vedaldi, “Understanding deep image representations by inverting them," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5188–5196.

[68] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, “Understanding neural networks through deep visualization," 2015, arXiv preprint arXiv:1506.06579.

[69] C. F. Cadieu *et al.*, “Deep neural networks rival the representation of primate IT cortex for core visual object recognition," In PLoS Comput Biol, vol. 10, no. 12, 2014, e1003963.

[70] J. J. DiCarlo, D., Zoccolan, N. C. Rust, “How does the brain solve visual object recognition?," In Neuron, vol. 73, no. 3, 2012, pp. 415–434.

[71] S. Bridle, “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," In Neurocomputing, Springer Berlin Heidelberg, 1990, pp. 227–236.

[72] D. R. Cox, “The regression analysis of binary sequences," In Journal of the Royal Statistical Society. Series B (Methodological), 1958, pp. 215–242.

[73] M. N. Galtier and G. Wainrib, “A biological gradient descent for prediction through a combination of STDP and homeostatic plasticity," In Neural computation, vol. 25, no. 11, 2013, pp. 2815–2832.

[74] X. Xie and H. S. Seung, “Equivalence of backpropagation and contrastive Hebbian learning in a layered network," In Neural computation, vol. 15, no. 2, 2003, pp. 441–454.

[75] R. C. O’Reilly, D. Wyatte, S. Herd, B. Mingus, and D. J. Jilk, “Recurrent processing during object recognition Supporting Information," ISO 690.

[76] D. L. Yamins *et al*, “Performance-optimized hierarchical models predict neural responses in higher visual cortex," In Proceedings of the National Academy of Sciences, vol. 111, no. 23, 2014, pp. 8619–8624.

[77] Y. Bengio, “Learning deep architectures for AI," Foundations and trends® in Machine Learning, vol. 2, no. 1, 2009, pp. 1–127.

[78] S. M. Khaligh-Razavi and N. Kriegeskorte, “Deep supervised, but not unsupervised, models may explain IT cortical representation," In PLoS Computational Biology, vol. 10, no. 11, 2014, e1003915.

[79] T. Hannagan, A. Amedi, L. Cohen, G. Dehaene-Lambertz, and S. Dehaene, “Origins of the specialization for letters and numbers in ventral occipitotemporal cortex," In Trends in cognitive sciences, vol. 19, no. 7, 2015, pp. 374–382.

[80] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines," In Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[81] G. E. Hinton *et al.*, “Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580, 2012.

[82] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, 2014, pp. 1929–1958.

[83] P. A. Devijver and J. Kittler, “Pattern recognition: A statistical approach," Vol. 761, London: Prentice-Hall, 1982.

[84] F. Bastien *et al*, “Theano: new features and speed improvements," arXiv preprint arXiv: 1211.5590, 2012.

[85] F. Chollet, “Keras: Theano-based deep learning library," Code: https://github.com/fchollet. Documentation: http://keras.io., 2015.

[86] C. E. Shannon and W. Weaver, “The mathematical theory of communication," Urbana: University of Illinois Press, 1949.

[87] G. Brown, A. Pocock, M. J. Zhao and M. Lujan, “Conditional likelihood maximisation: a unifying framework for information theoretic feature selection," In Journal of Machine Learning Research, vol. 13, January 2012, pp. 27–66.

[88] B. New, M. Brysbaert, J. Veronis, and C. Pallier, “The use of film subtitles to estimate word frequencies," In Applied psycholinguistics, vol. 28, no. 4, 2007, pp. 661–677.

[89] L. V. D. Maaten and G. E. Hinton, “Visualizing data using t-SNE," In Journal of Machine Learning Research, vol. 9, November 2008, pp. 2579–2605.

[90] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading text in the wild with Convolutional Neural Networks," International Journal of Computer Vision, vol. 116, no. 1, 2016, pp. 1–20.

[91] M. R. Marques, X. Jiang, O. Dufor, C. Berrou, and D. H. Kim-Dufor “A connectionist model of reading with error correction properties," In LTC 2015: 7th Language and Technology Conference, November 2015, pp. 455–460.

[92] C. Szegedy *et al* “Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.

[93] F. Chollet, https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html, 2016.

[94] R. Salakhutdinov, J.B. Tenenbaum, and A. Torralba, “One-Shot learning with a hierarchical nonparametric Bayesian model," In ICML Unsupervised and Transfer Learning, 2012, pp. 195–206.

[95] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, “Review of classifier combination methods," In Machine Learning in Document Analysis and Recognition, Springer Berlin Heidelberg, 2008, pp.361–386.

[96] H. C. Kim and Z. Ghahramani, “Bayesian classifier combination," In Artificial Intelligence and Statistics, March 2008, pp. 619–627.

[97] E. Simpson, S. J. Roberts, I. Psorakis, and A. Smith, “Dynamic Bayesian combination of multiple imperfect classifiers," Decision making and imperfection, vol. 474, 2013, pp. 1–35.

