Efficient Representations for Graph and Neural Network Signals

Vincent Gripon

Manuscript to defend

Habilitation à Diriger des Recherches (Habilitation)

of

Vincent GRIPON

Efficient Representations for Graph and Neural Network Signals

Defended at ENS Lyon on Dec. 4th, 2020, in front of the jury, composed of:

President:	Rémi Gribonval, Senior Research Scientist at INRIA
Reviewers:	Sophie Achard, Senior Research Scientist at CNRS
	Pascal Frossard, Professor at EPFL
	Julie Grollier, Senior Research Scientist at CNRS/Thales
Examiners:	Sébastien Lefèvre, Professor at Université Bretagne Sud
	Antonio Ortega, Professor at University of Southern California

Contents

Chapter I

Curriculum Vitae

I.1 Summary

Personal Information	
First name:	Vincent
Last name:	Gripon
Birth date:	April 3rd, 1985
Nationality:	French
ORCID ID:	0000-0002-4353-4542
Website:	`https://www.vincent-gripon.com`
Position and Research Subjects	
Position:	Permanent Researcher with IMT Atlantique
Keywords:	Efficient Deep Learning, Associative Memories, Graph Signal Processing, FPGA Implementation
Quantitative Summary	
Number of citations:	$\approx$ 1200 according to Google Scholar
H-index:	$\approx$ 19 according to Google Scholar
Books and chapters:	2
Papers in journals:	17
Other papers:	59 (including 2 best papers)
Obtained funding:	$\approx$1M€

I.2 Education and Past Positions

Date	Position
2020	Chair of Excellence at Université de Côte d'Azur. Host: Benoît Miramond
2018-2019	Invited Professor at Université de Montréal and Mila. Host: Yoshua Bengio
2012-	Permanent Researcher at IMT Atlantique
2011-2012	Postdoc at McGill University. Supervisers: Michael Rabbat and Warren Gross
2008-2011	PhD student at IMT Atlantique. Superviser: Claude Berrou
2006-2008	Master at École Normale Supérieure Paris-Saclay. Specialty: Computer Science and Telecommunications

I.3 PhD Thesis Supervisions

2011-2014	Bartosz Bogulawski, supervising at 25% Subject: Dynamic Power Management of MPSoC using Networks of Neural Cliques Now: Permanent Researcher with Schneider Electric Note: Joint PhD with CEA LETI
2013-2016	Alan Aboudib, supervising at 50% Subject: Neuro-inspired Architectures for the Acquisition Processing of Visual Information Now: Head of AI with DiGEiZ
2013-2017	Philippe Tigréat, supervising at 25% Subject: Sparsity, Redundancy and Robustness in Artificial Neural Networks for Learning and Memory

2013-2018	Robin Danilo, supervising at 25% Subject: Approches connexionnistes pour la vision par ordinateur embarquée Now: Teacher Note: Joint PhD with University of South Brittany
2015-2018	Bastien Pasdeloup, supervising at 50% Subject: Extending Convolutional Neural Networks to Irregular Domains Through Graph Inference Now: Professor with IMT Atlantique
2015-2018	Jean-Charles Vialatte, supervising at 50% Subject: On Convolution of Graph Signals And Deep Learning on Graph Domains Now: Permanent Researcher with Senx Note: Joint PhD with Senx
2016-2019	Ghouthi Boukli Hacene, supervising at 33% Subject: Processing and Learning Deep Neural Network on Chip Rewards: Ghouthi obtained the prize for the best PhD in 2019 from Institut Mines Telecom, as well as the prize for the best PhD 2019 from the French Association for Artificial Intelligence (AFIA). Now: Postdoc at Mila
2017-	Carlos Rosar Kos Lassance, supervising at 50% Subject: Graph Methods for Visualization and Robustness of Deep Learning Architectures
2018-	Myriam Bontonou, supervising at 50% Subject: Processing of Time-Varying Graph Signals
2019-	Yuking Hu, supervising at 50% Subject: Deep Learning in Few-Shot Settings Note: Joint PhD with Orange
2019-	Hugo Tessier, supervising at 50% Subject: Compression Methods for Deep Learning Architectures Note: Joint PhD with PSA

I.4 Other Supervisions

I.4.1 Postdocs and Research Engineers

2015-2016	Nicolas Farrugia, Postdoc, supervising at 100% Project: Neural Communications, with LabEX CominLabs Subject: Tracking the Dynamics of Cognitive Ability Now: Professor with IMT Atlantique
2016-2017	Mathilde Ménoret, Postdoc, supervising at 50% Project: Neural Communications, with LabEX CominLabs Subject: Graph Signal Processing for fMRI Classification Now: Research Engineer with IMT Atlantique
2017-2018	Olivier Dufor, Postdoc, supervising at 25% Subject: Data Acquisition for EEG measures during Sound Recognition Now: Professor with ISEN
2017-2018	Ehsan Sedgh Gooya, Research Engineer, supervising at 50% Subject: Demonstrators for Compressed Deep Learning Architectures Now: Professor with ISEN
2020-	Mounia Hamidouche, Postdoc, supervising at 33% Subject: Using graph signal processing to improve few-shot classifiers

I.4.2 Master Students and Interns

2013	Baptiste Tessiau, from ENS Rennes Subject: Associative Memories for Relational Algebra Thibault Ehret, from ENS Rennes Subject: Sparse Associative Memories and Dictionary Learning Jean-Charles Vialatte, from IMT Atlantique Subject: Classification with Networks of Neural Cliques

	Alan Aboudib, from ENIB Subject: Comparison of Sparse Associative Memories
2014	Chendi Yu, from IMT Atlantique Subject: Associative Memories for Nearest Neighbor Search
	Baptiste Bourès, from University of Brest Subject: Developing a Web-based Contest Platform
	Nissim Zerbib, from ENS Paris Subject: Extended Principles of Associative Memories
	Réda Alami, from IMT Atlantique Subject: Uncertainty with Graph Signal Processing
2015	Demetrio Ferro, from Polytechnico di Turino Subject: Accelerating Search with Associative Memories
	Jules Pondard, from ENS Paris Subject: Hypergraphs for Associative Memories
	Martin Guy, from ENS Lyon Subject: Time-Supervised Classification of Video Frames
	Emma Kerinec, from ENS Lyon Subject: Visualisation of Time-Varying Graph Signals
2016	Tristan Sterin, from ENS Lyon Subject: Comparison of LSTMs and Vanilla RNNs
	Ghouthi Boukli Hacene, from Polytechnique Alger Subject: Search with Associative Memories
2017	Guillaume Duboc, from ENS Lyon Subject: Automatic Discovery of Virtual Environments
	Simon Fernandez, from ENS Lyon Subject: Semantics of Recurrent Neural Networks
	Justine Delomenie, from IMT Atlantique Subject: Automatic Music Video Clip Generation
	Houda Abichou, from IMT Atlantique Subject: Graph Signal Processing and Distributed Optimization
	Rolland Xavier, from ENS Rennes Subject: Analysing Trajectories of Heat Diffusion on Graphs

2018	Amal Ben Soussia, from ENIT Subject: Robustness of Deep Neural Network Architectures Mehdi Rezzoug, from IMT Atlantique Subject: Universality of Convolutional Neural Networks Nicolas Grelier, from IMT Atlantique Subject: Finding Regular Visualizations of Graphs
2020	Guillaume Coiffier, from ENS Lyon Subject: Deep learning architectures with a tiny parameter budget Louis Béthune, from ENS Lyon Subject: Predicting the accuracy of few-shot classifiers Théo Giraudon, from ENPC Subject: Towards a robust definition of the robustness of classifiers Tom Pégeot, from IMT Atlantique Subject: Relation between Pruning and Robustness in Neural Network Classifiers

I.5 Funded Projects

In this section we start with a summary of funded projects, then we present some highlights in a few paragraphs. Note that the results of these projects are also discussed in Chapter II. Privately funded projects are not discussed because of intellectual property.

I.5.1 Summary of Funded Projects

Date	Amount	Funding institution and subject
2012	6k€	Lab-STICC Subject: Large Scale Simulation of Associative Memories Role: PI
2015	110k€	Futur et Ruptures Subject: Emergence of Error Correcting Coding in Neural Networks Role: Partner

2015	291k€	LabEX CominLabs Subject: Neural Communications Role: PI
2017	110k€	PRACom and Brest Metropole Subject: Efficient Implementations of Deep Learning Role: PI
2017	5k€	CNRS GDRs and IMT Atlantique Subject: NeuroSTIC in Brest Role: Organizer
2018	30k€	Orange Labs Subject: Online Sound Recognition Role: PI
2018	150k€	Thales Subject: Associative Memories for Classification and Novelty detection Role: PI
2018	50k€	Région Bretagne Subject: Graph Signal Processing for Deep Learning Visualization and Robustness Role: PI
2018	55k€	MOOC Funding Subject: Advanced Algorithmics and Graph Theory with Python Role: PI
2019	150k€	Thales Subject: Few-Shot Learning Role: PI
2019	10k AUD	FASIC Subject: Enhancing Indirect visual Localization with Graphs Role: PI

2019	7k€	GDR ISIS Subject: Adversarial Robustness and Training biases Role: PI
2019	20k€	ERE DGA Subject: Incremental Learning in Complex Situations Role: PI
2020	50k€	TSN Carnot Subject: Graph Signal Processing to represent inner layers of Deep Neural Networks Role: PI
2020	30k€	Orange Labs (CIFRE) Subject: Few-shot Learning Role: PI
2020-2022	75k€	PSA (CIFRE) Subject: Compression of Deep Neural Networks Role: PI
2020-2021	15k€	Schneider Subject: Few-shot and continual learning Role: PI
2020-2022	202k€	Brittany region Subject: Optimizing observation, detection, classification and tracking of maritime objects Total funding: 1.1M€ Role: PI
2021-2023	10k€	AXA (CIFRE) Subject: Explanability and Interpretability in Deep Neural Networks Role: PI

I.5.2 Neural Communications

The project “Neural Communications” was funded by the LabEX CominLabs for a duration of two years. It was intended to be a continuation of the LabEX CominLabs project “Neural Coding”, led by Professor Claude Berrou. The project was built upon

a collaboration between three main entities: IMT Atlantique, the LTSI Laboratory of INSERM Rennes, and Orange Labs. In total, 291k€ were funded, covering roughly the expanses related to the recruitment of three postdocs/research engineers for about two years.

The main motivation of the project was to investigate neural communications at the millisecond scale, thanks to a twofold approach. On the one hand, wecreated a theoretical model of neural communications [1]. On the other hand, experiments were conducted so as to reinforce or contradict the theoretical developments [2, 3, 4].

The project led to the development of signal processing methods to track the cognitive activity at the millisecond scale. In Figure I.1 are shown illustrations extracted from the project report, where spatio-temporal dynamic patterns of spontaneous brain connectivity have been automatically determined.

The project also introduced novel methods to better represent brain activity for downstream tasks, such as classification [5]. Extracted from this article, Table I.7 shows how graphs can advantageously benefit such tasks, when using the low or high frequencies (LF or HF) in conjunction with graph sampling (GS) or graph Fourier transform (GFT).

Table I.7: Accuracy of the classification on the Haxby dataset (face vs. houses) for several graph-based sampling methods (in %) for two difficulty groups. Starred numbers indicate best scores in their category.

Graph Types	GFT LF	GFT HF	GFT ANOVA	GS LF	GS HF
Difficult					
Full	54.8%	51.1%	**66.0%**	52.0%	51.3%
Geometric	56.7%	64.8%	64.8%	50.5%	65.2%
|Correlation|	52.4%	66.8%	64.7%	50.9%	60.3%
|Covariance|	52.4%	67.6%	65.2%	51.2%	66.2%
Kalofolias	**61.6%**	51.9%	65.9%	**61.6%**	51.9%
Semilocal	53.8%	**69.5%**	65.6%	50.3%	**72.5%***
Fundis	54.9%	64.2%	65.1%	49.7%	62.8%
Easy					
Full	65.1%	60.0%	88.9%	49.6%	60.6%
Geometric	71.3%	79.4%	86.0%	57.8%	79.1%
|Correlation|	59.5%	86.5%	86.9%	53.5%	75.2%
|Covariance|	57.2%	87.0%	88.8%	52.8%	84.8%
Kalofolias	**88.2%**	54.2%	**89.4%**	**87.5%**	52.6%
Semilocal	61.3%	**87.9%**	88.6%	52.3%	**90.9%***
Fundis	67.4%	77.5%	86.7%	52.4%	77.5%

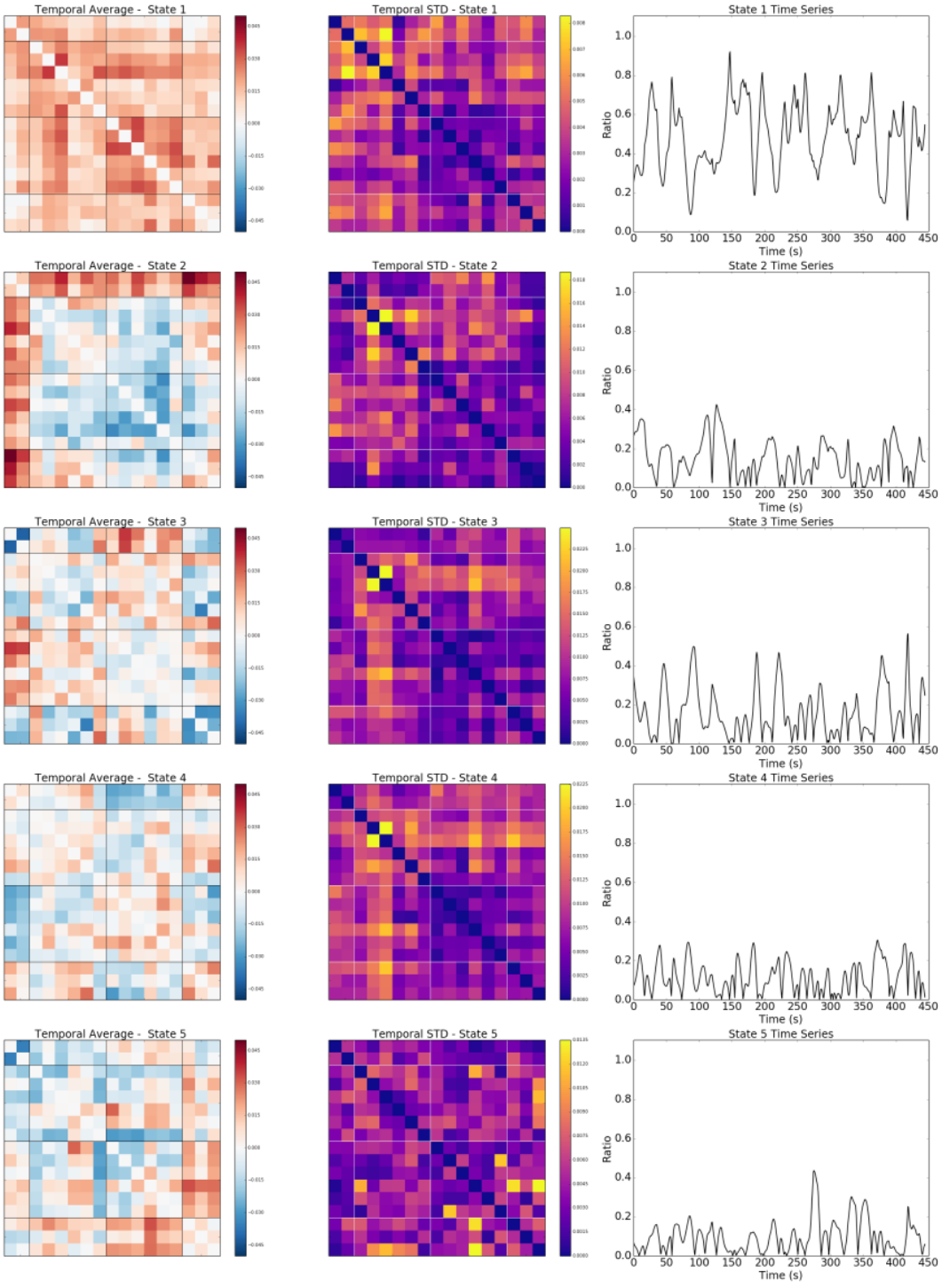


Figure I.1: Spatio-temporal patterns of spontaneous brain activity during resting state, along with their time-course prominence.

I.5.3 Advanced Algorithmics and Graph Theory with Python MOOC

This project was funded by IMT. The funding was used partly to cover the costs of the service providers, and partly to cover the commitment of the staff. We developed a MOOC addressed at students familiar with programming, but looking for a way to improve their skills, while learning basics of graph theory. The whole project revolves around a game where two players compete to grab pieces of cheese in a maze as quickly as possible. The MOOC covers graph traversal, the traveling salesman problem, greedy approaches and operational research. One year after its launch in November 2018, the MOOC had more than 10'000 students, of which more than 500 finished all the exercises. The students come from all over the world, with more than 50 countries represented. This MOOC was built upon the PyRat course I created in 2010 (see Section I.6).

The MOOC benefited of a professional team for creating video clips and illustrations. An extracted screenshot is in Figure I.2.

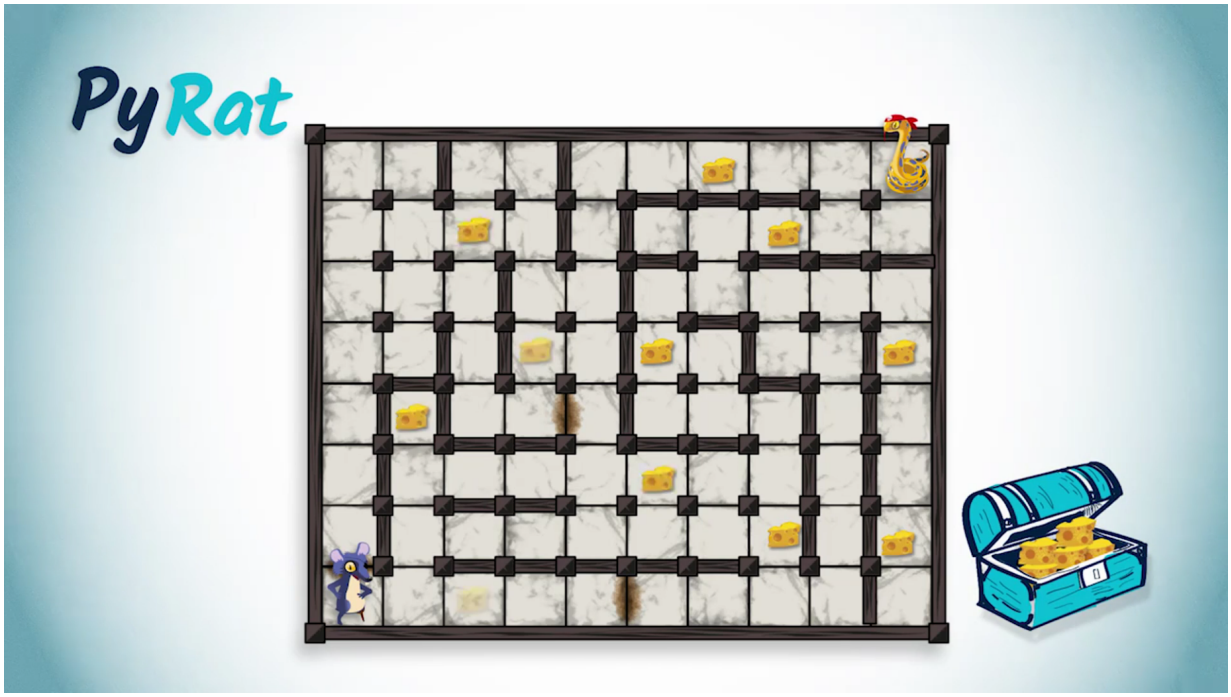


Figure I.2: Screenshot from one of the videos of the MOOC. It displays the maze game the students play with during the lessons.

At the end of the course, students are supposed to be able to:

- Express a computational problem (such as pathfinding) using graph theory,
- Choose the appropriate algorithm to solve the given computational problem,

- Code the algorithmic solution in python,
- Evaluate the proposed solution in terms of its complexity (amount of resources, scalability) or performance (accuracy, latency).

I.5.4 Incremental Learning of Complex Situations

During my sabbatical at Université de Montréal, the main purpose was to study the possibility to incrementally learn in complex situations. We thus investigated the question of learning from streaming data in a very constrained settings where both memory and computational complexities are limited [6]. We also considered the case of heterogeneous inputs supported by graphs [7, 8].

I.5.5 Adversarial Robustness and Training Biases

Since the end of 2019, we are investigating the impact of the training sampling policy on the robustness of trained architectures towards deviations of their inputs. The purpose is to bring together a pluridisciplinary team of scientists to address this problem: myself, Franck Vermet, a specialist in applied mathematics at University of Brest, and Matthias Löwe, a mathematician at Münster University in Germany. The funding is mostly used to cover an intern, Théo Giraudon.

I.5.6 Enhancing Indirect Visual Localization with Graph Filters

This project is funded by the FASIC program of the Australian Academy of Science. This is a joint project between Lab-STICC and the Australian Institute for Machine Learning at the University of Adelaide. The funding is mainly used to fund missions between France and Australia. The motivation of the project is to propose graph methods to denoise latent feature representations of images taken with a dash cam in a car. These representations are used to perform visual localization using a support dataset. A publication is in the process of being written on the subject.

I.5.7 Small and One-Person Fundings

I.5.7.1 Large Scale Simulation of Associative Memories

This project was funded by the CNRS Laboratory Lab-STICC in 2012, for a total amount of 6k€. The funding was primarily used to fund a desktop computer with advanced GPU components. This machine was used to perform simulations related to several publications, including [9, 10].

Another important part of the project was related to the acquisition of data related to source localization in binaural setting. In partnership with Orange laboratories in Lannion, this part of the project was devoted to testing the methods previously developed on visual tasks (in the previous project "Neural Coding"). Indeed, if resting state analysis offers the best substrate for brain connectivity analysis, we wanted to develop methodological tools and processing pipelines for task-related signals.

A startup project emerged from this project (Neurokyma), and is now technically led by Mahmoud Hassan, postdoc of the project, in Rennes. This startup is interested in providing toolboxes of signal processing for neuroimaging data.

I.5.7.2 Emergence of Error Correcting Coding in Neural Networks

This project was funded by the IMT program "Futur et Ruptures" in 2015, for a duration of three years, corresponding to the funding of a PhD student. The project focused on the plausibility of the emergence of coding techniques in the storage process of mental information, from a theoretical point of view. It consisted of two main parts. In a first part, we investigated the impact of synaptic noise on the performance of storage in neural networks [11, 12], which included introducing mathematical models of synaptic noise [13]. In a second part, we looked at the emergence of error correcting coding principles using simple hypothesis about the dynamic of neural networks [14].

I.5.7.3 Efficient Implementations of Deep Learning

This project was funded in part by PRACom ("Pole de Recherche Avancée en Communications") and in part by Brest Metropole. Most of the funding was used to recruit a PhD student. The motivation of the project was to develop techniques to allow efficient deep learning on chip. The project can be split into two parts. In a first part, the effort of research focused on proposing lightweight implementations of deep learning methods for the inference (not for learning). In a second part, transfer incremental learning approaches were proposed and implemented, using lightweight implementations of deep learning methods and novel tools for transfer learning. This project led to several publications [6, 15, 16, 16, 17].

I.5.7.4 Graph Signal Processing for Deep Learning Visualization and Robustness

This project is funded by Région Bretagne. Most of the funding is used to cover a PhD student. The idea of the project is to develop fine grain visualization techniques that allow to track the evolution of training at the scale of a layer in deep neural networks. Indeed, by looking at how the topology evolves between training examples, it is pos-

sible to detect overfitting, to detect convergence of training, or even to detect robustness defects [18]. Next, the project aims at proposing novel methods to improve the robustness of deep neural network architectures, by using graph representations of each layer [19, 20].

I.5.7.5 Organization of NeuroSTIC 2017 in Brest

Since 2016, I coorganize the NeuroSTIC days. The aim of NeuroSTIC is to bring together scientists in various fields connected with learning: cognitive scientists, computer scientists, electrical scientists, neuroscientists, psychologists... These french-speaking days have been organized in Grenoble (2016), Brest (2017) and Nice (2019). They usually attract between 100 and 200 scientists. In 2016, part of the funding was obtain at different GDRs of CNRS, and part of the funding at IMT Atlantique.

I.6 Teaching Experience

Being a permanent researcher, I have no obligation to teach in my institution. However I always considered teaching as a rewarding and enthusiastic experience, allowing to transfer knowledge to students, to connect them with research and to challenge myself in acquiring a deeper understanding of some domains. Beside these points, I feel it is of prime importance to democratize science and its methodology, in a context where dogmatism is strong and threatening. In addition to teaching, I regularly give popularization conferences, as a mean to promote the scientific method.

In the next paragraphs, I summarize my teaching experience.

Date	Total hours	Institution and subject
2007	12h	University of Rennes I TP C2I (teaching assistant)
2008	12h	IMT Atlantique Project “Introduction to Large Scale Systems”. I was responsible of the computer science part of the project including proposing and preparing teaching material, teaching and evaluating
2008-2009	31h	Lycée Kérichen Interrogation in Mathematics (MPSI)

2010	16h	IMT Atlantique Courses in Graph Theory and Programming with OCaml. I had full responsability of this one time course addressed at PhD students and Professors
2009-2011	130h	ENIB Course in “Operational Research”. I had full responsability of creating, teaching, and evaluating a course on operational research addressed at Licence 3 level students
2010-2011	65h	ENIB Course in “Linear Algebra”. I had full responsability of creating, teaching, and evaluating a course on operational research addressed at Licence 3 level students
2013	10h	IMT Atlantique Interrogations in Computer Science
2013	8h	IMT Atlantique Summer School on “Mental Information” addressed at teachers
2013-	14-40h each year	IMT Atlantique In charge of a course in the “research” excellence track for top students at IMT Atlantique. The principle of the course is to guide groups of two students towards a great understanding of a research subject, enough to discuss the use of a method in two distinct application areas or to discuss two possible methods to solve a given problem
2013-2015	≈100h each year	Lycée Kérichen Interrogation in Mathematics (MPSI and MP*)
2017	≈ 3h	BIOCOMP Summer School Course on associative memories
2014-2018	≈200h then 21h/year	IMT Atlantique PyRat course (see subsection below)

2018-	100h then ≈100h/year	IMT Atlantique We adapted PyRat to a MOOC. Now I have the charge of taking care of students on the online platform and updating the content based on the comments we receive
2020	2h	Université Côte d'Azur Introductory course to machine learning

I.6.1 PyRat Course

The PyRat course is by far the most evolved course I have created, since it has been through three major versions since its creation in 2010. Originally, it was a course designed to be oriented towards "operational research". It was addressed to Licence students, newcomers in the ENIB engineering school, with various backgrounds. As such, the course required to be more or less self-contained, and appealing to a very diverse audience. I had the idea of creating the course because the initial material I was given was very little appealing to students.

So, with the support and blind trust of the professor that put me in charge[1], I created a simple video game in which two players compete to find gold coins spread in a maze as quickly as possible. The game served as an excuse to introduce complex operational research algorithms (e.g. backtracking, branch and bound) with a very concrete application case. The course was a great success but only lasted for one year. Indeed, in 2011 I defended my PhD and moved to Canada for a postdoc.

In 2014, I was back at IMT Atlantique. Philippe Picouet, in charge of computer science in the school, heard of this course and proposed me to adapt it to the audience at IMT Atlantique. With the help of several colleagues, and a deeply involved PhD student (Bastien Pasdeloup), the course was transformed in a much more mature version. I coded everything from scratch to offer a much more straight-forward approach to students. For three years the course was played. In 2017, several adjustments had to be made to update the course to the merged school (in 2018, older institutions Télécom Bretagne and Mines de Nantes merged into IMT Atlantique).

In 2018, the project began of adapting the course to become a MOOC. Once again, everything had to be recoded from scratch so as to make the experience even simpler for the student. With the help of Anja Hopma of IMT Atlantique, a lot of changes have been made to the course. As of 2019, this course is one of the achievements of my career that I am the most proud of. It required countless hours of work, but the result

[1] This professor is Pascal Redou, and I deeply thank him for his trust and the opportunity he gave me back then.

is definitely very rewarding. In Figure I.3, you can see graphics of the three versions of the game. In Figure I.4, there is a picture advertising for a student party, having the game as a theme, and showing the adoption of the game by the students. A recording of the ending competition in 2015 is available online: `https://www.youtube.com/watch?v=zI47xd_2zDs`.

I.7 Reviewing Activities

Since July 2019, I have the honor of being an **Associate Editor** for IEEE Transactions on Signal Processing. I also served in 2019 as a **Guest Editor** for a special session on *network topology inference* for IEEE Transactions on Signal and Information Processing over Networks.

I also served as a reviewer for various journals, including: IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Signal Processing, IEEE Transactions on Signal and Information Processing over Networks, Journal of Machine Learning Research, Information Sciences, Journal of Selected Topics in Signal Processing, Neural Computation, IEEE Transactions on Circuits and Systems, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Information Theory, Mathematical and Computational Applications, Artificial Intelligence. Of course, I also often review for conferences.

I also participated as reviewer in the PhD defense of George Stamatescu, who defended in University of Adelaide in 2020. I was member of the jury of the thesis of Ludovic Danjean who defended in ENSEA in 2012.

I.8 Community Service

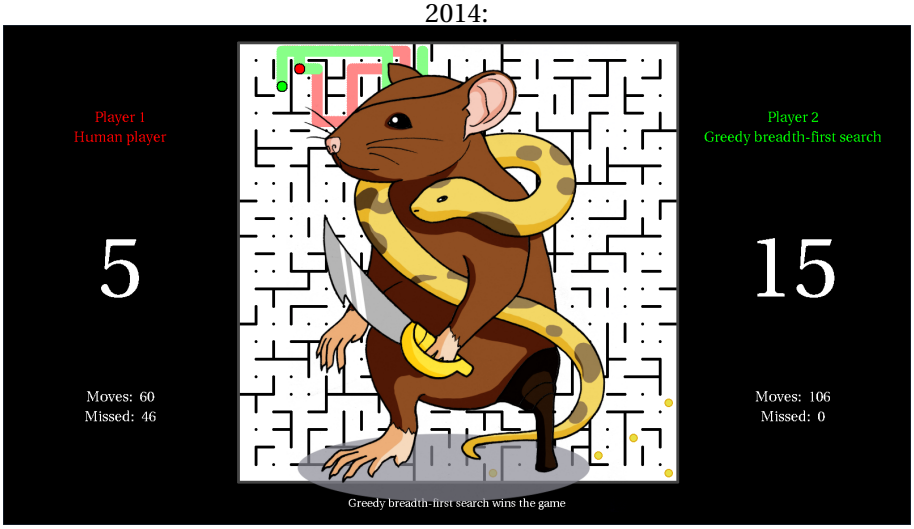
In the past few years, I have been in the **TPC** of several conferences, including EUSIPCO, IEEE GlobalSIP, ISTC and Cognitive. I also participated in the organization of ISTC 2016 and I am participating to that of ISTC 2020.

Since 2013, I have served as a member of the jury of the IEEE Xtreme online programming competition. IEEE Xtreme attracts each year of the order of a few thousands of competitors. It consists of a 24h programming marathon. As a member of the jury, I propose an exercise each year, I review other exercises and I answer to questions by the candidates during the competition.

In the summers of 2015, 2016, 2017 and 2018, I served as a jury for the Écoles Normales Supérieures of Paris-Saclay, Lyon and Rennes, national competitive examination. The first three years, I served in the “fundamental computer science” discipline.

2010:

2014:



2018:

Figure I.3: Evolution of the PyRat main game, from its initial version in 2010, to its updated version in 2014 then its final version in 2018.

Figure I.4: Poster to advertise for a “Pyrat Party”.

Each year, I had to prepare a dozen of exercises, and review about double that amount. Then, I interviewed about 60 students for a duration of 45 minutes each, for each of these three years. For the last year, I served in the "practical programming" discipline, which was a bit less demanding.

In 2011, I created an online programming competition for students in "classes préparatoires". It was named TaupIC, and typically attracted about 40 students each year. The principle was to propose increasingly difficult exercises, mixing applied mathematics and programming. For a duration of 6 hours the students were able to compete. The last session of the project was in 2015, when I started working as an interviewer at Écoles Normales Supérieures, since it would have been difficult to combine the two.

Also, I participated to the organisation of the first session of the Écoles Normales Supérieures webchat in 2014 at Institut Henri Poincaré. The principle was to provide an online platform for interactive discussions between representative of the Écoles Normales Supérieures and candidates.

I.9 Events Organization

I have coorganized several special sessions at international conferences:

1. In 2016, the special session was about "Coding and Machine Learning" and was organised during ISTC 2016 in Brest.
2. In 2017, the special session was about "Mental Information" and was organised during Cognitive 2017 in Athens, Greece.
3. In 2019, the special session was about "Graph Neural Networks" and was organised during SDM 2019 in Calgary, Canada.

Since 2016, I am also a coorganizer of the NeuroSTIC days in France. NeuroSTIC brings together scientists from various domains related to learning: computer science, electrical and computer engineering, neuroscience, psychology, signal processing... It usually attracts between 100 and 200 persons each time. In 2016 we organised it in Grenoble at INRIA. In 2017, it was in Brest. In 2019 it was in Sophia Antipolis.

I.10 Major Collaborators

Throughout the years, I have built several main collaborations with scientists in various countries. These include:

1. Professor Michael Rabbat, who was my supervisor in McGill University during my postdoc. He since joined Facebook AI Research in Montréal. Prof. Rabbat

is a specialist of distributed optimization and graph signal processing. We co-authored many articles since 2012 [21, 22, 23, 24, 25, 26, 27, 28, 9, 29, 30, 31, 32, 33, 34, 35].

2. Professor Warren Gross, who was also my supervisor in McGill University during my postdoc. He is since the head of the Electrical and Computer Engineering Department. Prof. Warren Gross is a specialist of hardware implementations of algorithms in coding and machine learning. Many of my contributions in the field of hardware implementations were co-authored with him [36, 25, 37, 38, 39, 40, 41, 30, 42, 43, 34, 35, 44].

3. Professor Matthias Löwe works with University of Münster in the Mathematics Department. Professor Franck Vermet is also a mathematician with the University of Brest. Since 2014, we worked together on a number of questions related to probabilities and learning [45, 15, 46, 47]. We worked on associative memories, transfer learning and nearest neighbor search. We started a project on deep neural networks robustness in 2019.

4. Professor Yoshua Bengio is a specialist of Artificial Intelligence and Deep Learning at Université de Montréal and Mila. He was the host during my one-year stay at Université de Montréal in 2019. Together, we worked on various problems related to compressing deep neural network architectures, including the introduction of trained shift layers [17, 48, 49, 50].

5. Professor Jian Tang is a specialist of Graph Neural Networks at HEC Montréal and Mila. During my stay at Université de Montréal, we worked together on various problems related to graph and deep neural networks, including the introduction of a graph smoothness loss to train deep neural network architectures, and the use of graph signal processing to increase robustness of deep neural networks [19, 20, 51, 52].

6. Professor Antonio Ortega is a specialist of Graph Signal Processing and Vision with the University of Southern California in Los Angeles. Together, we worked on numerous problems related to using graphs in deep neural networks, including visualization and training [19, 20, 18, 51, 52].

I.11 Invitations and Hosting

I.11.1 Main events

I have had the opportunity to visit multiple institutions:

1. McGill University for one year, as a postdoctoral researcher, under the supervision of Prof. Michael Rabbat and Prof. Warren Gross,
2. Université de Montréal for one year, as an invited professor, under the supervision of Prof. Yoshua Bengio,
3. University of Toronto for one week, hosted by Prof. Andreas Moshovos,
4. University of Southern California for ten days, hosted by Prof. Antonio Ortega,
5. Brown University, for one week, hosted by Prof. Thomas Serre,
6. Tohoku University for a few days, hosted by Prof. Hanyu,
7. INRIA Rennes for one week, hosted by Permanent Researcher Hervé Jégou and Director of Research Rémi Gribonval,
8. ENSEA for a few days, hosted by Prof. David Declercq,
9. Université Paris Diderot for a week, hosted by Permanent Researcher Olivier Serre,
10. Université of Nice Sophia Antipolis for a week, hosted by Prof. Benoît Mirramond,
11. École Polytechnique Fédérale of Lausanne for two weeks, hosted by Prof. Pierre Vandergheynst,
12. University of Adelaide for a week, hosted by the group of Prof. Ian Reid,
13. University of Tartu for one week, hosted by Prof. Vitaly Skachek.

We received many researchers throughout the years. Prof. Michael Rabbat notably stayed four months during his sabbatical. Prof. Warren Gross and Naoya Onizawa stayed for a week.

I.11.2 Seminars and Invited Talks

1. Exposé, "Graphs for Deep Neural Networks Latent Representations", *Edge Seminar*, Nice, France, July 2020.
2. Invited talk, "A Review of Compression Methods for Deep Convolutional Neural Networks", *TinyML*, Montréal, July 2020.
3. Invited talk, "A Review of Compression Methods for Deep Convolutional Neural Networks", *TinyML*, Montréal, July 2020.
4. Invited talk, "Deep Learning with Few Resources", *Pracom Seminar*, Brest, June 2020.

5. Invited talk, "A Review of Compression Methods for Deep Convolutional Neural Networks", *International Workshop of Emerging Technologies for Brainware LSI and its Applications*, Honolulu, Hawaii, USA, December 2019.
6. Invited talk, "Using Graphs to Visualize, Train and Improve Deep Neural Networks", *SNT seminar*, Luxembourg, November 2019.
7. Invited talk, "Efficient Implementations of Deep Learning Architectures", *Seminar of University of Rochester*, Rochester, USA, September 2019.
8. Invited talk, "Artificial Intelligence: the goods and bads", *Dialogues: les éclaireurs*, Brest, France, August 2019.
9. Exposé, "A Unified Deep Learning Formalism for Processing Graph Signals", *Graph Signal Processing workshop*, Minneapolis, June 2019.
10. Exposé, "A Unified Deep Learning Formalism for Processing Graph Signals", *GSP 2019*, Minneapolis, Minnesota, June 2019.
11. Invited talk, "Robust Deep Learning Inference with Limited Resources", *CUG*, Montréal, May 2019.
12. Invited talk, "Robust Deep Learning Inference with Limited Resources", *CUG*, Montréal, May 2019.
13. Invited talk, "Matching Convolutional Neural Networks with Graph Signals", *STATOS workshop*, Roma, Italy, September 2018.
14. Invited talk, "Convolutional Neural Networks for Signals on Graphs", *Deep Learning Workshop*, Technicolor, Rennes, September 2018.
15. Exposé, ""Graph Signal Processing for Machine Learning"", *FASIC workshop*, Adelaide, Australia, July 2018.
16. Invited talk, "Neural Networks and Artificial Intelligence", *Beyond Gynecological Surgery*, Clermont Ferrand, April 2018.
17. Exposé, "Convolutional Neural Networks on Irregular Domains", *Learning Theory reading group*, MILA, Montréal, April 2018.
18. Exposé, ""Dangers of AI"", *Semaine du cerveau*, March 2018.
19. Exposé, "Informational Neuroscience and Artificial Intelligence", ENIB, January 2018.
20. Exposé, "Extending Convolutional Neural Networks to Irregular Domains", University of South California, November 2017.
21. Invited talk, "L'IA et le HPC", *Round table at Collège de France to celebrate the 10 years of GENCI*, Collège de France, October 2017.

22. Exposé, "Generalizing Convolutional Neural Networks to Irregular Domains", *Visit at McGill*, McGill University, Montréal, July 2017.
23. Exposé, "Supervised Classification of Brain Imaging using Graph Signal Processing", *GSP-17*, Pittsburgh, PA, June 2017.
24. Exposé, "Tropical Graph Signal Processing", *GSP-17*, Pittsburgh, PA, June 2017.
25. Exposé, "S'inspirer du cerveau pour l'Intelligence Artificielle", *Brain's week*, Brest, March 2017.
26. Invited talk, "An attempt at characterizing graph translations in the vertex domain", *Barbados McGill gathering on Graph Signal Processing*, Barbados, February 2017.
27. Seminar, "Intelligence artificielle et neurosciences informationnelles", ENS-Lyon, January 2017.
28. Seminar, "Intelligence artificielle et neurosciences informationnelles", Kérichen high school, November 2016.
29. Seminar, "Vers une théorie de l'information mentale", *Century of the birth of Claude Shannon*, Institut Henri Poincaré, Paris, October 2016.
30. Invited talk, "Neurosciences informationnelles et intelligence artificielle", *Journée Intelligence Artificielle : le renouveau*, French Academy of Science, October 2016.
31. Seminar, "Réseaux de neurones binaires et applications", *Séminaire Institut Brestois du Numérique et des Mathématiques*, Brest, September 2016.
32. Invited talk, "Coding for machine learning and neural networks", *International symposium on turbo codes and iterative information processing*, Brest, September 2016.
33. Invited talk, "Mémoire associative basse consommation avec jonctions tunnel magnétiques", *Journée conférence débat "Atteindre une efficacité énérgétique extrême dans les systèmes de calcul avec la bio-inspiration"*, Orsay, April 2016.
34. Invited talk, "Binary neural networks and applications", *ENS Lyon ski seminar*, les sept laux, January 2016.
35. Exposé, "Binary associative memories and applications", Brown University, December 2015.
36. Exposé, "Binary associative memories and applications", McGill University, November 2015.