[98] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana, “On the effect of calibration in classifier combination," In Applied intelligence, vol. 38, no. 4, 2013, pp. 566–585.

[99] M. A. Bagheri, G. A. Montazer, and S. Escalera, “Error correcting output codes for multiclass classification: application to two image vision problems," In Artificial Intelligence and Signal Processing (AISP), 16th CSI International Symposium on, IEEE, 2012, pp. 508-513.

[100] S. Bridle, “Probabilistic interpretation of feed-forward classification network outputs, with relationships to statistical pattern recognition," In Neurocomputing, Springer Berlin Heidelberg, 1990, pp.227–236.

[101] J. I. Arribas and J. Cid-Sueiro, “A model selection algorithm for a posteriori probability estimation with neural networks," In IEEE Transactions on Neural Networks, vol. 16, no. 4, July 2005, pp. 799–809.

[102] Y. LeCun, C. Cortes, and C. J. Burges, “The MNIST database of handwritten digits," 1998.

[103] M. Pezeshki, L. Fan, P. Brakel, A. Courville, and Y. Bengio, “Deconstructing the ladder network architecture," In International Conference on Machine Learning, June 2016, pp. 2368–2376.

[104] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning," In NIPS workshop on deep learning and unsupervised feature learning, Vol. 2011, Granada, Spain, p.4, 2011.

[105] S. Ioffe, and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv: 1502.03167., 2015.

[106] D. Kingma and J. Ba, “Adam: A method for stochastic optimization," arXiv preprint arXiv: 1412.6980, 2014.

[107] F. Bastien *et al.*, “Theano: new features and speed improvements, " arXiv preprint arXiv: 1211.5590,2012.

[108] F. Chollet, “Keras: Theano-based deep learning library," Code: https://github.com/fchollet. Documentation: http://keras.io., 2015.

[109] R. Graham and N. Sloane, “Lower bounds for constant weight codes," In IEEE Transactions on Information Theory, vol. 26, no. 1, 1980, pp. 37–43.

[110] K. Zilles, N. Palomero-Gallagher, and K. Amunts, “Development of cortical folding during evolution and ontogeny," In Trends in neurosciences, vol. 36, no. 5, 2013, pp.275–284.

[111] R. W. Hamming, “Error detecting and error correcting codes," In Bell Labs Technical Journal, vol. 29, no. 2, 1950, pp. 147–160.

[112] R. Pyndiah, A. Glavieux, A. Picart and S. Jacq, “Near optimum decoding of product codes," In Global Telecommunications Conference, GLOBECOM’94. Communications: The Global Bridge., IEEE, November 1994, pp. 339–343.

[113] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading text in the wild with convolutional neural networks," International Journal of Computer Vision, vol. 116, no. 1, 2016, pp. 1–20.

[114] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks," In AIstats, vol. 9, May 2010, pp. 249–256.

[115] B. Larras, C.Lahuec, M. Arzel, and F. Seguin, “Analog implementation of encoded neural networks,” In Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, 2013, pp. 1612-1615.

[116] Z. Yao, V. Gripon, and M. Rabbat, “A GPU-based associative memory using sparse Neural Networks,” In High Performance Computing and Simulation (HPCS), 2014 International Conference on, 2014, pp. 688–692.

[117] Z. Yao, V. Gripon, and M. Rabbat, “A massively parallel associative memory based on sparse neural networks,” arXiv preprint arXiv:1303.7032., 2013

[118] H. Poon and P. Domingos, “Sum-product networks: A new deep architecture," In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, November 2011, pp. 689–690.

[119] O. Delalleau and Y. Bengio, “Shallow vs. deep sum-product networks," In Advances in Neural Information Processing Systems, pp. 666–674.

[120] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain," In Psychological review, vol. 65, no. 6, 1958, p. 386.

[121] J. Bergstra *et al.*, “Theano: Deep learning on GPUs with python," In NIPS 2011, BigLearning Workshop, Granada, Spain, vol. 3.

[122] P. Vincent *et al.*, “Extracting and composing robust features with denoising autoencoders," In Proceedings of the 25th international conference on Machine learning, ACM, July 2008, pp. 1096–1103.

[123] R. K. Srivastava *et al.*, “Compete to compute," In Advances in neural information processing systems, 2013, pp. 2310–2318.

[124] A. Makhzani and B. Frey, “K-sparse autoencoders," arXiv preprint arXiv:1312.5663, 2013.

[125] S. Grossberg, and D. Levine, “Some developmental and attentional biases in the contrast enhancement and short term memory of recurrent neural networks," In Journal of Theoretical Biology, vol. 53, no. 2, 1975, pp. 341–380.

[126] Y. Sakurai, “How do cell assemblies encode information in the brain?," In Neuroscience & Biobehavioral Reviews, vol. 23, no. 6, 1999, pp.785–796.