37. Exposé, "Error correcting graphs for explaining long term memory", Nice, March 2015.
38. Invited talk, "Is information encoding in the brain analogic or digital?", *Panel Cognitive 2015*, Nice, March 2015.
39. Seminar, "Informational neurosciences: error correcting codes in the brain", *Recent advances in computationnal neurosciences seminar*, ENS Lyon, January 2015.
40. Exposé, ""Computing with associative memories"", *Ski-week of ENS Lyon*, Les sept Laux, January 2015.
41. Exposé, "Exploiting high dimensionality for similarity search", *NIPS 2014 workshop*, Montréal, December 2014.
42. Exposé, "Error correcting codes and long term memory", EPFL, November 2014.
43. Invited talk, "Associative memories for computing", *Hipeac HPC Workshop*, Athens, Greece, October 2014.
44. Exposé, "Neurosciences informationnelles", *GRETSI summer school*, Peyresq, June 2014.
45. Associated with an invited talk, ""L'information mentale"", *UPMC Colloquium*, University Pierre et Marie Curie, March 2014.
46. Seminar, "Reconstructing a graph from path traces", *DECIDE team seminar*, Télécom Bretagne, February 2014.
47. Seminar, "Signal processing on graphs", *Télécom Bretagne lunch seminar*, February 2014.
48. Invited talk, "Resilient and energy efficient memories based on neuro-inspired codes", *2nd RIEC Symposium on Brain Functions and Brain Computer*, Sendai, Japan, February 2014.
49. Invited talk, "Un modèle numérique de la mémoire à long terme : l'information mentale", *Cantine numérique*, Quimper, November 2013.
50. Associated with an invited talk, "Codes sur graphes et mémoire cérébrale", *XXIV colloque Gretsi*, Brest, September 2013.
51. Invited talk, "L'information mentale", *Sicma doctoral school day*, Télécom Bretagne, September 2013.
52. Seminar, "Calculating using associative memories", *Thursday Seminar*, Tartu University, Estonia, June 2013.

53. Seminar, "Calculating using associative memories", *68nqrt Seminar*, IRISA, Rennes, June 2013.
54. Invited talk, "L'information mentale", *Tuesdays at Espace des sciences*, Rennes, May 2013.
55. Seminar, "When neural networks meet error correcting codes: towards new architectures for associative memories", *NeuroMathComp seminar*, INRIA Sophia Antipolis, Nice, April 2013.
56. Associated with an invited talk, "When neural networks meet error-correction coding: new perspectives in associative memories", *International Workshop on Neuromorphic and Brain-Based Computing Systems*, Grenoble, France, March 2013.
57. Seminar, "Les mémoires associatives : point de rencontre naturel entre calcul et mémoires", *ENS-Cachan, Dept. Computer Science and Telecommunications*, Rennes, March 2013.
58. Invited talk, "When neural networks meet error correcting codes: new perspectives for resilient associative memories", *Neuro Inspired Accelerators for Computing workshop, HiPEAC conference*, Berlin, Germany, January 2013.
59. Seminar, "Neural coding: from error correcting codes to associative memories", *ICI seminar*, ETIS, ENSEA, November 2012.
60. Seminar, "When neural networks meet error correcting codes: towards resilient associative memories", CEA-LETI, Grenoble, November 2012.
61. Seminar, "How to improve associative memories using neural coding?", *Neucod seminar*, Télécom Bretagne, Brest, September 2012.
62. Associated with an invited talk, "Looking at the neocortex as a distributed decoder", *7" International Symposium on Turbo Codes*, Gothenburg, Sweden, August 2012.
63. Invited talk, "Neural coding: a perspective for new associative memories", *Japan-France Frontiers of Engineering program*, Kyoto, Japan, February 2012.
64. Invited talk, "Nearly-optimal associative memories based on distributed constant weight codes", *Information Theory and Applications workshop*, San Diego, CA, February 2012.
65. Exposé, "Networks of Neural Cliques", Université de Montréal, November 2011.
66. Associated with an invited talk, "Graphs, codes and the brain", *14th International Symposium on Wireless Personal Multimedia Communications*, October 2011.

67. Seminar, "Neural computation: min, sum and max", *UBO mathematical department seminar*, University of Western Brittany, May 2011.
68. Exposé, Télécom Bretagne, March 2011.
69. Exposé, "Networks of Neural Cliques", McGill University, February 2011.
70. Exposé, "Réseaux de neurones parcimonieux à grande diversité d'apprentissage", École Supérieure de Physique et Chimie Industrielles, December 2010.
71. Seminar, "Networks of Neural Cliques: Some (not so) open issues", Télécom Bretagne, September 2010.
72. Seminar, *Breizh seminar of mathematics PhD students*, Western Brittany University, December 2009.
73. Seminar, *4th year students seminar*, École Normale Supérieure of Rennes, January 2009.

I.12 Publications

I.12.1 Journals and Books

1. B. Pasdeloup, V. Gripon, R. Alami and M. Rabbat, "Uncertainty Principle on Graphs," L. Stankovic and E. Sejdic, Vertex-Frequency Analysis of Graph Signals, pp. 317–340, April 2019.
2. C. Berrou and V. Gripon, "Petite mathématique du cerveau," Odile Jacob, September 2012.
3. A. Iscen, T. Furon, V. Gripon, M. Rabbat and H. Jégou, "Memory vectors for similarity search in high-dimensional spaces," in *IEEE Transactions on Big Data*, pp. 65–77, 2018.
4. G. B. Hacene, V. Gripon, N. Farrugia, M. Arzel and M. Jezequel, "Transfer Incremental Learning Using Data Augmentation," in *Applied Sciences*, Volume 8, Number 12, 2018.
5. B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor and M. Rabbat, "Characterization and Inference of Graph Diffusion Processes from Observations of Stationary Signals," in *IEEE Transactions on Signal and Information Processing over Networks*, Volume 4, Number 3, pp. 481–496, September 2018.
6. A. Mheich, M. Hassan, M. Khalil, V. Gripon, O. Dufor and F. Wendling, "SimiNet: a Novel Method for Quantifying Brain Network Similarity," in *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, Volume 40, Number 9, pp. 2238–2249, September 2018.

7. V. Gripon, M. Löwe and F. Vermet, "Associative Memories to Accelerate Approximate Nearest Neighbor Search," in *Applied Sciences*, Volume 8, Number 9, September 2018.
8. V. Gripon, J. Heusel, M. Löwe and F. Vermet, "A Comparative Study of Sparse Associative Memories," in *Journal of Statistical Physics*, Volume 164, pp. 105–129, 2016.
9. B. Boguslawski, V. Gripon, F. Seguin and F. Heitzmann, "Twin Neurons for Efficient Real-World Data Distribution in Networks of Neural Cliques. Applications in Power Management in Electronic circuits," in *IEEE Transactions on Neural Networks and Learning Systems*, Volume 27, Number 2, pp. 375–387, 2016.
10. X. Jiang, V. Gripon, C. Berrou and M. Rabbat, "Storing sequences in binary tournament-based neural networks," in *IEEE Transactions on Neural Networks and Learning Systems*, Volume 27, Number 5, pp. 913–925, 2016.
11. H. Jarollahi, V. Gripon, N. Onizawa and W. J. Gross, "Algorithm and Architecture for a Low-Power Content-Addressable Memory Based on Sparse-Clustered Networks," in *Transactions on Very Large Scale Integration Systems*, Volume 27, Number 2, pp. 375–387, 2016.
12. A. Aboudib, V. Gripon and G. Coppin, "A Neural Network Model for Solving the Feature Correspondence Problem," in *Lecture Notes in Computer Science*, Volume 9887, pp. 439–446, September 2016.
13. G. Soulié, V. Gripon and M. Robert, "Compression of Deep Neural Networks on the Fly," in *Lecture Notes in Computer Science*, Volume 9887, pp. 153–170, September 2016.
14. A. Aboudib, V. Gripon and G. Coppin, "A Biologically Inspired Framework for Visual Information Processing and an Application on Modeling Bottom-Up Visual Attention," in *Cognitive Computation*, pp. 1–20, September 2016.
15. F. Leduc-Primeau, V. Gripon, M. Rabbat and W. J. Gross, "Fault-Tolerant Associative Memories Based on c-Partite Graphs," in *IEEE Transactions on Signal Processing*, Volume 64, Number 4, pp. 829–841, 2015.
16. H. Jarollahi, N. Onizawa, V. Gripon, N. Sakimura, T. Sugibayashi, T. Endoh, H. Ohno, T. Hanyu and W. J. Gross, "A Non-Volatile Associative Memory-Based Context-Driven Search Engine Using 90 nm CMOS MTJ-Hybrid

Logic-in-Memory Architecture," in *Journal on Emerging and Selected Topics in Circuits and Systems*, Volume 4, pp. 460–474, 2014.

17. H. Jarollahi, N. Onizawa, V. Gripon and W. J. Gross, "Algorithm and Architecture of Fully-Parallel Associative Memories Based on Sparse Clustered Networks," in *Journal of Signal Processing Systems*, pp. 1–13, 2014.
18. B. K. Aliabadi, C. Berrou, V. Gripon and X. Jiang, "Storing sparse messages in networks of neural cliques," in *IEEE Transactions on Neural Networks and Learning Systems*, Volume 25, pp. 980–989, 2014.
19. V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," in *IEEE Transactions on Neural Networks*, Volume 22, Number 7, pp. 1087–1096, July 2011.

I.12.2 Conference Proceedings and Preprints

1. Y. Hu, V. Gripon and S. Pateux, "Exploiting Unsupervised Inputs for Accurate Few-Shot Classification," in *ArXiv Preprint: 2001.09849*, 2020.
2. M. Nikolić, G. B. Hacene, C. Bannon, A. D. Lascorz, M. Courbariaux, Y. Bengio, V. Gripon and A. Moshovos, "BitPruning: Learning Bitlengths for Aggressive and Accurate Quantization," in *ArXiv Preprint: 2002.03090*, 2020.
3. G. B. Hacene, C. Lassance, V. Gripon, M. Courbariaux and Y. Bengio, "Attention Based Pruning for Shift Networks," in *Arxiv Preprint*, 2019.
4. Q. Jodelet, V. Gripon and M. Hagiwara, "Transfer Learning with Sparse Associative Memories," in *International Conference on Artificial Neural Networks*, pp. 497–512, 2019.
5. G. B. Hacene, V. Gripon, N. Farrugia, M. Arzel and M. Jezequel, "Efficient Hardware Implementation of Incremental Learning and Inference on Chip," in *ArXiv Preprint*, 2019.
6. C. Lassance, Y. Latif, R. Garg, V. Gripon and I. Reid, "Improved Visual Localization via Graph Smoothing," in *ArXiv Preprint*, 2019.
7. C. Lassance, M. Bontonou, G. B. Hacene, V. Gripon, J. Tang and A. Ortega, "Deep geometric knowledge distillation with graphs," in *ArXiv Preprint: 1911.03080*, 2019.
8. M. Bontonou, C. Lassance, V. Gripon and N. Farrugia, "Comparing linear structure-based and data-driven latent spatial representations for sequence prediction," in *Wavelets and Sparsity XVIII*, San Diego, USA, August 2019.

9. C. Lassance, V. Gripon, J. Tang and A. Ortega, “Robustesse structurelle des architectures d’apprentissage profond,” in *GRETSI*, August 2019.
10. M. Bontonou, C. Lassance, J. Vialatte and V. Gripon, “Un modèle unifié pour la classification de signaux sur graphe avec de l’apprentissage profond,” in *GRETSI*, August 2019.
11. G. B. Hacene, F. Leduc-Primeau, A. B. Soussia, V. Gripon and F. Gagnon, “Robustesse des réseaux de neurones profonds aux défaillances mémoire,” in *GRETSI*, August 2019.
12. M. Bontonou, C. Lassance, G. B. Hacene, V. Gripon, J. Tang and A. Ortega, “Introducing Graph Smoothness Loss for Training Deep Learning Architectures,” in *Data Science Workshop*, pp. 160–164, June 2019.
13. C. Lassance, V. Gripon, J. Tang and A. Ortega, “Structural Robustness for Deep Learning Architectures,” in *Data Science Workshop*, pp. 125–129, June 2019.
14. M. Bontonou, C. Lassance, J. Vialatte and V. Gripon, “A Unified Deep Learning Formalism For Processing Graph Signals,” in *SDM Special Session on Graph Neural Networks*, May 2019.
15. G. B. Hacene, F. Leduc-Primeau, A. B. Soussia, V. Gripon and F. Gagnon, “Training modern deep neural networks for memory-fault robustness,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1–5, May 2019.
16. C. E. R. K. Lassance, J. Vialatte and V. Gripon, “Matching Convolutional Neural Networks without Priors about Data,” in *Proceedings of Data Science Workshop*, pp. 234–238, 2018.
17. N. Grelier, C. R. K. Lassance, E. Dupraz and V. Gripon, “Graph-Projected Signal Processing,” in *IEEE GlobalSIP*, pp. 763–767, 2018.
18. G. B. Hacene, V. Gripon, M. Arzel, N. Farrugia and Y. Bengio, “Quantized Guided Pruning for Efficient Hardware Implementations of Convolutional Neural Networks,” in *Arxiv Preprint*, 2018.
19. V. Gripon, G. B. Hacene, M. Löwe and F. Vermet, “Improving Accuracy of Nonparametric Transfer Learning Via Vector Segmentation,” in *proceedings of ICASSP*, pp. 2966-2970, April 2018.
20. V. Gripon, A. Ortega and B. Girault, “An Inside Look at Deep Neural Networks using Graph Signal Processing,” in *Proceedings of ITA*, pp. 1–9, February 2018.
21. J. Vialatte, V. Gripon and G. Coppin, “Learning Local Receptive Fields and their Weight Sharing Scheme on Graphs,” in *Proceedings of GlobalSip*, pp. 623–627, 2017.

22. M. Ménoret, N. Farrugia, B. Pasdeloup and V. Gripon, "Evaluating Graph Signal Processing for Neuroimaging Through Classification and Dimensionality Reduction," in *Proceedings of GlobalSip*, pp. 618–622, 2017.
23. G. B. Hacene, V. Gripon, N. Farrugia, M. Arzel and M. Jezequel, "Incremental Learning on Chip," in *Proceedings of GlobalSip*, pp. 789–792, 2017.
24. G. B. Hacene, V. Gripon, N. Farrugia, M. Arzel and M. Jezequel, "Budget Restricted Incremental Learning with Pre-Trained Convolutional Neural Networks and Binary Associative Memories," in *Proceedings of SIPS*, pp. 1063–1073, 2017.
25. V. Gripon, "Tropical Graph Signal Processing," in *Proceedings of the Asilomar conference*, pp. 50–54, October 2017.
26. E. Coyac, V. Gripon, C. Langlais and C. Berrou, "Robust Associative Memories Naturally Occuring From Recurrent Hebbian Networks Under Noise," in *Arxiv Preprint*, September 2017.
27. T. Stérin, N. Farrugia and V. Gripon, "An Intrinsic Difference Between Vanilla RNNs and GRU Models," in *Proceedings of Cognitive*, pp. 76–81, February 2017.
28. G. B. Hacene, V. Gripon, N. Farrugia, M. Arzel and M. Jezequel, "Finding All Matches in a Database using Binary Neural Networks," in *Proceedings of Cognitive*, pp. 59–64, February 2017.
29. E. Coyac, V. Gripon, C. Langlais and C. Berrou, "Performance of Neural Clique Networks Subject to Synaptic Noise," in *Proceedings of Cognitive*, pp. 4–9, February 2017.
30. N. Grelier, B. Pasdeloup, J. Vialatte and V. Gripon, "Neighborhood-Preserving Translations on Graphs," in *Proceedings of GlobalSIP*, pp. 410–414, October 2016.
31. P. Tigréat, C. R. K. Lassance, X. Jiang, V. Gripon and C. Berrou, "Assembly Output Codes for Learning Neural Networks," in *Proceedings of the 9th International Symposium on Turbo Codes and Iterative Information Processing*, pp. 285–289, September 2016.
32. A. Aboudib, V. Gripon and G. Coppin, "A Turbo-Inspired Iterative Approach for Correspondence Problems of Image Features," in *Proceedings of the 9th International Symposium on Turbo Codes and Iterative Information Processing*, pp. 226–230, September 2016.
33. E. Coyac, V. Gripon, C. Langlais and C. Berrou, "Distributed Coding and Synaptic Pruning," in *Proceedings of the 9th International Symposium on Turbo Codes and Iterative Information Processing*, pp. 206–210, September 2016.

34. D. Ferro, V. Gripon and X. Jiang, "Nearest Neighbour Search Using Binary Neural Networks," in *Proceedings of IJCNN*, pp. 5106–5112, July 2016.

35. E. Coyac, V. Gripon and C. Langlais, "Impact du bruit synaptique sur les performances des réseaux de cliques neurales," in *Proceedings of the GRETSI conference*, 2015.

36. R. Danilo, V. Gripon, P. Coussy and L. Conde-Canencia, "Réseaux de Clusters de Neurones Restreints," in *Proceedings of the GRETSI conference*, 2015.

37. B. Pasdeloup, V. Gripon, G. Mercier and D. Pastor, "Vers une caractérisation de la courbe d'incertitude pour des graphes portant des signaux," in *Proceedings of the GRETSI conference*, 2015.

38. A. Mheich, M. Hassan, F. Wendling, M. Khalil, O. Dufor, V. Gripon and C. Berrou, "SimNet: A new algorithm for measuring brain networks similarity," in *Proceedings of the ICABME international conference*, pp. 119–122, 2015.

39. B. Pasdeloup, M. Rabbat, V. Gripon, D. Pastor and G. Mercier, "Graph Reconstruction from the Observation of Diffused Signals," in *Proceedings of the 53rd Allerton Conference*, pp. 1386–1390, October 2015.

40. B. P. R. A. V. Gripon and M. Rabbat, "Toward an uncertainty principle for weighted graphs," in *Proceedings of the 23rd European Signal Processing Conference*, pp. 1496–1500, July 2015.

41. R. Danilo, V. Gripon, P. Coussy, L. Conde-Canencia and W. J. Gross, "Restricted Clustered Neural Network for Storing Real Data," in *proceedings of GLSVLSI conference*, pp. 205–210, May 2015.

42. R. Danilo, H. Jarollahi, V. Gripon, P. Coussy, L. Conde-Canencia and W. J. Gross, "Algorithm and Implementation of an Associative Memory for Oriented Edge Detection Using Improved Clustered Neural Networks," in *Proceedings of ISCAS Conference*, pp. 2501–2504, May 2015.

43. A. Aboudib, V. Gripon and G. Coppin, "A Model of Bottom-Up Visual Attention Using Cortical Magnification," in *Proceedings of ICASSP*, pp. 1493–1497, April 2015.

44. A. Mheich, M. Hassan, V. Gripon, O. Dufor, M. Khalil, C. Berrou and F. Wendling, "A novel algorithm for measuring graph similarity: application to brain networks," in *Proceedings of the IEEE EMBS Neural Engineering Conference*, pp. 1068–1071, April 2015.

45. A. Aboudib, V. Gripon and B. Tessiau, “Implementing Relational-Algebraic Operators for Improving Cognitive Abilities in Networks of Neural Cliques,” in *Proceedings of Cognitive*, pp. 36–41, March 2015.

46. C. Yu, V. Gripon, X. Jiang and H. Jégou, “Neural Associative Memories as Accelerators for Binary Vector Search,” in *Proceedings of Cognitive*, pp. 85–89, March 2015.

47. S. Larroque, E. S. Gooya, V. Gripon and D. Pastor, “Using Tags to Improve Diversity of Sparse Associative Memories,” in *Proceedings of Cognitive*, pp. 1–7, March 2015.

48. E. S. Gooya, D. Pastor and V. Gripon, “Automatic face recognition using SIFT and networks of tagged neural cliques,” in *Proceedings of Cognitive*, pp. 57–61, March 2015.

49. V. Gripon, V. Skachek and M. Rabbat, “Sparse Binary Matrices as Efficient Associative Memories,” in *Proceedings of the 52nd Allerton conference*, pp. 499–504, October 2014.

50. H. Jarollahi, N. Onizawa, V. Gripon, T. Hanyu and W. J. Gross, “Algorithm and Architecture for a Multiple-Field Context-Driven Search Engine Using Fully-Parallel Clustered Associative Memories,” in *Proceedings of SiPS*, pp. 1–6, October 2014.

51. C. Berrou, O. Dufor, V. Gripon and X. Jiang, “Information, Noise, Coding, Modulation: What about the Brain?,” in *Proceedings of the 8th symposium on Turbo Codes and Iterative Information Processing*, pp. 167–172, August 2014.

52. Z. Yao, V. Gripon and M. Rabbat, “A GPU-based Associative Memory using Sparse Neural Networks,” in *Proceedings of the PCNN-14 conference*, pp. 688–692, July 2014.

53. B. Boguslawski, V. Gripon, F. Seguin and F. Heitzmann, “Huffman Coding for Storing Non-uniformly Distributed Messages in Networks of Neural Cliques,” in *proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, volume 1*, pp. 262–268, July 2014.

54. A. Aboudib, V. Gripon and X. Jiang, “A study of retrieval algorithms of sparse messages in networks of neural cliques,” in *Proceedings of Cognitive 2014*, pp. 140–146, May 2014.

55. M. Rabbat and V. Gripon, “Towards a Spectral Characterization of Signals Supported on Small-World Networks,” in *ICASSP*, pp. 4793–4797, May 2014.