[127] M.H. Histed, V. Bonin, and R.C. Reid, “Direct activation of sparse, distributed populations of cortical neurons by electrical microstimulation," In Neuron, vol. 63, no. 4, 2009, pp. 508–522.

[128] L. Chang and D.Y. Tsao, “The code for facial identity in the primate brain," In Cell, vol. 169, no. 6, 2017, pp. 1013–1028.

[129] A. Rasmus *et al.*, “Semi-supervised learning with ladder networks," In Advances in Neural Information Processing Systems, 2015, pp. 3546–3554.

[130] T. Salimans *et al.*, “Improved techniques for training GANs," In Advances in Neural Information Processing Systems, 2016, pp. 2234–2242.

[131] D.H. Lee, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," In Workshop on Challenges in Representation Learning, ICML, vol. 3, 2013, p. 2.

[132] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design," In Test Symposium (ETS), 2013 18th IEEE European, IEEE, May 2013, pp. 1–6.

[133] D. Dubois, and H. Prade, “On the use of aggregation operations in information fusion processes," In Fuzzy sets and systems, vol. 142, no. 1, 2004, pp. 143–161.

Résumé

L'objectif de la recherche en Intelligence Artificielle (IA) est de répliquer les capacités cognitives humaines au moyen des ordinateurs modernes. Les résultats de ces dernières années semblent annoncer une révolution technologique qui pourrait changer profondément la société.

Nous focalisons notre intérêt sur deux aspects cognitifs fondamentaux, l'apprentissage et la mémoire.

Les mémoires associatives offrent la possibilité de stocker des éléments d'information et de les récupérer à partir d'une partie de leur contenu, et imitent ainsi la mémoire cérébrale.

L'apprentissage profond permet de passer d'une perception analogique du monde extérieur à une représentation parcimonieuse et plus compacte.

Dans le chapitre 2, nous présentons une mémoire associative inspirée des réseaux de Willshaw, avec une connectivité contrainte. Cela augmente la performance de récupération des messages et l'efficacité du stockage de l'information.

Dans le chapitre 3, une architecture convolutive a été appliquée sur une tâche de lecture de mots partiellement affichés dans des conditions similaires à une étude de psychologie sur des sujets humains. Cette expérimentation montre la similarité de comportement du réseau avec les sujets humains concernant différentes caractéristiques de l'affichage des mots.

Le chapitre 4 introduit une méthode de représentation des catégories par des assemblées de neurones dans les réseaux profonds. Pour les problèmes à grand nombre de classes, cela permet de réduire significativement les dimensions d'un réseau.

Le chapitre 5 décrit une méthode d'interfaçage des réseaux de neurones profonds non supervisés avec les mémoires associatives à cliques.

Mots-clés : Apprentissage machine, Réseaux de neurones, Codage, Parcimonie, Robustesse

Abstract

The objective of research in Artificial Intelligence (AI) is to reproduce human cognitive abilities by means of modern computers. The results of the last few years seem to announce a technological revolution that could profoundly change society.

We focus our interest on two fundamental cognitive aspects, learning and memory.

Associative memories offer the possibility to store information elements and to retrieve them using a sub-part of their content, thus mimicking human memory.

Deep Learning allows to transition from an analog perception of the outside world to a sparse and more compact representation.

In Chapter 2, we present a neural associative memory model inspired by Willshaw networks, with constrained connectivity. This brings an performance improvement in message retrieval and a more efficient storage of information.

In Chapter 3, a convolutional architecture was applied on a task of reading partially displayed words under similar conditions as in a former psychology study on human subjects.This experiment put in evidence the similarities in behavior of the network with the human subjects regarding various properties of the display of words.

Chapter 4 introduces a new method for representing categories using neuron assemblies in deep networks. For problems with a large number of classes, this allows to reduce significantly the dimensions of a network.

Chapter 5 describes a method for interfacing deep unsupervised networks with clique-based associative memories.

Keywords: Artificial intelligence, Neural networks, Deep learning, Machine learning, Associative memory, Classification, Parallel programming, Error-correcting codes

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

N° d'ordre : 2017IMTA0046

IMT Atlantique Bretagne-Pays de la Loire - www.imt-atlantique.fr

Campus de Brest	Campus de Nantes	Campus de Rennes
Technopôle Brest-Iroise	4, rue Alfred Kastler - La Chantrerie	2, rue de la Châtaigneraie
CS 83818	CS 20722	CS 17607
29238 Brest Cedex 03	44307 Nantes Cedex 3	35576 Cesson Sévigné Cedex
T +33 (0)2 29 00 11 11	T +33 (0)2 51 85 81 00	T +33 (0)2 99 12 70 00
F +33 (0)2 29 00 10 00	F +33 (0)2 51 85 81 99	F +33 (0)2 99 12 70 08