56. F. Leduc-Primeau, V. Gripon, M. Rabbat and W. Gross, "Cluster-based Associative Memories Built From Unreliable Storage," in *ICASSP*, pp. 8370–8374, May 2014.
57. V. Gripon, V. Skachek and M. G. Rabbat, "Sparse Structured Associative Memories as Efficient Set-Membership Data Structures," in *Proceedings of the 51st Allerton conference*, pp. 500–505, October 2013.
58. V. Gripon and X. Jiang, "Mémoires associatives pour observations floues," in *Proceedings of XXIV-th Gretsi seminar*, September 2013.
59. V. Gripon and M. Rabbat, "Maximum Likelihood Associative Memories," in *Proceedings of Information Theory Workshop*, pp. 1–5, September 2013.
60. V. Gripon and M. Rabbat, "Reconstructing a Graph from Path Traces," in *Proceedings of International Symposium on Information Theory*, pp. 2488–2492, July 2013.
61. H. Jarollahi, V. Gripon, N. Onizawa and W. J. Gross, "A Low-Power Content-Adressable-Memory Based on Clustered-Sparse-Networks," in *Proceedings of 24th International Conference on Application-specific Systems, Architectures and Processors*, pp. 642–653, June 2013.
62. H. Jarollahi, N. Onizawa, V. Gripon and W. J. Gross, "Reduced-complexity binary-weight-coded associative memories," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pp. 2523–2527, May 2013.
63. V. Gripon, M. Rabbat, V. Skachek and W. J. Gross, "Compressing multisets using tries," in *Proceedings of Information Theory Workshop*, Lausanne, Switzerland, pp. 647–651, September 2012.
64. V. Gripon, V. Skachek, W. J. Gross and M. Rabbat, "Random clique codes," in *Proceedings of 7" International Symposium on Turbo Codes and Iterative Information Processing*, Gothenburg, Sweden, pp. 121–125, August 2012.
65. X. Jiang, V. Gripon and C. Berrou, "Learning long sequences in binary neural networks," in *Proceedings of Cognitive 2012*, Nice, France, pp. 165–170, July 2012.
66. H. Jarollahi, N. Onizawa, V. Gripon and W. J. Gross, "Architecture and Implementation of an Associative Memory Using Sparse Clustered Networks," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 2901–2904, May 2012.
67. V. Gripon and C. Berrou, "Nearly-optimal associative memories based on distributed constant weight codes," in *Proceedings of Information Theory and Applications Workshop*, San Diego, CA, USA, pp. 269–273, February 2012.

68. V. Gripon and C. Berrou, “A simple and efficient way to store many messages using neural cliques,” in *Proceedings of IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, Paris, France, pp. 54–58, April 2011.

69. C. Berrou and V. Gripon, “Coded Hopfield networks,” in *Proceedings of 6" International Symposium on Turbo Codes and Iterative Information Processing*, Brest, France, pp. 1–5, September 2010.

70. V. Gripon and O. Serre, “Qualitative Concurrent Stochastic Games with Imperfect Information,” in *Proceedings of 36th International Colloquium of Automata, Languages and Programming*, Springer, Lecture Notes in Computer Science, Rhodes, Greece, pp. 200–211, July 2009.

Chapter II

Summary of Past Contributions and Research

Because they have revolutionised Artificial Intelligence (AI) by providing the missing link between the actual physical world and digital representations and processing, it goes without saying that (artificial) neural networks have become a central technology of our time.

Interestingly, when I began my PhD back in 2008, neural networks were almost unanimously considered obsolete. Their inaptitude to solve real world problems, added to the impossibility to formally and mathematically summarize their key properties, made them very unpopular, in particular in France, where a large part of the research in the field is driven by theorists. I remember applying to professor positions in 2013 in France. In the computer science section, counting more than 100 open positions that year, only one of them mentioned the term “neural networks”.

In this chapter, we shall review my main contributions in the fields of neural networks and graph signal processing.

II.1 Introduction

As an introduction to this chapter, we present in the following sections basic notations and concepts about Neural Networks, Indexation, Associative Memories and Classification.

II.1.1 Neural Networks

Artificial neural networks, or simply “neural networks”, were originally introduced as a simplified model for describing their biological counterpart. We will discuss these

aspects later in the document, but for now let us focus on neural networks as mathematical objects, disregarding any bounds with biological aspects. Neural networks are obtained by assembling elementary blocks, often called "layers" in the literature. In the vast majority of cases, a layer is a mathematical function of the form:

$$\mathbf{x} \mapsto f(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{II.1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are tensors containing tunable parameters, $\mathbf{x}$ is an input tensor and f is a nontrainable nonlinear function. We will often make use of tensors in this document. The fact is the theory of tensor spaces can quickly become unnecessarily complicated and cumbersome. For the sake of better readability, we shall make many details implicit in this document, like in this equation the way the product between $\mathbf{W}$ and $\mathbf{x}$ is computed.

Assemblies can be various, ranging from line architectures, in which layers are stacked one of top of another, to elaborate recurrent networks in which the flow of information can take the form of a complex graph (e.g. [53]). When the layers are assembled in a line, the result neural network function is a composition of the functions of each layer. In more complicated architectures, layer outputs can be aggregated using various operators: sums, concatenations or element-wise extremum functions for example. In all cases, a neural network is thus a particularly-shaped function F from an input tensor space $\mathcal{I}$ to an output tensor space $\mathcal{O}$.

All trainable parameters of the network function are usually randomly initialized [54] or set to a neutral value (in most cases 0). As such, a nontrained architecture is meaningless, unless looking for fancy ways to generate random projections. But when parameters are tuned, neural networks can solve a variety of problems. This is mainly due to the fact that neural networks are universal approximators [55, 56][1]. The consequence is that it is known that for almost any real world problem that consists in finding a mathematical function, there exists a neural network whose function approximately solves (arbitrarily closely) the given problem.

The terminology of neural networks can be confusing to newcomers. Let us define some of the most generic terms here. A *neuron* is a coordinate of a tensor $\mathbf{x}$, where $\mathbf{x}$ is either the input or output of a layer in a neural network architecture. When processing a certain input, the value of a neuron is called *activation*. A neuron that is neither part of the input or output of the network function F is termed *hidden*.

A *connection* is a coordinate of a weight tensor $\mathbf{W}$ of a layer of the architecture. Each connection has a specific value, called a *weight*. In certain cases such as convolutional

[1] At least for continuous functions of a finite number of real variables with a compact support, following mild conditions

layers, various connections can share the same weight. When that happens, those connections can be thought of as pointers to the same memory address.

The set of all neurons and connections of a given architecture are part of the so-called *hyperparameters*, among others. To the contrary, the weights are referred to as *parameters*. In most cases hyperparameters are handcrafted (or obtained through random or grid searches), whereas parameters are tuned using automatic optimization techniques.

Given an input tensor $\mathbf{i}$, a neural network produces an output tensor $\mathbf{o}$, which is often compared to a *target* tensor $\mathbf{t}$, having the same shape as $\mathbf{o}$.

In some contexts, it might be useful to define an *energy function* associated with a neural network. This is in particular useful for some classes of recurrent neural networks in which the output tensor has the same shape as the input tensor, and a given input is processed several times through the whole architecture. The energy function is then often defined so that its minimum points are the fixed points of this iterative process.

Having fixed the hyperparameters of an architecture, a loss function L takes three arguments: an input tensor $\mathbf{i}$, a set of parameters θ, and a target tensor $\mathbf{t}$. Often it associates a nonnegative real value which is 0 if and only if the output of the neural network when processing $\mathbf{i}$ with parameters θ equals $\mathbf{t}$.

If both the loss function and the network function are differentiable, a neural network can be *trained* to minimize the loss over a set of couples $(\mathbf{i}, \mathbf{t})$, called *training examples*, using variants of the stochastic gradient descent algorithm. Thanks to the compositional nature of a neural network function F, this descent can be efficiently implemented using a backpropagation mechanism, making use of the chain rule of differentiation.

To leverage the parallel abilities of graphical processing units, most libraries made to optimize neural network architectures make use of batches and mini-batches. A batch is a set of examples that are processed concurrently, so that the descent is performed using the average gradients computed through the batch. In some cases, batches are subdivided in mini-batches, that are subdivision of batches used for exploiting increased parallelism or for limiting the memory usage.

Finally, let us give a few examples of nonlinear functions f commonly used for neural networks. Those include `sign`, `sigmoid`, `heavyside`, `tanh` and `relu` and are applied dimension-wise. A less common nonlinear function is a winner-takes-all (or `max`), that outputs a binary ($\{0, 1\}$) tensor containing 1 only where the value is maximum.

Note that the choice of the gradient descent algorithm, of its parameters, of the dimension of each tensor weight, of the nonlinear functions, of the loss function are

also often considered hyperparameters.

In the next subsections, we introduce two important problems that are the core of the contributions described in this document.

II.1.2 Indexing and Associative Memories

Even long before the rise of deep learning, it used to be very common to store data elements as tensors. Of course, any element stored using digital memory can be represented as an array of bits. But the key idea behind the field of indexing is to use vector representations so that there exists a metric on these vectors that is meaningful with respect to the data.

Let us point out that this idea is not as trivial as it may seems at first. In many cases, databases are filled with data that is inherently poorly fitted to linear transforms. Examples are text documents, compressed vision or audio signals or even programming code. Because they are among the simplest mathematical objects, vector or tensor spaces are very attractive domain candidates to embed such data. But finding morphisms from a very ambiguous domain where proximity is ill-defined to tensor spaces is definitely a very hard challenge.

For example, in the past few years it has become increasingly popular to find embeddings for natural language processing, where words, sentences or even paragraphs are represented as vectors such that the proximity between vectors is strongly related to a proximity in the semantic of the corresponding texts [57]. Other embeddings have been used for images for decades [58]. A naive way to obtain such vectors is to use bag-of-words representations [59, 60]. In that case, dimensions of the vector represent features, and objects are represented as indicator vectors on this set of features (with possible repetitions).

With the exponential growth of data, efficiently finding elements in a database of embeddings has become a major problem. Two examples of queries are:

1. **Exact match:** is this element already in the database?
2. **Approximate match:** what are the elements closest to this element in the database?

Of course, it is always possible to exactly answer these queries by using a bruteforce approach where all elements in the database are compared against the query. But the complexity of this solution scales at least linearly with the number of elements stored in the database, leading to often unpractical delays to obtain the answer to a query.

Related to the question of indexing are associative memories. Associative memories are devices able to store then retrieve pieces of information (most of the time represented using vectors), using the content instead of an explicit address.

As such, associative memories are able to solve the problems of exact and approximate match. We shall see in the coming chapters that most effective implementations of approximate associative memories use neural networks.

Formally, in the remaining of this document, when interested in the problem of indexing we will denote $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ a database comprising n elements. A query vector (or tensor) is denoted $\mathbf{x}$.

Then the exact match problem consists in computing: $\mathbb{1}_{\mathbf{x}\in\mathcal{X}}$, and the approximate match problem consists in computing: $\arg\min_{\mathbf{x}'\in\mathcal{X}} d(\mathbf{x}, \mathbf{x}')$, where d is a given metric (or quasi-metric).

Note that the approximate match problem is often referred to as nearest neighbor search (NN-search) in the literature. In some cases, one can prove the complexity required to solve NN-search scales linearly with the number of elements in the dataset and their dimension. As such, achieving sub-linear complexities requires to approximately solve the NN-search problem.

II.1.3 Classification and cross-validation

In the field of machine learning, classification has always been one of the most prominent problems. Classification is at its core an ill-posed problem of regression, where the objective is to find a function that extrapolate examples sampled from an initial distribution, and the output can only take a finite number of values. A very common application domain of classification is the automatic labelling of natural images captured using a digital camera.

Nontrivial classification problems are clearly ill-posed as there are multiple ways to extrapolate the given examples. Worse, it is often preferred a solution function that is not fully consistent with all provided examples if it yields more regularity, what is known as the "bias-variance" trade-off in the field of machine learning.

More generally, the implicit goal of classification is to achieve generalization. As such, classification can be considered an alternative to classical programming. Indeed, in the latter the principle is to rely on the ability of an expert programmer to solve a problem by making explicit a nonambiguous sequence of instructions (i.e. an algorithm) that, when applied to a given input, produces the expected output. In the former, the fundamental principles that are required to associate the input with the output are to be found automatically by the algorithm through optimization techniques.

There can be many reasons to use classification: for example the impossibility to explicitly describe a solution to the problem, the complexity of an exact solution or

the prohibitive cost of experts...

In most cases described in this document, the main reason to rely on classification is the fact no expert knows how to explicitly solve the corresponding problems. As a consequence, it is delicate to assess whether the obtained solution is actually an authentic generalization or not. In almost all works in the domain, measuring generalization is performed thanks to cross-validation.

Cross-validation is a technique in which a dataset used for classification is split into two parts. The first part is used to train a given solution. The second one to assess the generalization abilities of the obtained solution. We will see in this document that cross-validation can be problematic, in particular when the two parts are sampled uniformly, and thus following the same distribution. The fact a classifier specializes too much in biases of the training example distribution is often referred to as "overfitting" in the literature. In order to avoid overfitting, authors use different techniques. Those include *data augmentation*, where the number of training examples is artificially augmented by creating new ones from original ones and the use of *regularizers* that prevent the gradient descent from being too sharp by constraining the network function, either hardly (e.g. using convolutions) or softly (e.g. L_2-regularization).

Usually, the performance of a classifier is measured thanks to the accuracy, which is the complement of the predictor error rate. When challenges are hard, a classifier is allowed to propose several guesses, and an error is counted in the extreme case where none of the guesses is correct. In that case, we speak of "top-n" accuracy, where n is the number of guesses. For most competitive classification benchmarks, *deep learning* methods are the golden standard in terms of accuracy. The term "deep learning" refers to neural networks comprising a significant number of layers, trained through backpropagation.

If originally most of the efforts of the community used to be focused on classification algorithms to improve accuracy of the methods (notably in the 20th century), it has been known for quite a while that finding adequate representations of data is actually the bottleneck for accuracy. Interestingly, deep learning methods abilities to reach state-of-the-art accuracies have often been explained by the fact both representations and classification are trained jointly [61].

It is worth mentioning that classification is a *supervised learning* problem, meaning that learning is performed with the help of a teacher (here, the teacher is used to obtain the target tensors of the training examples). A related problem is that of *semi-supervised classification* in which only a portion of the examples contain the target tensor.

II.1.4 Biological inspiration

As previously mentioned, neural networks were originally introduced as a simplified mathematical model to explain the behavior of biological neural networks [62]. Despite this initial relation, it is fair to say that modern artificial neural networks share little similarities with their biological counterpart.

As a matter of fact, biological neural networks contain hundreds of different types of neurons in the hypocampus alone [63], yielding various connection ranges and behaviors. The role of each type of neuron remains a highly open question. At the scale of architectures, simplified models of the cortex suggest an organization in layers, for example in the visual areas [64]. But these layers are very different from the model of artificial neural networks: layer-wise connections and backward connections are numerous. There is no such thing as a parallel computation in which the neurons are simultaneously updated in a same layer. Time is a crucial factor, and synchronization is argued to be key of the dynamic of neurons [65].

The recent literature in the field of deep learning tend to reduce the number of references to the actual brain [66]. This is a good thing, because the terminology is confusing to the general public and even to some of the scientific communities. The key differences between neuro-inspiration and neuro-mimetism are only known by the experts.

Understanding the organization and functioning of the brain is a domain that has known important breakthrough in the past few years, notably thanks to the advances in neuroimaging. Two techniques massively used to observe a living brain are fMRIs and EEGs. Scans in fMRIs allow to observe the brain with a very good spatial resolution (up to several 100k of regions), but with a very poor time resolution ($\approx$1s). To the contrary, EEGs offer a limited spatial resolution ($\approx$100s of regions) but with a far better time resolution ($\approx$1ms). Because of the irregular structure of the neocortex, analyzing these signals is an open major scientific discipline. We discuss this problem in the next chapter.

If artificial neural networks differentiate themselves from biological ones, the problems of indexing and classification echo to cognitive abilities of the human brain. As a matter of fact, the associativity of the human memory has been discussed in the psychology literature for decades [67]. Similarly, the generalization abilities of the humain brain are still unmatched for a variety of challenges [68].

In the light of these observations, multiple scientific paths are being followed in the literature. In the main conferences interested in deep learning, the inspiration from the brain now occupies but a very small portion of the accepted papers. The

main motivation is drawn from engineering and fundamental mathematical or artificial intelligence problems. To the contrary, other scientific communities reclaim the terminology of neural networks, motivated by the fact there are still a lot of problems that can only be solved by the human brain.

Diversity is a fundamental aspect of research. In the multiple contributions summarized in this document, we followed these two paths, being sometimes clearly inspired by the functioning of the brain, or even the will to better understand it, and sometimes motivated by engineering aspects of machine learning. There is no reason to think that these two motivations are compatible in practice, and as such we will make sure to be clear about the motivations of each contribution while introducing them.

II.1.5 Notations and Outline

Throughout the document, we try to use notations as consistently as possible. Bold letters usually refer to tensors, such as for example $\mathbf{x}$. When indices are added such as $\mathbf{x}_i$, they refer to the index of tensor in a list. A specific dimension of the tensor can be denoted $\mathbf{x}[i]$. We follow the Python conventional notations, where $\mathbf{x}[:, i]$ refers to the (complete) i-th column of $\mathbf{x}$. Capital letters refer to sets and tuples, such as $\mathcal{G}$ or E.

In Section II.2, we introduce contributions related to graphs and graph signals. We discuss introducing translations and an uncertainty principle for graph signals. We explain how to infer a graph structure from the observation of signals. We also deal with the case of semiring algebraic structures. We apply these tools to the study of time-varying graph signals, neuroimaging data, retina-inspired visual processing systems and graph matching.

In Section II.3, we discuss contributions in the field of neural networks for indexing. We introduce the problem of looking at the brain as an information processing system. We discuss proposed families of associative memories and their extensions to deal with multiple problems. We introduce a mathematical comparison of sparse associative memories. We discuss applications to nonuniformly distributed data, to perform approximate nearest neighbor search and to implement lightweight content adressable memories and context search engines.

In Section II.4, we present contributions in the field of neural networks for learning. We introduce methods to extend convolutional neural networks to irregular domains. We explain how to exploit specific statistical properties of transfer learning features to improve accuracy of downstream classifiers. We propose novel definitions of neural network robustness and define new loss function based on graphs. We present meth-

ods to compress deep learning architectures, to perform incremental learning on chip, to solve the feature correspondence problem in image matching and we explain how to use graph signals to monitor the training of deep learning architectures.

II.2 Graphs and Graph Signals

II.2.1 Graph Signal Processing

II.2.1.1 Graphs

Graphs are ubiquitous. As a generic tool to model relationships between objects, they are suited to countless problems in computer science and related disciplines. For example, in [69] we used graphs to model games. The reader can find a comprehensive and modern introduction to graphs in [70]. In the following paragraphs we introduce fundamental notions of Graph Theory.

Definition II.1 (graph)**.** In its simplest form, a *graph* is a tuple $\mathcal{G} = \langle V, E \rangle$, where V is the countable set of *vertices*, and $E \subset V \times V$ is the set of *edges*.

Vertices represent objects, whereas edges represent relations between objects. In many cases, relations are symmetric and irreflexive, in which case it makes more sense that $E \subset \binom{V}{2}$. Here, the notation $\binom{V}{2}$ denotes the set of unordered pairs of vertices. When $E \subset \binom{V}{2}$, we say the graph is both *symmetric* (accounting for the fact edges are unordered pairs of vertices) and *simple* (accounting for the fact there is no edge comprising twice the same vertex). We call *neighbor* of u in $\mathcal{G}$ the set $\mathcal{N}(u) = \{v \in V, (u, v) \in E\}$.

The number of vertices in the graph, denoted $|V|$, is called the *order* of the graph. The *size* of the graph refers to the number of edges, $|E|$.

Definition II.2 (clique)**.** A clique in a graph is a subset of vertices from which any pair is an edge of the graph.

Sometimes graphs are best represented with their square *adjacency matrix* $\mathbf{W}$, indexed by vertices. The most used convention (although we discuss this point in [71]) is to define $\mathbf{W}$ as follows:

$$\mathbf{W}[u, v] = \begin{cases} 1 & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases} . \tag{II.2}$$

Since the adjacency matrix of the graph conveys exactly the same information as the set of edges, we sometimes consider graphs $\mathcal{G} = \langle V, W \rangle$.

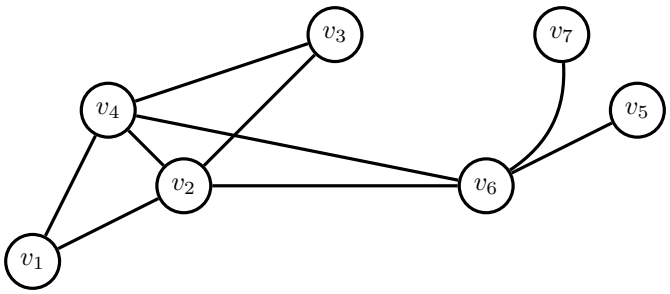


Figure II.1: Example of a simple and symmetric graph. Vertices are represented within circles and edges as lines connecting vertices.

In some cases, graphs can be *weighted*, as to capture more subtle relationships between their vertices. In that case the adjacency matrix contains real values, with the convention (again discussed in [71]) that $\mathbf{W}[u,v] = 0$ if and only if there is no edge (u,v) in the graph. The value $\mathbf{W}[u,v]$ is called the weight of the edge (u,v).

A *walk* in the graph is a sequence of vertices, such that any two consecutive vertices form an edge in the graph. It is said to be elementary if each vertex appears at most once (with the only possible exception being that the two extremities are identical). Walks are often confused with *paths*. A path is a sequence of edges, obtained by taking consecutive vertices in a walk, but such that all edges are distinct from one other.

Consider the simple and symmetric graph depicted in Figure II.1, where vertices are represented as circles and edges as lines between circles. In this example, $\{v_1, v_2\}$, $\{v_2, v_6\}, \{v_4, v_6\}, \{v_3, v_4\}$ is a path obtained from the walk v_1, v_2, v_6, v_4, v_3. But $\{v1, v2\}$, $\{v2, v4\}, \{v2, v4\}$ (obtained from the walk v_1, v_2, v_4, v_2) is not a path, because of the repetition of the edge $\{v_2, v_4\}$.

A *cycle* in a graph is a path in which extremities of the underlying walk are identical. For example, in the previous graph $\{v_2, v_6\}, \{v_4, v_6\}, \{v_3, v_4\}, \{v_2, v_3\}$ is a cycle.

The length of a path is the number of edges it contains. For weighted graphs, the length often refers to the sum of the weight of its edges. Using lengths it is possible to define the distance between two vertices in a graph. Let us first denote $P(u,v)$ the set of all paths connecting vertex u to vertex v (u and v are resp. starting and ending vertices of the corresponding walk). Given a path p, we denote $len(p)$ its length.

Definition II.3 (geodesic distance)**.** Given a graph $\mathcal{G} = \langle V, W \rangle$, the geodesic distance $d_{\mathcal{G}}$ is the function:

$$d_{\mathcal{G}} : \begin{cases} V^2 & \to & \mathbb{R} \cup \{+\infty\} \\ (u,v) & \mapsto & \min_{p \in P(u,v)} len(p) \end{cases} \tag{II.3}$$

A graph is said to be *connected* if there exists a path between any pair of vertices.

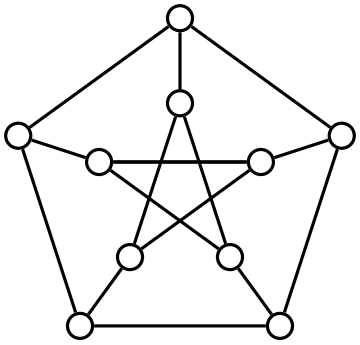


Figure II.2: Another example of a graph. In this case, it is the celebrated Petersen graph.

Equivalently, it means that the geodesic distance output domain is $\mathbb{R}$ (without $+\infty$).

Some graphs are particularly useful for proving interesting properties. For example *trees* are connected graphs that are cycle-free. A *complete* graph is a graph containing all possible edges. A *ring* graph is a graph which edges are precisely those that occur in a cycle obtained from an elementary walk that goes through all vertices of the graph[2]. A *bipartite* graph is a graph which unweighted version only contains cycles of even length. An Erdős-Rényi graph is a graph where each edge is drawn independently with probability p.

II.2.1.2 Signals on Graphs

Among other applications, graphs are well adapted to represent dependencies between dimensions of multivariate signals. Consider for instance observing a signal showing the neural activity in regions of interest in the brain. Disregarding how these regions may interact, this signal is basically a vector which indexation is arbitrary. Instead, it might be useful to exploit prior knowledge about how regions of interest interact with each other.

In the coming sections, we will see how to apply graph signal processing to the study of neural networks. But for now, let us introduce basic concepts of the field.

In its general form, Graph Signal Processing is a field that aims at extending harmonic analysis, including Fourier transform, convolution, translation, filtering... to ad hoc domains described with graphs.

Let us consider a graph $\mathcal{G} = \langle V, W \rangle$. We call *signal* a real-valued vector $\mathbf{x}$, indexed by V.

Using simple linear algebra, signals can interact with graphs. For example, it is possible to *diffuse* the signal $\mathbf{x}$ over the graph $\mathcal{G}$. It simply consists in constructing

[2] I apologize to the reader for this very non-intuitive definition of ring graphs.

the signal $\mathbf{x}' = \mathbf{W}^n\mathbf{x}$, where n is typically a nonnegative integer (or a nonnegative real number). To understand why we call this operation a diffusion requires to look at its details. Indeed, let us focus on a specific vertex u. It holds that:

$$\mathbf{x}'[u] = \sum_{v \in V} \mathbf{W}^n[u, v]\mathbf{x}[v]. \tag{II.4}$$

In other words, the value at vertex u in the diffused signal is a linear combination of the values in its neighbors in the original signal, when $n = 1$, hence the term "diffusion"[3].

Often we look at diffusion in the context of normalized adjacency matrices. There are many ways to normalize the symmetric matrix, and pros and cons of each method are heavily discussed in the literature [72]. Just to name a few, random-walk normalization consists in making the matrix right-stochastic. To do so we consider $\mathbf{A} = \mathbf{D}^{-1}\mathbf{W}$ instead of $\mathbf{W}$, where:

$$\mathbf{D}[u] = \sum_{v \in V} \mathbf{W}[u, v]. \tag{II.5}$$

Other authors prefer to use the (standard) normalization $\mathbf{A} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.

When dealing with multiplications in linear algebra, it is often meaningful to look at the spectrum of the matrices. Let us consider a symmetric graph $\mathcal{G}$. Being symmetric means that $\mathbf{W}$ is also symmetric. Now, $\mathbf{W}$ real-valued and symmetric means that it can be written as:

$$\mathbf{W} = \mathbf{F}\mathbf{\Lambda}\mathbf{F}^\top, \tag{II.6}$$

where $\mathbf{F}$ is an orthonormal matrix, $\mathbf{F}^\top$ is its transposed version, and $\mathbf{\Lambda}$ is a diagonal matrix which diagonal is made of the eigenvalues of $\mathbf{W}$, in descending order: $\mathbf{\Lambda}[i, i] = \lambda[i], \forall i$. In [29], we analyzed the spectrum of random small-world graphs. We derived closed-form expressions and concentration bounds.

Interestingly, normalized versions of the adjacency matrix have the interest of being such that:

Proposition: 1. *The normalized adjacency matrix of a symmetric graph is such that its largest eigenvalue is exactly 1, and the corresponding eigenvectors are constant.*

Proposition: 2. *The normalized adjacency matrix of a symmetric graph is such that its lowest eigenvalue is at least -1, and this lower bound is only achieved if the graph is bipartite.*

For details and proofs about these statements, the reader can refer to the book chapter we wrote in [21].

[3] Note that the term diffusion actually refers to the study of the "heat diffusion".

If the adjacency matrix of the considered graph has the property of being *circulant*, an interesting consequence is that $\mathbf{F}$ can take the form of a standard discrete Fourier transform. This is in particular the case for a ring graph, for the correct indexation of vertices. Extrapolating this identity, authors [73] define the Graph Fourier Transform (GFT):

Definition II.4 (graph Fourier transform)**.** Given a graph $\mathcal{G} = \langle V, W \rangle$ and a signal $\mathbf{x}$. The GFT $\hat{\mathbf{x}}$ of $\mathbf{x}$ on $\mathcal{G}$ is defined as:

$$\hat{\mathbf{x}} \triangleq \mathbf{F}^\top \mathbf{x}. \tag{II.7}$$

The (possibly normalized) adjacency matrix of a graph allows to define another operator, that is called the Laplacian of the graph. The Laplacian of the graph is defined as the matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Other versions of the Laplacian, including the random walk Laplacian or the normalized Laplacian, are obtained using the same normalization as described for the adjacency matrices.

Note that diffusion can be written as:

$$\mathbf{x}' = \mathbf{F} \sum_{i=1}^{|V|} \left(\lambda_i^n \mathbf{F}^\top[i,:]\mathbf{x} \right) = \mathbf{F} \sum_{i=1}^{|V|} \left(\lambda_i^n \hat{\mathbf{x}}[i] \right). \tag{II.8}$$

Together with Proposition 1 and Proposition 2, this writing has the interest of showing that, on a nonbipartite graph, diffusion has the effect of *smoothing* the signal on the graph, in the sense that neighboring vertices have more similar values after diffusion than they had before diffusion. As a matter of fact, when diffusing the signal, only its constant part is fully maintained (as it is associated with eigenvalue 1, as stated in Proposition 1), whereas other parts are weakened (as their corresponding eigenvalues are strictly between -1 and 1, as a consequence of Proposition 1 and Proposition 2).

More generally, we can define the smoothness of the signal on a graph.

Definition II.5 (graph smoothness)**.** Given a graph $\mathcal{G} = \langle V, W \rangle$ and a signal $\mathbf{x}$, the smoothness of $\mathbf{x}$ over $\mathcal{G}$ is the quantity:

$$s_{\mathcal{G}}(\mathbf{x}) = \mathbf{x}^\top L \mathbf{x} = \hat{\mathbf{x}}^\top (\mathbf{I} - \Lambda) \hat{\mathbf{x}} = \sum_{i=1}^{|V|} (1 - \lambda_i) \hat{\mathbf{x}}[i]^2 = \sum_{u,v \in V} \mathbf{W}[u,v] \left(\mathbf{x}[u] - \mathbf{x}[v] \right)^2, \tag{II.9}$$

where $\mathbf{I}$ is the identity matrix.

As such, smoothness is 0 when the signal values of strongly connected vertices is identical, and positive otherwise.

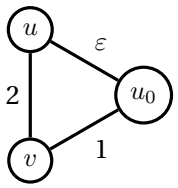


Figure II.3: Example weighted graph where one edge weight is parametrized by ε.

II.2.1.3 Uncertainty

A fundamental result of classical signal processing is that a signal cannot be localized in both space and frequency domains. This result, often referred to as the "uncertainty principle" [74], has strong implications to the field. Following the seminal work of [75, 76], we published several works extending this principle to graph signals.

In [75, 76], the authors define two notions:

Definition II.6 (graph spread)**.** The graph spread of a signal $\mathbf{x}$ on a graph $\mathcal{G}$, around a vertex u_0, is:

$$\Delta^2_{\mathcal{G},u_0}(\mathbf{x}) \triangleq \frac{1}{\|\mathbf{x}\|_2^2} \sum_{u \in V} d^2_{\mathcal{G}}(u_0, u)\mathbf{x}[u]^2. \tag{II.10}$$

Definition II.7 (spectral spread)**.** The spectral spread of a signal $\mathbf{x}$ on a graph $\mathcal{G}$ is:

$$\Delta^2_s(\mathbf{x}) \triangleq \frac{1}{\|\mathbf{x}\|_2^2} s_{\mathcal{G}}(\mathbf{x}). \tag{II.11}$$

They then show that on some graphs, couples $(\Delta^2_{\mathcal{G},u_0}(\mathbf{x}), \Delta^2_s(\mathbf{x}))$, for unit-norm signals, cannot be made arbitrarily close to 0. Said otherwise, there exists a trade-off between how graph spread and spectral spread can be made small. For some families of graphs, they are even able to obtain a closed form expression of the lower curve of the region $(\Delta^2_{\mathcal{G},u_0}(\mathbf{x}), \Delta^2_s(\mathbf{x}))$. In [77], we extended this mathematical formulation to other families of graphs.

In another paper [27], we adapted the definitions proposed in [75, 76] to make them compliant with weighted graphs. As a matter of fact, when used as is, the problem is that the use of the geodesic distance in the definition of the graph spread causes strong discontinuities. Consider the example graph of Figure II.3. We have $\lim_{\varepsilon \to 0} (d_{\mathcal{G}}(u_0, u)) = 0 \neq 3 = d_{\lim_{\varepsilon \to 0} \mathcal{G}}(u, u_0)$, leading to similar discontinuities for the corresponding values of the graph spread.

The reason for this problem comes from an incoherent interpretation of edge weights. As a matter of fact, in Definition II.6, the graph spread increases with distances in the

graph, meaning that edge weights are interpreted as "distances" between vertices. On the contrary, in Definition II.5, a signal is smooth if large edge weights link vertices with similar values, so edge weights represent "similarities". To overcome this contradiction, we introduced in [27] a transform on edge weights so that the geodesic distance can be applied meaningfully to weighted graphs.

Later, in [21], we introduced new definitions of graph and spectral spreads, in an effort to make a closer connection with classical signal processing. Namely, we removed the squared distance in the definition of the graph spread, and the dependence on a specific central vertex u_0. We obtained the following definition:

Definition II.8 (redefined graph spread)**.** The redefined graph spread of a signal $\mathbf{x}$ on a graph $\mathcal{G}$, around a vertex u_0, is:

$$\Delta_{\mathcal{G}}(\mathbf{x}) \triangleq \frac{1}{\|\mathbf{x}\|_2^2} \inf_{u_0 \in V} \left[\sum_{u \in V} d_{\mathcal{G}}(u_0, u) \mathbf{x}[u]^2 \right]. \tag{II.12}$$

Similarly, we wanted the spectral spread to better reflect the idea of a smooth variation, rather than forcing most components of the Fourier transform of the signal to be 0. We proposed the following definition:

Definition II.9 (redefined spectral spread)**.** The redefined spectral spread of a signal $\mathbf{x}$ on a graph $\mathcal{G}$ is:

$$\Delta_s(\mathbf{x}) \triangleq \frac{1}{\|\mathbf{x}\|_2^2} \inf_{i=1}^{|V|} \left[\sum_{j=1}^{|V|} \|\lambda_i - \lambda_j\|_2 \hat{\mathbf{x}}_j^2 \right]. \tag{II.13}$$

Despite these new definitions being more challenging to deal with, due to the presence of a $\inf$ operator, we managed to use them to prove the existence of an uncertainty principle on weighted graphs.

Among other results, the existence of an uncertainty principle has the interest of showing that having limited observations in the graph domain (resp. in the spectral domain) lead to a strong uncertainty in the spectral domain (resp. in the graph domain). This result is strongly tied with the graph inference problem we discuss in Section II.2.2.1.

II.2.1.4 Translations

Graph Signal Processing can be used to define convolutions by analogy with classical Fourier analysis. As a matter of fact, convolutions are simple pointwise multiplication in the spectrum domain in classical signal processing. By extension, it is possible to obtain counterparts of convolutions for complex domains represented with graphs.

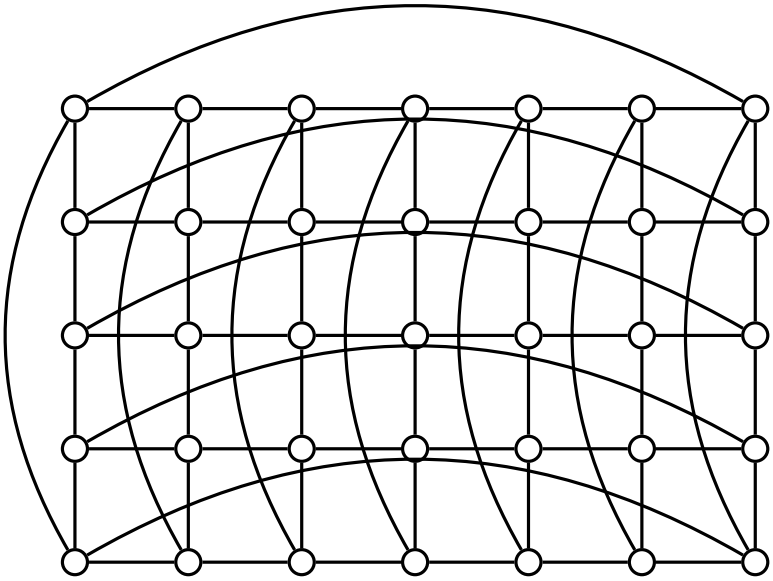


Figure II.4: Example of a torus grid graph.

Following this lead, translations can be obtained by convolving with Dirac signals (one-hot-bit vectors), again following the analogy with classical signal processing.

Interestingly, when starting with a ring graph, both convolutions and translations match exactly their definitions in the time domain. This is because the adjacency matrix can be made circulant and thus its decomposition matches perfectly the classical Fourier transform. But when the input graph is more complex the analogy does not guarantee exact matches with usual conventions.

A trivial example is obtained when considering a torus grid graph, as depicted in Figure II.4. In general, torus grid graphs are defined by considering the set of vertices to be $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$, and edges connect vertices (u, v) and (u', v') if and only if $u = u' \wedge |v - v'| \equiv 1 \pmod{m}$ or $v = v' \wedge |u - u'| \equiv 1 \pmod{n}$.

Torus grid graphs are natural graph representations of regular 2D structures. As such, 2D Fourier transform, convolutions and translations are well defined. But using Graph Signal Processing the obtained operators are very different from their classical counterparts. Consider the example depicted in Figure II.5. Here we consider an image signal on a torus grid graph. In (a) is depicted the original image, in (b) its representation on the graph, and in (c) and (d) two arbitrary translations defined using the Graph Fourier Transform. It is quite clear from these examples how the translations defined this way do not match the classical translations in a 2D domain.

The reason for this shortcoming is that the Graph Fourier Transform is defined with

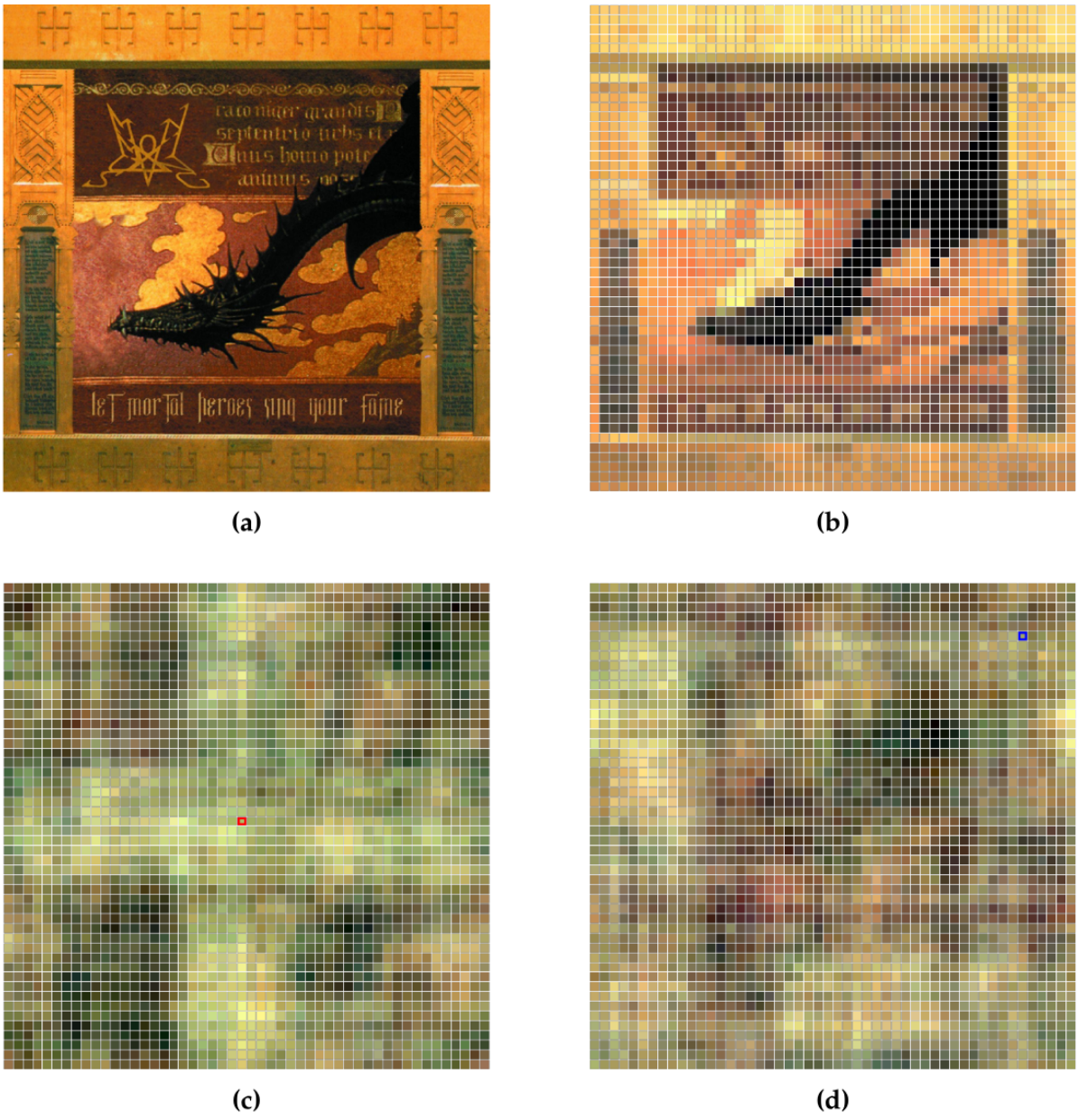


Figure II.5: Example of the translation obtained with a torus grid graph. (a) is the original image, (b) the image on the torus grid graph and (c) and (d) are two arbitrary translations using the Graph Fourier Transform of the torus grid graph. This image is extracted from [78].

analogy to a 1D ring graph. When applied to complex topologies, the result is that the Graph Fourier Transform adapts the 1D transform to this domain. In the case of 2D graphs, translations have the effect of smoothing the signal while translating it.

For this reason, we developed alternative ways to define translations on graph structures. In [79], we proved that it is possible to retrieve classical translations from torus grid graphs by considering the group of functions generated from an initial set of transforms. This initial set of transform is characterized by:

1. Functions should be from vertices to vertices,
2. Functions should be associating a vertex with one of its neighbor,
3. Functions should associate neighbor vertices with neighbor vertices, and inverse functions should do so as well.

Not only does this paper shows that the structure of a torus grid graph is informative enough to recover the disregarded coordinates of the vertices, but it shows that in general it is possible to define translations on graphs directly in the vertex domain (not requiring to go through the Graph Fourier Transform).

II.2.1.5 Tropical Algebra

An obvious shortcoming of Graph Signal Processing is that it is based on the assumption that diffusion (as defined in Section II.2.1.2) makes sense for the considered problem.

But in many cases, multiplying the adjacency matrix of the graph with a signal is meaningless. Consider for instance the case of a communication network, where edge weights represent delays. In such an example, graphs signals could be observations of timestamps at which a given message was received at the various nodes of the communication network. In such a setting, diffusion would imply the multiplication of delays and timestamps, what is physically meaningless.

To overcome this lock, we proposed in [71] a generalization of Graph Signal Processing to semiring structures, such as the tropical algebra. We proposed novel definitions of diffusion, the Laplacian operator (which cannot be a matrix with semiring structures), and even an uncertainty principle for this generalized framework. For example, in Figure II.6, we show the evolution of the errors in reconstructing graphs from random smooth signals, a typical problem in the classical graph signal processing framework [80].

II.2.2 Graph Inference

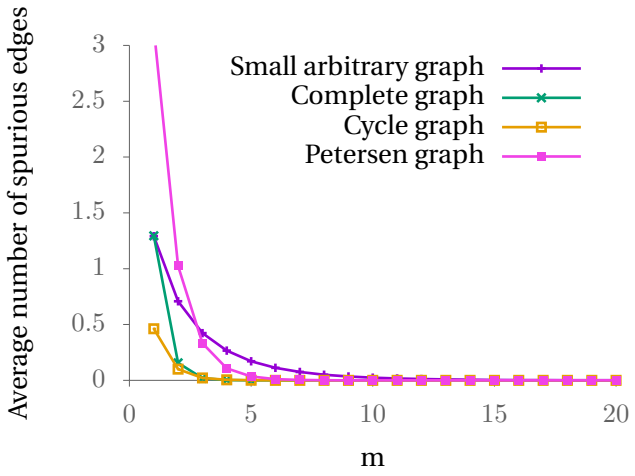


Figure II.6: Average number of spurious edges when reconstructing a graph from random smooth signals, as a function of the number of observations m.

II.2.2.1 From Signals

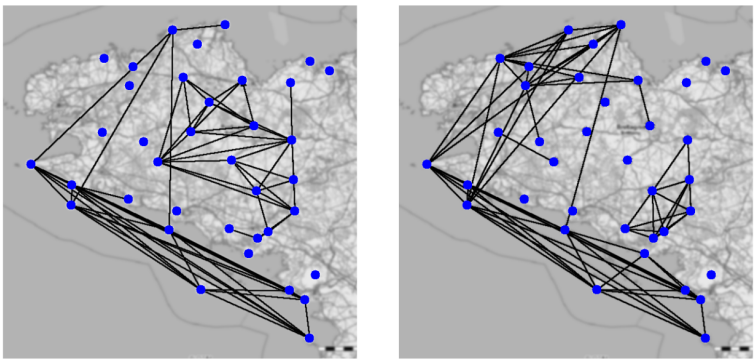
Another strong important shortcoming of Graph Signal Processing is that the framework supposes an access to a graph representing the relations between signal variables. In many cases, some of which shall be discussed in Section II.4.3.2, such a graph is not explicitly available. When that is the case, it becomes interesting to look at the problem of inferring a graph structure from a set of signals.

There are many ways to obtain a graphs from a set of signals. Some popular methods include covariance selection [81], thresholding the empirical covariance matrix [82], or enforcing specific relationships between recovered graphs and signals [83, 84, 80]. In [26], we decided to tackle the problem under the angle of diffused signals.

In our work, we considered the following paradigm. Starting from some white distribution, random signals are diffused using the adjacency matrix of a graph. So that we observe only the diffused version of these signals. Interestingly, it is straightforward to show that in such a case, the empirical covariance matrix of observed signals is a polynomial of the adjacency matrix of the graph. As such, the eigenvectors of the graph are directly estimated using the eigenvectors of the empirical covariance matrix.

The problem then boils down to estimating the (lost) eigenvalues of the graph. We proposed an optimization procedure trying to maximize sparsity in the reconstructed graph, as sparsity is often a strong asset of downstream methods. This work motivated for the introduction of new approaches to the problem [85].

In [23], we extended our framework to the more general case of stationary signals. Stationary signals are signals which covariance matrix commutes with the graph ad-



(a) (b)

Figure II.7: Example of two inferred graphs from temperature sensors distributed in the Brittany region. On the example of the left, the graphs was recovered assuming signals were smooth. On the example of the right, we combined smoothness with stationarity using our proposed method. Interestingly, we observe that the graph on the right is better at uncovering regularity in measurements along the seashore.

jacency matrix. Again, this means that the eigenvectors of the covariance matrix of signals are the same as the ones of the graph adjacency matrix. This equality introduces a polytope of possible solutions. We introduced several optimization methods to enforce various properties of recovered graphs, including sparsity, simplicity and smoothness. We also performed experiments on both synthesized datasets and real data. For example in Figure II.7, we compare the inferred graphs using two methods. In case (a), we used the method described in [80], which assumes that signals are smooth on the graph. In case (b), we combined this method with ours, to enforce both smoothness and stationarity. As a result, the graph on the right shows better regularity along the seashore.

II.2.2.2 From Traces

Sometimes, recovering a hidden graph structure can be performed thanks to a collection of indirect observations. In [33], we considered the case where we observe sets of vertices corresponding to short walks in symmetric graphs. Observing the walks directly could be very informative about the edges of the graph. But observing unordered sets of vertices, that we call *traces*, carries much more ambiguity. This problem makes sense for applications where, for example, we observe the vertices a packet traversing the graph has encountered, but we do not have access to the order in which vertices have been encountered.

The problem is trivial when traces are made of pairs of vertices, since traces convey exactly as much information as edges. Interestingly, we were able to determine neces-

sary and sufficient conditions for an edge to be correctly retrieved, or for a nonedge to be unconsidered, when observing traces of three vertices. We also derived probabilistic formulations in the case of traces generated uniformly at random in Erdős-Rényi graphs.

II.2.3 Direct Applications

II.2.3.1 Time-varying Graph Signals

In [7], we tackled the problem of predicting the future of time-varying graph signals using latent spatial representations. The main question we asked in the paper was whether using a prior about the structure using Graph Signal Processing, or learning directly from scratch using autoencoders would lead to the best predictions.

To this end, we introduced two new datasets. The first one was created using the STL10 image dataset, that comprises 96x96 pixels images of natural objects. We extracted short sequences of images from STL10 by selecting an initial crop of the image and moving the crop towards a unique direction frame by frame. The second one was created from fMRI measurements of resting state subjects.

In both cases, we compared three alternatives:

1. A completely unlearned representation. Here we used a grid graph for the images and a structural connectivity graph for the fMRI dataset. Then we used the low spectrum of the Graph Fourier Transform as latent features,

2. A completely structure-agnostic learnable method. Here, we used an autoencoder that completely disregarded any disposable prior about the structure of signals,

3. A mixed case, where the idea here was to infer a graph from the signals (c.f. Section II.2.2.1) and to use this graph to obtain latent representations through the low spectrum of the Graph Fourier Transform.

To make the comparison fair, the autoencoder was constrained to adapt the same mathematical formulation as the Graph Fourier Transform.

We then used the latent representations to predict the future of sequences, through the use of a Long Short-Term Memory (LSTM) neural network [86, 87].

We found that in the end-to-end task of predicting sequences, all methods performed similarly, whereas the autoencoder was able to extract more efficient latent spatial representations of the input. We concluded that the LSTM were probably expressive enough so as to compensate for the poorer representation of graph-based methods. In Figure II.8 we depict examples of sequence predictions with the different

methods: GFT refers to the use of the grid graph, GEO to a grid graph with inferred weights and AE to the use of the structure-agnostic autoencoder.

II.2.3.2 Neuroimaging Data

A previously mentioned promising application domain for the tools and methods developed using Graph Signal Processing is that of neuroimaging. As a matter of fact, there has been numerous recent publications in the field showing that graphs can be very useful to analyze these signals [88, 89, 90].

In [5], we looked at the possibility to use Graph Fourier Transforms to improve the accuracy of classifiers using fMRI inputs. We considered the Haxby dataset [91] where measurements were performed while subjects were looking at objects on a screen, which objects could belong to 7 possible distinct categories. The challenge here was to guess what category the image shown on the screen belonged to based only on the fMRI measurements.

We considered various graphs inferred from the available data, and various ways to select components of the Graph Fourier Transforms. Overall, we demonstrated the effectiveness of Graph Fourier Transform to improve the accuracy of classification in this specific context.

II.2.3.3 Bio-inspired Visual Processing

In [92, 93], we looked at the possibility to use graphs to define image filters inspired by the human retina. As a matter of fact, most computer vision methods treat images as matrices of pixel values, whereas the retina has a uneven distribution of sensors with various spectral responses. This phenomenon, referred to as “cortignal magnification” in the literature, has important consequences to how humans analyse and process visual inputs.

We proposed a mathematical model and a Python implementation of a retina-inspired framework for processing visual signals.

In [92], we demonstrated the interest the proposed framework in order to increase the performance of saliency maps estimators. A saliency map is a map obtained using eye-trackers on humans while they observe images on a screen. It estimates the probability distribution of an eye fixation on each pixel. In Figure II.9 we depict examples of saliency maps obtained using our framework. At the time of releasing this work, the proposed model of visual attention was ranked first on the celebrated MIT Saliency Benchmark. It remained first among the models that did not use training data for several years.

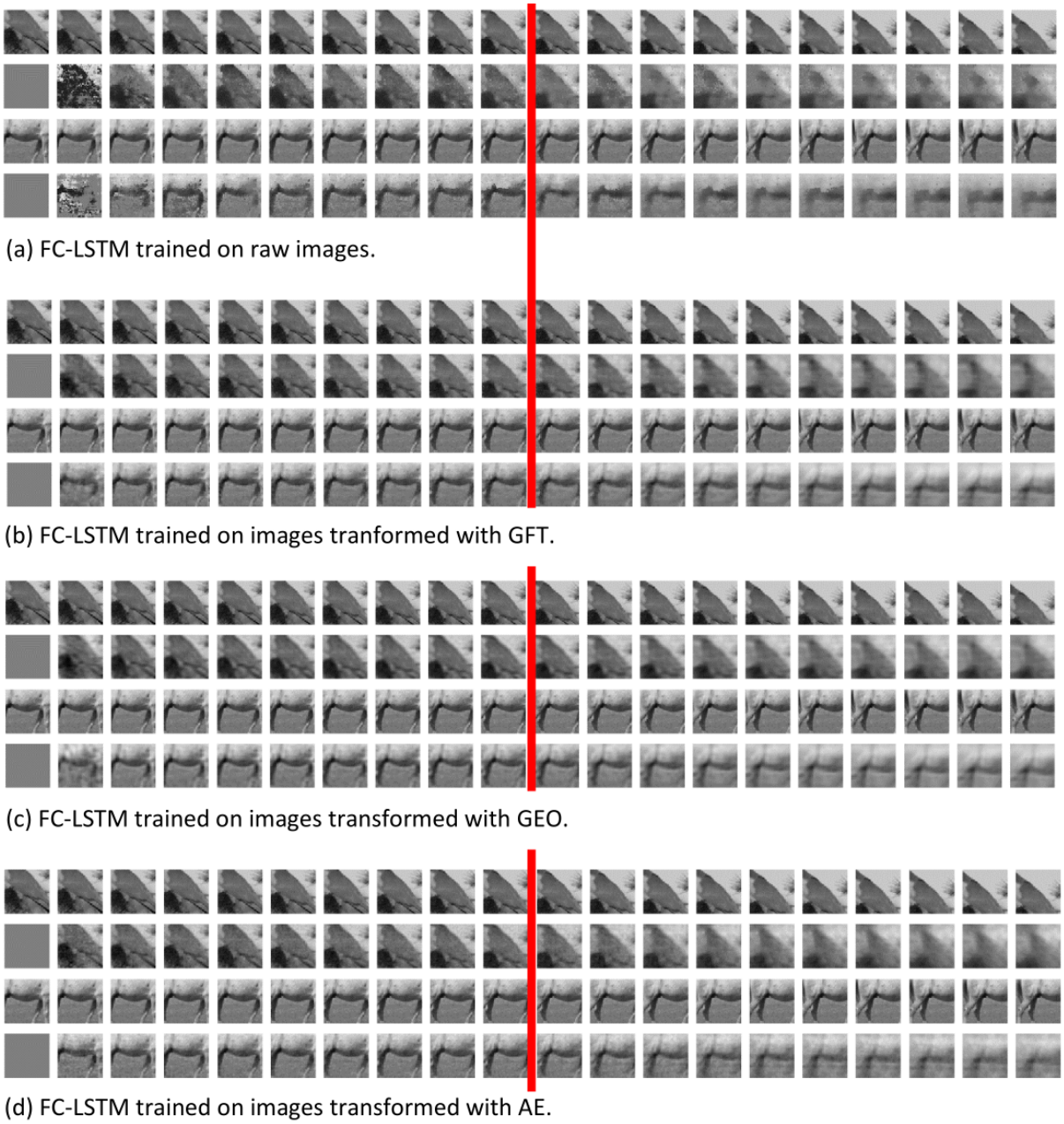


Figure II.8: Example of prediction of the future of image sequences using various latent spatial representations of the inputs.

Figure II.9: Saliency maps extracted using the retina-inspired framework introduced in [93].

II.2.3.4 Graph Matching

As previously mentioned, the use of brain networks is becoming increasingly popular in the literature [94, 95]. Among other problems, a key question of the field is the ability to efficiently compare graphs between subjects and/or measurements.

The problem of graph matching is very classical in computer science [59]. But in the specific case of neuroimaging data, the problem is quite different. As a matter of fact, graph similarity usually refers to the question of comparing two graphs for which vertices are anonimized. In the context of neuroimaging data, additional information is available under the form of GPS coordinates of vertices representing regions of interest in the brain.

Of course in some cases these regions of interest are obtained independently from the considered data, and are shared among subjects and experiments, in which case the graph similarity problem boils down to edit distances or a simple Frobenius norm. But when regions of interest are evaluated using the processed data, it is likely that two graphs will have distinct structures and distinct vertices coordinates.

In this specific context of matching graphs taking into account both the structure of the graph (i.e. the edges) and the coordinates of the vertices, we authored a number of publications introducing tools and experiments [4, 3, 2].

II.2.4 Summary of Contributions of the Section

In the field of graphs and graph signals, the main contributions are:

1. Fundamental questions:
   (a) Contributing to the definition of an uncertainty principle for graph signals [27, 77, 21],
   (b) Proposing alternate definitions for translations of graph signals [79, 96],
   (c) Contributing to the field of graph inference [26, 23, 33],
   (d) Extending Graph Signal Processing to semiring structures [71].
2. Applications:
   (a) Time-varying graph signals [7],
   (b) Neuroimaging data [5, 4, 3, 2],
   (c) Retina-inspired visual processing systems [93, 92],
   (d) Neuroimage-based graphs matching [4, 3, 2].

II.3 Neural Networks for Indexing

II.3.1 Source and Channel Coding

II.3.1.1 Source Coding

In the field of information theory, two very important concepts are that of source and channel coding. Source coding usually refers to the compression of data issued from a random source. By exploiting redundancy of the source through its potentially nonuniform distribution, source coding has seen the emergence of numerous methods deployed in modern digital systems. In [34], we contributed to the field by looking at the problem of compressing multisets.

Multisets are sets in which elements can be repeated. They naturally occur in many application domains, including the bag-of-words representations of data. Contrary to sequences of elements, multisets completely disregard the order of elements, which can be seen as a form of information reduction.

As a matter of fact, consider m messages, containing n bits each, generated uniformly at random using the same source S. When order in which these messages have been generated is important, the problem becomes equivalent to compressing a sequence of mn bits generated uniformly at random. Shannon's source coding theorem gives that any lossless compressed version of this sequence must contain at least H bits, where H is the corresponding entropy, which is also mn. In other words, one cannot expect to (losslessly) compress such data.

Now, consider that the order in which messages have been generated is unimportant. In the extreme case where the source generates only one message containing a total of mn bits, we obtain the same result that no compression can be achieved. In the other extreme case where the source generates mn messages of 1 bit each, then it is sufficient to store the number of times each bit has been seen, which can be performed using $\log_2(mn)$ bits. More generally, the two question are: what gains can be expected in nontrivial settings? How to achieve them?

Authors considered in the past two settings [97, 98, 99, 100]: one where $m \ll 2^n$, meaning that repetitions are very unlikely to occur. In this setting, the problem boils down to that of compressing sets (without repetition). The other where $m \gg 2^n$, in which case the problem boils down to the compression of a histogram.

In our work, we focused on the intermediate case where $m = \Omega(2^n)$. We first derived bounds based on the entropy of the source, then derived an algorithm using prefix trees (tries) to approach the bound by a constant factor of 5/3. An illustrative example of a trie based on a small set of vectors is depicted in Figure II.10.

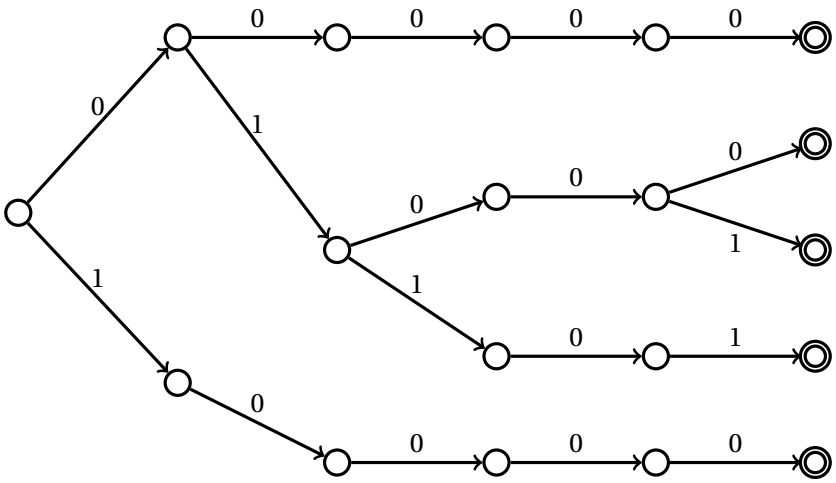


Figure II.10: Trie associated with the set of vectors $\{00000, 01000, 10000, 01001, 01101\}$.

II.3.1.2 Channel Coding

Another important problem in the field of Information Theory is that of Channel Coding. Usually, channel coding refers to the use of well-designed representations of data, that are able to suffer radius-limited distortions without loss of information.

The key principle behind channel coding is simple. The idea is to embed data in a large dimensional space in which any two embeddings are distant apart. As such, any perturbations of at most half the minimum distance between two such embeddings can be mitigated. These embeddings are called *error correcting codes*. In [35], we looked at the possibility to define error correcting codes by using cliques in graphs.

Let us develop on that. Imagine again being in the context of facing bag-of-words data. So basically, data is made of subsets of a fixed set of features, and is represented through indicator vectors. For the sake of simplicity, consider that each subset is made of the same number of elements.

In this case, the problem can be formalized as follows: let n be the number of features, and consider binary ($\{0, 1\}$) vectors of size n containing the same number c of 1s (we call them feature vectors). Without error correcting codes, these vectors are susceptible to the erasure of a 1 or to the creation of a spurious 1. In both cases, multiple feature vectors can be equidistant to the created erroneous vector. An example is given here: consider $n = 5$, and the feature vectors 11100 and 01110, here written as sequences. Let us start with the feature vector 11100, and imagine that the first 1 is erased. We obtain the vector 01100, which could result from an erasure in 11100 or from an erasure in 01100.

Instead, consider the feature vector dimensions as being vertices of an empty graph

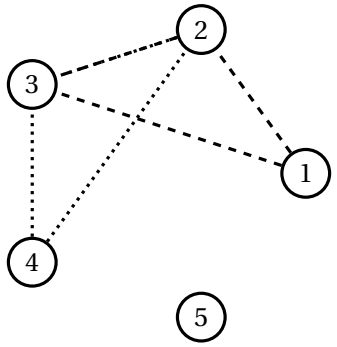


Figure II.11: Example of a graph generated from feature vectors 11100 and 01110. For better readability, edges are represented differently depending on whether they were added when considering 11100 or 01110, but the graph structure completely disregards this information.

(i.e. a graph with no edge). Then for each feature vector, add the required edges so that each pair of vertices with signal value 1 are connected through an edge. Now, using edges instead of vertices to represent feature vectors, the deletion of an edge or the creation of an edge does not create ambiguities. This is depicted for the same example in Figure II.11. We can see two cliques of order 3 in the graph. One connecting vertices 1, 2 and 3, and corresponding to feature vector 11100, and the other connecting vertices 2, 3 and 4, and corresponding to feature vector 01110.

Indexing edges in the lexicographic order, we obtain representations 1100100000 and 0000110100. Interestingly, the erasure of any 1 in the obtained representation does not cause ambiguity anymore: cliques can be used as error correcting codes. This idea, that we developed theoretically in [35], is at the core of all results in Section II.3.2.

II.3.1.3 Error Correcting Codes for Neural Networks

Error correcting codes have been used in the context of neural networks. For example, the domain of Error Correcting Output Coding [101] is interested in designing outputs of neural network architectures that can benefit from the robust properties of error correcting codes.

In [10], we introduced assembly codes for learning neural networks. The idea is the following: usually, one one-hot-bit vector is used to represent the output of a network for a given input, where the nonzero bit indicates the corresponding class. This correspond to choosing $\binom{c}{1}$ output dimensions where c is the number of classes in the considered problem.

Instead, we can consider $\binom{c}{k}$ output dimensions. These combinatorial factor can be thought of as listing all binary vectors containing exactly k 1s. So, when processing an input of class i, all output dimensions where the corresponding vector contains a 1 at

position i would be set to 1 and the others to 0. The resulting error correcting code was proven in [10] to be powerful. By doing so, the decision of the network can be robust even if some dimensions are wrong. We proved significant gains in performance on toy datasets.

In [102], we introduced the idea of using error correcting codes to improve the robustness of Hopfield neural networks. Hopfield neural networks are associative memories that projects data to itself. By projecting data to random vectors (which are known to be powerful error correcting codes), we were able to significantly increase the robustness of the storing process. We developed later a similar idea in the context of sparse associative memories [37], the latter being the core of the next section.

II.3.2 Associative Memories

II.3.2.1 Sparse Associative Memories

In the context of storing bag-of-words feature vectors in neural networks for indexing, sparse associative memories have long been the reference [103].

Sparse associative memories are built from sparse vectors. So let us introduce some parameters. We consider here binary vectors ($\{0,1\}$) of dimension n, each containing exactly c 1s. We are given a set $\mathcal{M}$ of such vectors that we typically consider to be uniformly distributed. Of course in practice, and in particular if these vectors are bag-of-words representations, it is expected that the distribution can be highly nonuniform, what we shall cover later in this document.

The classical model introduced by Willshaw in 1969 [103] proposes to store these vectors as cliques in a graph, exactly the same way we described in Section II.3.1.2. This can be mathematically formalized as follows: we create a graph $\mathcal{G} = \langle V, W \rangle$, where $V = \{1, \ldots, n\}$ and:

$$W \triangleq \max_{\mathbf{x} \in \mathcal{M}} \mathbf{x}\mathbf{x}^\top. \tag{II.14}$$

Note that here the $\max$ operator is treated componentwise.

Now to solve the exact match problem on a query vector $\mathbf{x}$, a test procedure can be implemented, which consists in computing the integer:

$$C(\mathbf{x}) = \mathbf{x}^\top W \mathbf{x}. \tag{II.15}$$

As a matter of fact, $C(\mathbf{x})$ is the number of (nonsymmetric) edges between vertices corresponding to 1s in $\mathbf{x}$. If $\mathbf{x} \in \mathcal{M}$, then we know that $C(\mathbf{x}) = c^2$. The converse is not necessarily true, as edges forming the clique corresponding to $\mathbf{x}$ in $\mathcal{G}$ may have been added by several distinct elements of $\mathcal{M}$. There is thus an error probability when using

the test $C(\mathbf{x}) = c^2$ as a proxy for $\mathbf{x} \in \mathcal{M}$. However, when elements of $\mathcal{M}$ are uniformly drawn at random, it is expected that the error probability is small as long as cardinality of $\mathcal{M}$ remains small enough.

To solve the approximate match problem on a query vector $\tilde{\mathbf{x}}$, an iterative process can be performed, which consists in computing the sequence:

$$\begin{cases} \mathbf{x}^0 &= \tilde{\mathbf{x}}, \\ \mathbf{x}^{t+1} &= \mathbb{1}_{W\mathbf{x}^t \geq s}, \end{cases} \tag{II.16}$$

where s is a given threshold.

In [104, 105], we introduced block coding to classical sparse associative memories as a way to improve their storage abilities. The idea is to restrict vectors to be stored to be in $\{1, \ldots, \ell\}^c$. To store such messages, we first embed them into a binary space by converting each coordinate into the one-hot-bit encoded vector of dimension ℓ, then we use Equation (II.15). By doing so, we constrain the vectors to have specific forms, instead of the more generic expression of classical sparse associative memories. But we can then exploit this constraint when solving the approximate match problem for better performance. The idea of using block constraints is motivated by bio-inspiration, since neurons in the brain have the tendancy of being grouped in cortical microcolumns, with a winner-takes-all competition. If Figure II.12, we depict an example graph corresponding to a sparse associative memory following the block coding constraint.

In [106] we introduced a new iterative procedure, benefiting from the block constraint. It consists in computing the following sequence:

$$\begin{cases} \mathbf{x}^0 &= \tilde{\mathbf{x}}, \\ \mathbf{x}^{t+1}[\ell i + j] &= \sum_{i'=1}^{c} \max_{j=1}^{\ell} W[\ell i' + j', \ell i + j]\mathbf{x}^t[\ell i' + j']. \end{cases} \tag{II.17}$$

In [33], we proved this new family of associative memories is optimal in the sense of maximum-likelihood decoding. Then in [28] we introduced a formal description and analysis of these associative memories using an adhoc semiring structure. In the continuation of these works, we published in [107] an experimental comparison of various types of algorithms to solve the approximate match problem.

A few years later, we published in [46] a theoretical comparison of the various families of sparse associative memories. For the first time we were able to mathematically prove sharp bounds for these devices. If Figure II.13, we depict the performance in retrieving vectors when 4 of the initial 8 1s it contains are erased, depending on the

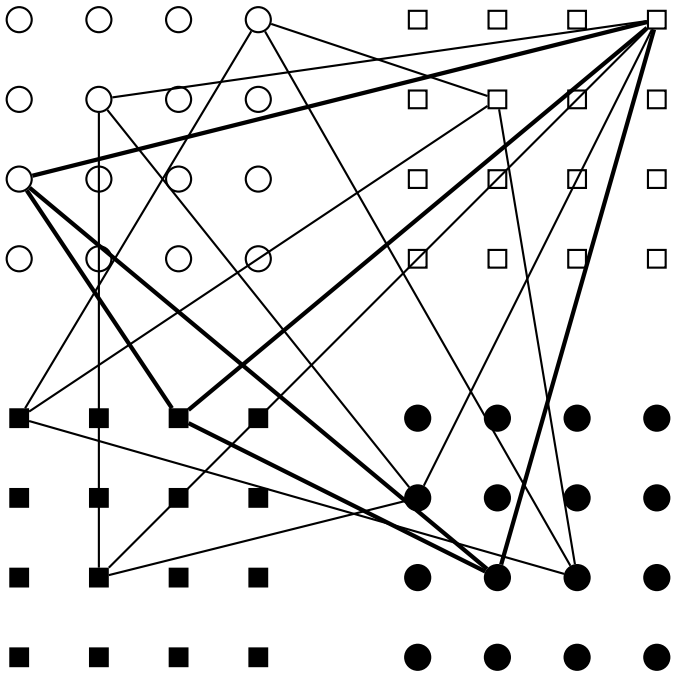


Figure II.12: Illustration of a graph corresponding to an associative memory obeying the block constraint. A vector correspond to a clique in the graph (e.g. thick edges) and connect one vertex from each block. In this example, $\ell = 16$ and $c = 4$.

number of stored vectors in the considered architectures. Are compared Amari, Willshaw and the block constrained models (GB) with the original and optimized retrieving procedure.

II.3.2.2 Extensions

In the continuation of these works, we developed several variations of the previously introduced model to cope with more diverse situations.

For example in [24], we introduced nonsymmetric graphs to store varying dimensions vectors. The idea is to exploit orientation of edges to traverse the graph multiple times during the retrieving procedure. In Figure II.14 we depict an illustration of this procedure.

In [108] we considered using weighted connections to incorporate more information during the storage process. In [109] we proposed an extension able to efficiently treat input queries with a lot of spurious ones. In [110] we considered the case of storing vectors with varying number of 1s.

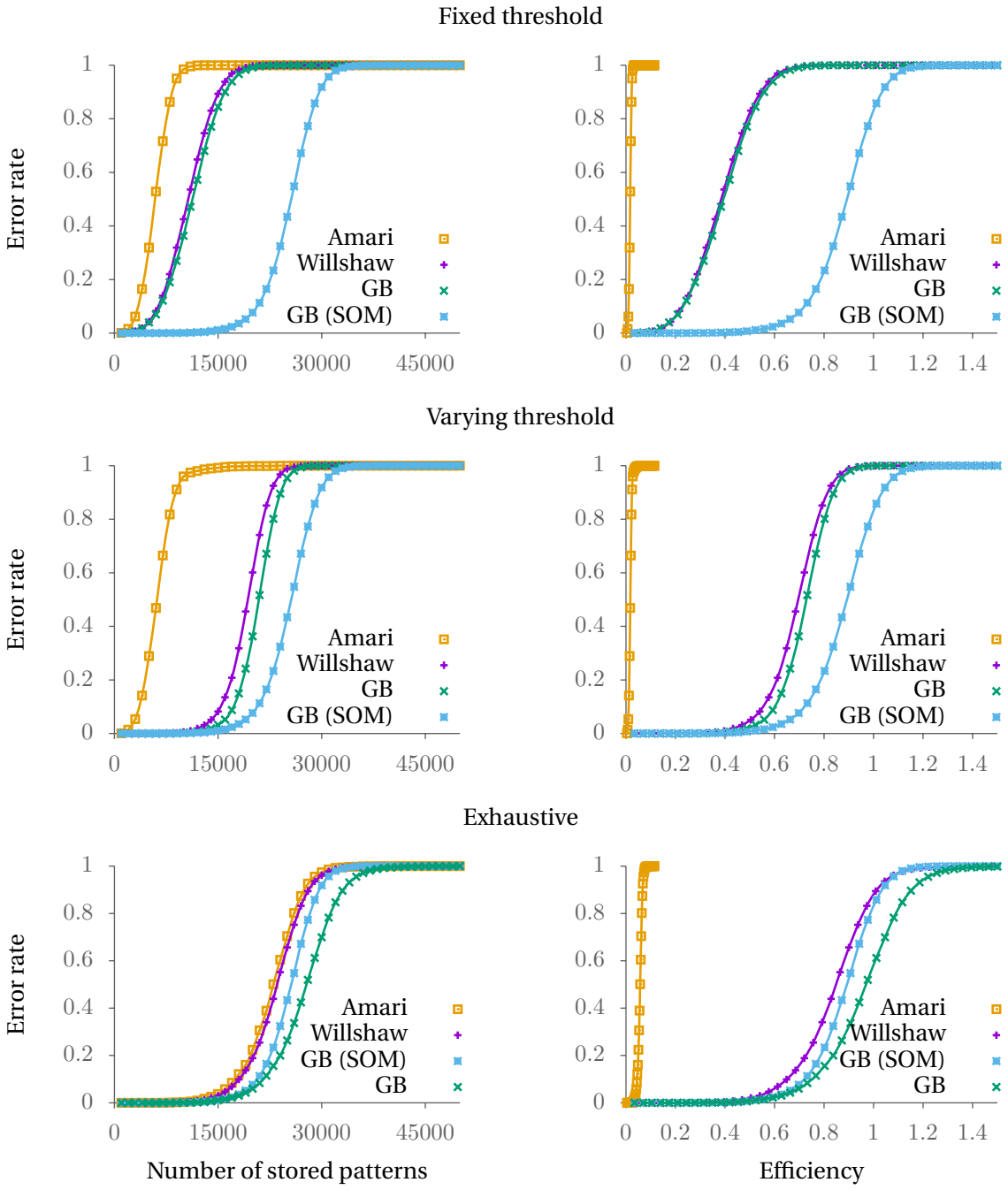


Figure II.13: Comparison of performance of Amari, Willshaw and GB (block constrained) models (with initial retrieving procedure and optimized one (called SUM-OF-MAX (SOM))). For each simulated point, there are $N = 2048$ vertices in the graph (grouped in $c = 8$ blocks of $l = 256$ vertices for the block constrained models), stored vectors contain exactly $c = 8$ 1s each and the objective is to retrieve a previously stored pattern when 4 out of the inital 8 1s are missing. Each point is the average of 100'000 tests. Figures in first column depict the evolution of the error rate as a function of the number of stored vectors. Figures in second column depicts the evolution of the error rate as a function of efficiency, defined by measuring the entropy of generated messages. First line correspond to fixed threshold dynamics, second line to varying threshold strategies and third line to exhaustive searches in the graphs.

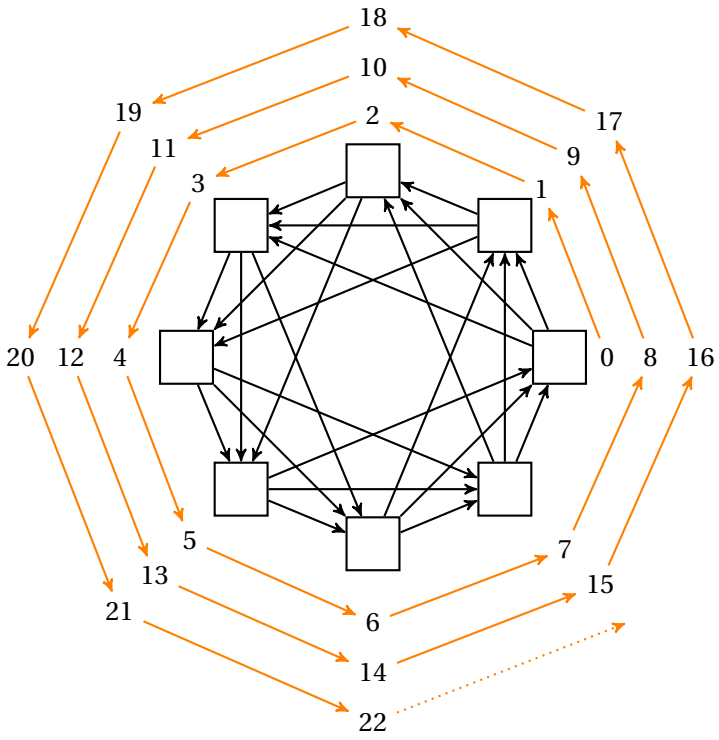


Figure II.14: Extension of sparse associative memories to dimension-varying vectors. Here squares represent blocks of the graph.

II.3.2.3 Implementations

A key motivation for the use of sparse associative memories is the fact the rely on binary graphs that can be implemented very efficiently on dedicated hardware. In a series of publications [44, 43, 40] we proposed efficient FPGA designs to implement sparse associative memories.

Since it mostly relies on matrix multiplication, the retrieving procedure can also benefits from Graphics Processing Units. In [9] we thus also designed a GPU-based solution with order of magnitude acceleration compared to a CPU implementation.

Another interesting aspect of sparse associative memories is that they are able to not only retrieve previously stored vectors when queries are approximate, but also when the graph storing the vectors is noisy. Motivated by this fact, we published in [25] a study of the fault tolerance of these associative memories. Interestingly, we observed that the information retrieval capabilities of these devices is very close to the information theoretic capacity.

II.3.3 Applications

II.3.3.1 Nonuniform Distributions

A key limitation of many results about associative memories is that they hypotheses the stored vectors are uniformly distributed. This assumption is known to favor most methods, where uneven distributions can have the effect of strongly biasing the retrieving procedure.

To overcome the effect of uneven distributions, we introduced several works modifying the graph structure. In these works, the core idea remains that overcoming uneven distributions requires to augment the size of the graph.

In [111], we introduced twin vertices. The idea is to identify vertices in the graph that are overloaded with neighbors, and to duplicate them so that the charge is evenly distributed among the two copies. As a result, the graph can adapt to specifically match a desired neighbor count for each vertex, resulting in a much better performance in storing nonuniformly distributed data.

In [112, 37] we introduced the idea of adding uniformly distributed random one-hot-bit encoded vectors to stored vectors. When performing an approximate search, the extra dimensions created at training are simply disregarded at input and output, but are used for all intermediate retrieving step. Because they are uniformly distributed, they have the effect of compensating for the noneven portion of the network.

II.3.3.2 Sets and Nearest Neighbor Search

As stated in the introduction, associative memories are devices particularly adapted to implementing set datastructures and to solving approximate nearest neighbor search. In the context of block constrained sparse associative memories, in [31] we theoretically derived the error probabilities when solving the exact match problem with uniformly distributed stored vectors.

In [113, 114, 45], we proposed to use associative memories to accelerate approximate nearest neighbor search in the context of dealing with real data. The main problem is that real data rarely takes the form of the concatenation of one-hot-bit vectors.

To create a correspondence between real data and vectors to be processed through block constrained sparse associative memories, we proposed to rely on the use of Product Quantization [115]. Product Quantization consists in splitting raw vectors into disjoint subvectors, each quantized independently. Often authors use random sampling of cluster centroids or variations of the k-means algorithm. Then, quantization of the raw vector is obtained by assembling concatenated versions of each sub-vector.

Interestingly, this quantization procedure has the effect of creating block constrained quantization, where a unique quantized representation is chosen for each block (i.e. for each sub-vector). Therefore these quantized vectors are perfectly suited for block constrained associative memories.

In Figure II.15, we compare various methods for the approximate nearest neighbor search of vectors of the GIST1M dataset, composed of 1 million Gist feature vectors with dimension 960 each. We draw curves using a standard Random Sampling (RS) clustering, the proposed method using associative memories on top of product quantization, and a hybrid method combining both approaches. The parameter r refers to the number of clusters and the parameter k to the number of associative memories used.

In [22] we proposed another technique where vectors to be searched are averaged in groups. This can be seen as an order 1 associative memory, where instead of looking at the Gram matrix of the dataset (order 2 statistics) we look at the average (order 1 statistics). Again, we demonstrated an interesting ability of the proposed method to reduce the computational cost compared to an exhausive search or to existing alternative methods, for real data in vision.

II.3.3.3 Other Applications

We used associative memories to solve a variety of real world problems.

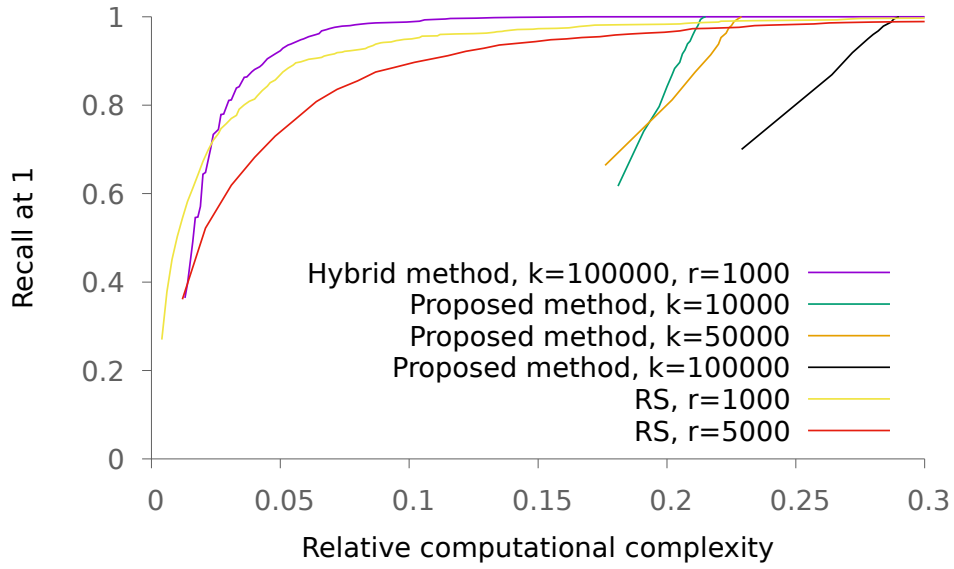


Figure II.15: Approximate match accuracy (or "recall at 1") on the GIST1M dataset as a function of the relative complexity of various methods with regards to an exhaustive nearest neighbor search, for various values of k or r.

In [116] we introduced associative memories as classifiers to be used on top of well chosen feature vectors. Their ability to solve approximate match make them interesting candidates to compare with simple nearest neighbor classifiers.

In [42, 36] we used associative memories to design and implement low-power content addressable memories. Content addressable memories are often used to store bindings between an input domain and an output domain, and can benefit from the approximate match capabilities of associative memories. In the same vein, in [39, 41] we designed and implemented a context-driven search engine using memristors.

In [111] we used associative memories to retrieve pre-computed ideal regimes of power management in complex circuits. The idea here was to exploit the ability of associative memories to retrieve the preregistered ideal power voltage for a situation similar to that encountered.

In [16] we extended associative memories to list all items corresponding to an input query, instead of selecting the closest match. In [117] we extended to all operations of relational algebra.

In [38], we used associative memories to provide efficient implementations of oriented edge detection for low-level image processing.

II.3.3.4 A Novel Look at the Brain

There is no doubt that the brain is one of the most fascinating device that is being heavily studied today. There are multiple ways to look at it. In biology, it is mostly the chemical properties of brain cells that is under the light of research. In neuropsychology, the functional abilities of brain regions. In neuroanatomy, the complex network of white matter.

In the field of artificial neural networks, it is mostly the computational abilities of the brain that have been the source of inspiration for many works. Within this paradigm, the brain is thought of as a device implementing a complex mathematical function, and artificial neural networks are a simple way to model this behavior.

In the field of computer science, there are two main ways to look at the semantic of a program. The first one is to rely on denotational semantics, in which a program is thought of as a mathematical function that associates an input with an output. The second one is to rely on operational semantics, in which a program is a series of transforms that act upon the memory state of the machine it is run upon.

Interestingly, almost all the neural network literature focus on this first aspect. As such, understanding how these algorithms process, store and modify pieces of information is a challenge, since these aspects are not explicitly mentioned in their defini-

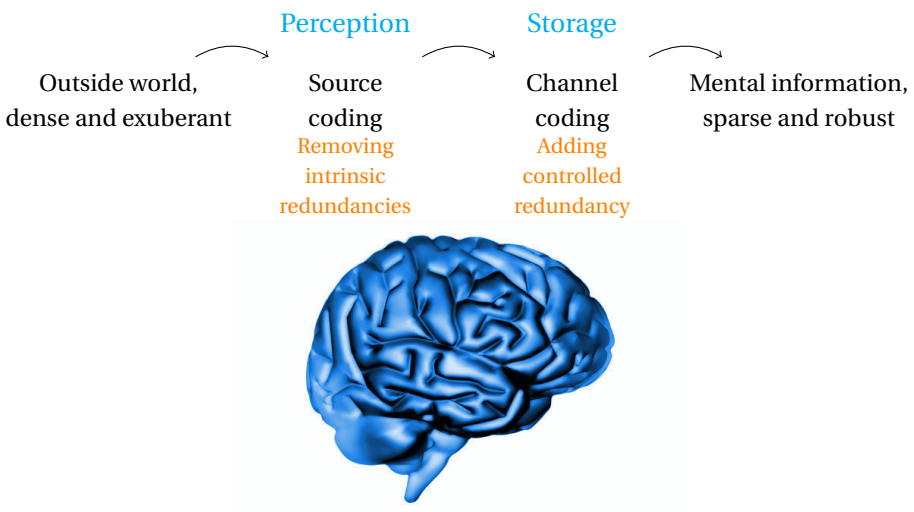


Figure II.16: The brain as an information processing device.

tions.

When we started working on the subject of neural networks in 2008, we wanted to bring a new paradigm to the table, namely looking at neural networks as algorithms that transform information, instead of as functions. Indeed, in the field of information theory, a common way to describe a model processing pieces of information is to use the Shannon-Weaver representation. Using the proof of the Channel Coding Theorem [118], it is possible to show that an optimal way to transmit information consists in concatenating two steps. The first step is to compress the input data, so that any natural redundancy is removed. The second step is to protect the obtained representation using exploitable redundancy, so that errors happening during transmission or storage can be corrected.

Looking at the brain using this representation led us to an original reading of its functioning, that would be split into two parts. In the first part, the brain acquires information from the outside world, and finds compressed inner representations of it. This first step, which corresponds to the perception, is where deep neural networks excel. The second step consists of finding reliable ways to store obtained representations, so that they are not lost through the aging of cells and connections, that irremediably affect the support of mental information. This novel vision of the functioning of the brain is depicted in Figure II.16.

Perception is definitely the ability of the brain that has been the most inspiring to generations of computer scientists. For example, in previous works [92, 93], we introduced models inspired from the retina and low levels of the visual cortex to process

visual information. But surprisingly, the admirable ability of the brain to reliably store pieces of information at the scale of a lifetime has been mostly disregarded in the scientific literature. And yet, the brain is a biological machine that faces lots of defects: about one neuron dies every second in the neocortex, synapses fail to release neurotransmitters more than half of the time, it is estimated that about 80% of the metabolic energy consumed by the brain is due to spontaneous firing of spikes [119]. In front of these attacks, the robustness of long-term memory can only be explained by the use of redundant coding of information, what error correcting coders have been studying for decades in the context of telecommunications.

To be fair, it is worth mentioning that some works have been conducted to stress the abilities of neural networks to maintain a good accuracy when subject to implementation defects, such as the reduction of the number of bits used to represent each value (e.g. [120]). Also, dropout [121], that consists in randomly erasing the activation value of neurons during the forward pass, is a commonly used technique to prevent overfitting.

In [1], we introduced the problem to the information theory community. We also introduced mathematical models of noise in the brain, and analyzed the robustness of proposed associative memories in [11, 13, 12, 14].

A popularization version of these works was published in [122].

II.3.4 Summary of Contributions of the Section

The main contributions in this Section are:

1. Fundamental questions:
   (a) Looking at the brain as an information processing system [1, 11, 13, 12, 14],
   (b) Proposing new models of associative memories with increased storage capabilities [104, 105, 33, 28, 107],
   (c) Extending associative memories to related problems [24, 108, 109, 110, 16, 117, 38, 111, 116],
   (d) Providing a theoretical comparison of associative memories [46].
2. Applications:
   (a) Proposing implementations of associative memories [44, 43, 40, 9, 25],
   (b) Dealing with nonuniform distributions [111, 112, 37],
   (c) Solving set-membership and approximate nearest neighbor search [31, 113, 114, 45, 115, 22],

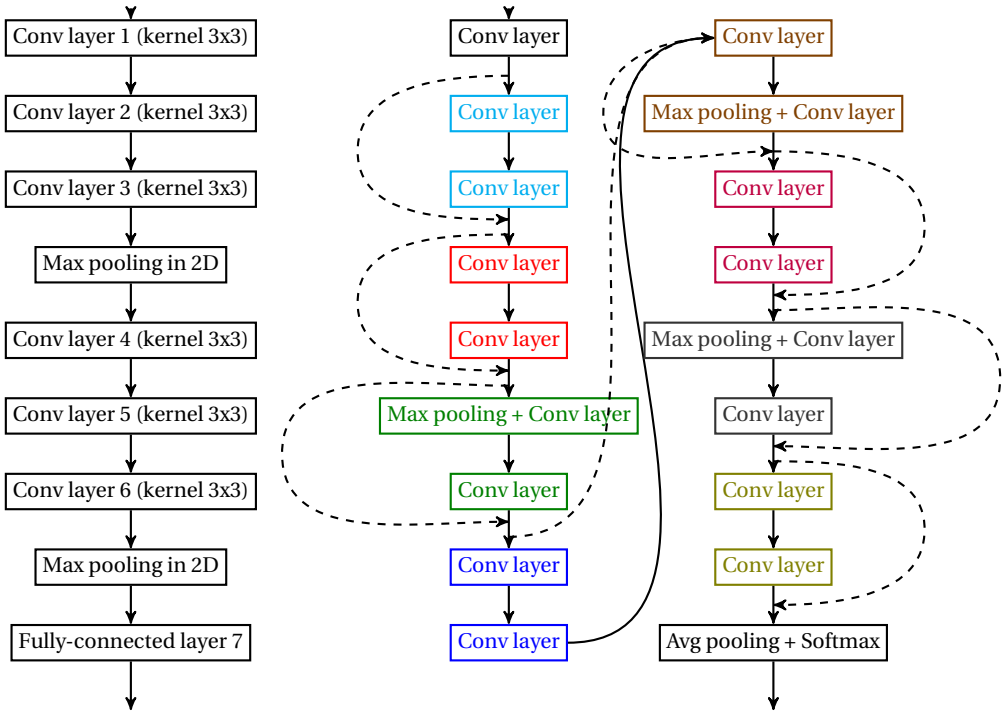


Figure II.17: Example of two deep learning architectures for processing images. The left example is a toy architecture with a cascade of layers, generating increasingly deep representations. The right example is the so-called ResNet18 architecture [126], in which shortcuts (represented through dashed lines) allow to directly transmit the information to deeper layers. As such, convolutions compute small differences to be applied to the directly transmitted input and act as residual filters.

(d) Implementing content addressable memories and light search engines [42, 36, 39, 41].

II.4 Neural Networks for Learning

II.4.1 Compression of Deep Learning Architectures

If neural networks have been successfully used to store and retrieve vectors, they are much more celebrated for their use in learning systems. As a matter of fact, during the past few years neural networks have become the state-of-the-art solution for many problems of machine learning, ranging from classification in vision [123], to sound recognition [124], speech processing [57] or even playing games [125]. In many of these cases, and specifically in vision, the success of these methods heavily depends on the use of appropriate filters in the considered architectures, such as convolutions.

Indeed, deep learning architectures are obtained by assembling layers, as depicted in Figure II.17. But of primary importance are the assembled layers.

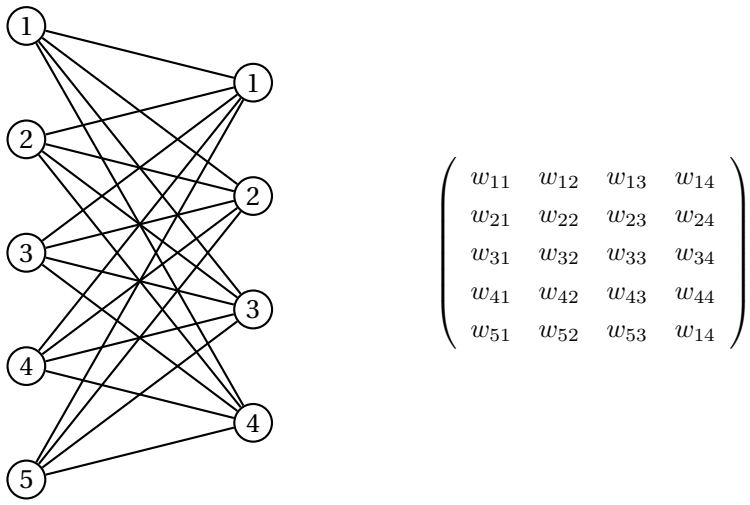


Figure II.18: Example of a fully connected layer, connecting the input space with dimension 5 (left) with the output space with dimension 4 (right). In this example, the linear part of the layer consists in a matrix containing 20 free parameters.

When the input of a layer has no known explicit structure, we often use "fully connected layers". In such a case the input is typically represented as a vector and the linear part simply consists in a simple matrix multiplication. An example of a fully connected layer is depicted in Figure II.18.

On the contrary, if the input domain has strong regularity, such as an existing underlying 2D discrete euclidean domain, one can use "convolutional layers" instead. In that case, the input is typically a tensor of dimension 3, where two dimensions correspond to spatial axis, and the last one to the number of feature maps of the input. Feature maps are parallel representations: for example in the input domain and when considering images, a feature map is considered for each primary color (red, green and blue). The linear part of the layer consists in a collection of convolution matrices, that operate on all feature maps of the input at the same time. These convolution use most of the time very localized kernels, so that the number of free parameters is independent on both the input and output space spatial dimensions. An example is depicted in Figure II.19.

II.4.1.1 Pruning

What is often hidden behind the outstanding achievements of deep learning systems is the fact they typically require lots of memory, computations and power. For example we depicted in Figure II.20 a representation of the error rate of several popular architectures on the ImageNet 2012 challenge, as a function of the memory size. We can

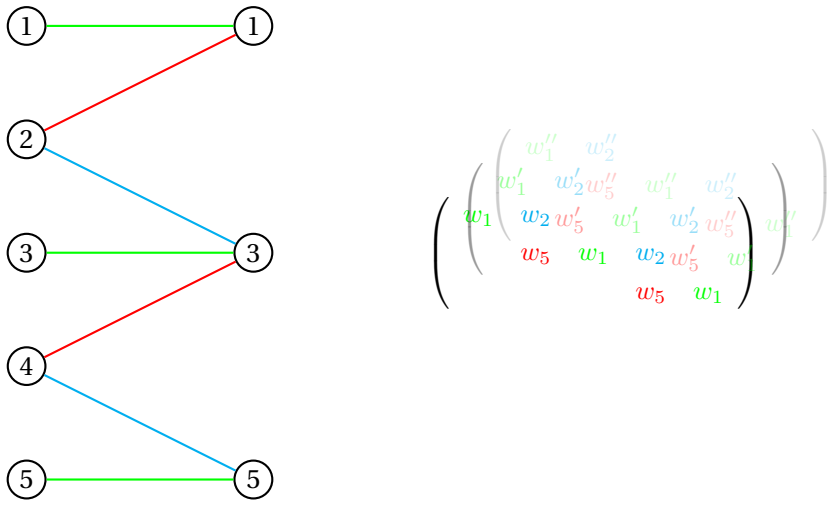


Figure II.19: Example of a 1D convolutional filter. Contrary to a fully connected layer, parameters are shared among lines of the matrix and a lot of 0s (not represented here) correspond to the fact kernels are localized. Since the input is composed of multiple feature maps, matrices are replaced with tensors. Typically convolutions are used jointly with downsampling techniques, such as strides or pooling.

clearly see that there exists some Pareto-type curve between accuracy and memory.

One of the first introduced technique to reduce the size of deep learning architectures was pruning [127]. Pruning usually consists in removing neurons in hidden layers of the architecture. In that case, we talk of "nonstructured pruning". The problem of nonstructured pruning is that it might be hard in practice to leverage the sparsity of matrices. Consider a trivial example where a matrix contains weights encoded using 8 bits each. And suppose that there are 1'000'000 parameters in the considered matrix. Then a sparse implementation with k remaining parameters would cost $(8 + \log_2(k))k$ bits to encode using a list of pairs $(address, value)$. Such a process becomes beneficial compared to a dense matrix when $k \approx 300'000$, so about 70% of the values are pruned. To achieve a reduction of an order of magnitude, this would require to reach more than 96% of sparsity. Instead, other popular methods [128, 129] considered pruning entire feature maps. The interest of pruning feature maps is that the resulting architecture is effectively reduced, with no need to store any indexing.

In [17] we introduced a pruning method that removes all weights in convolutional kernels but one. As a consequence, the effect of the kernel on the corresponding input feature map is a simple multiplication, with a possible shift in the spatial domain. An illustration of the method is in Figure II.21. As a result, considerable gains in both memory size and number of operations required to perform a feedforward pass have been obtained, as reported in Table II.1, Table II.2 and Table II.3.

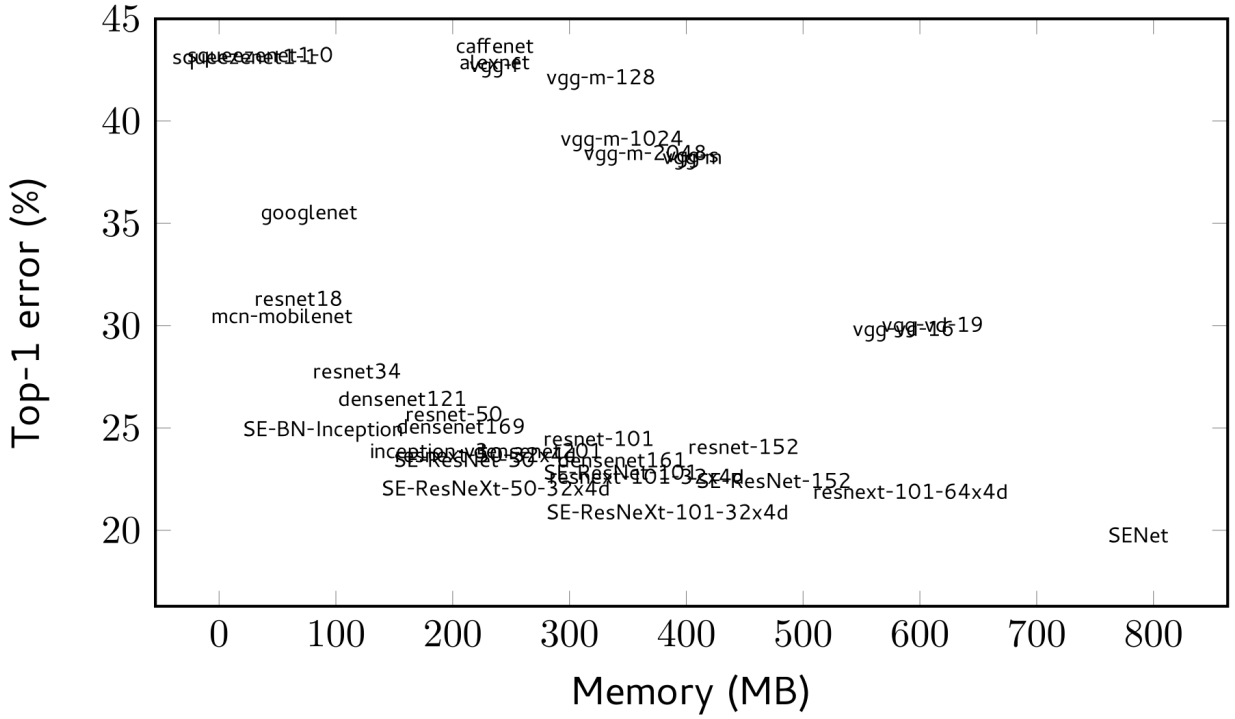


Figure II.20: Comparison of the error rate of several popular architectures on the ImageNet 2012 challenge, as a function of the memory size of the corresponding model.

Table II.1: Comparison of accuracy and number of parameters between the baseline architecture (ResNet20), ShiftNet, ASNet, and SANet (the proposed method) on CIFAR10 and CIFAR100.

		CIFAR10		CIFAR100	
		Accuracy	Params (M)	Accuracy	Params (M)
CLs	Baseline	94.66%	1.22	73.7%	1.24
SLs	ShiftNet [130]	93.17%	1.2	72.56%	1.23
	SANet (ours)	**95.52%**	**0.98**	**77.39%**	**1.01**
Interpolate	ASNet [131]	94.53%	0.99	76.73%	1.02

Table II.2: Comparison of accuracy, number of parameters and number of floating point operations (FLOPs) between baseline architecture (Resnet-56), SANet (the proposed method) , and some other pruning methods on CIFAR10. Note that the number between () refers to the result obtained by the baseline used for each method.

		CIFAR10		
		Accuracy	Params (M)	FLOPs (M)
Pruning	Pruned-B [129]	93.06%(93.04)	0.73(0.85)	91(126)
	NISP [132]	93.01%(93.04)	0.49(0.85)	71(126)
	PCAS [133]	93.58%(93.04)	0.39(0.85)	56(126)
	SANet (ours)	**94**%(93.04)	**0.36**(0.85)	**42**(126)

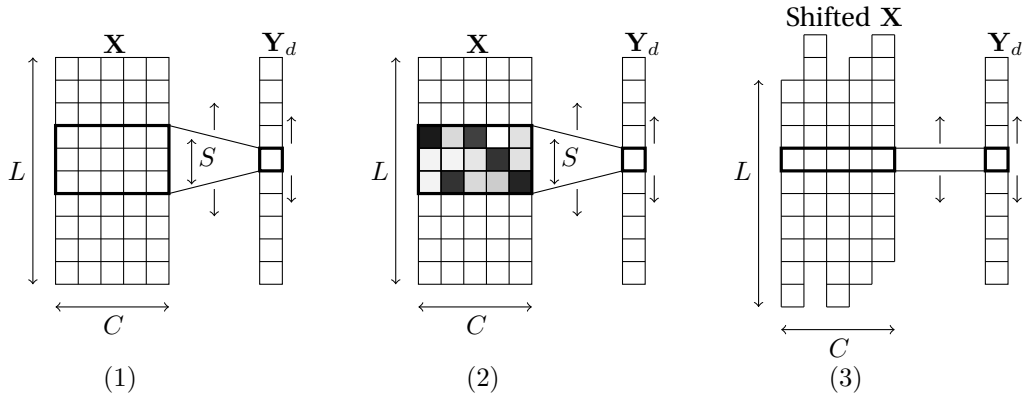


Figure II.21: Overview of the pruning method proposed in [17]: we depict here the computation for a single output feature map d, considering a 1d convolution and its associated shift version. Panel (1) represents a standard convolutional operation: the weight filter $\mathbf{W}_{d,\cdot,\cdot}$ containing SC weights is moved along the spatial dimension (L) of the input to produce each output in $\mathbf{Y}_d$. In panel (2), we depict the attention tensor $\mathbf{A}$ on top of the weight filter: the darker the cell, the most important the corresponding weight has been identified to be. At the end of the training process, $\mathbf{A}$ should contain only binary values with a single 1 per slice $\mathbf{A}_{d,c,\cdot}$. In panel (3), we depict the corresponding obtained shift layer: for each slice along the input feature maps (C), the cell with the highest attention is kept and the others are disregarded. As a consequence, the initial convolution with a kernel size S has been replaced by a convolution with a kernel size 1 on a shifted version of the input $\mathbf{X}$. As such, the resulting operation in panel (3) is exactly the same as the shift layer introduced in [130], but here the shifts have been trained instead of being arbitrarily predetermined.

Table II.3: Comparison of accuracy, number of parameters and number of floating point operations (FLOPs) between baseline architecture (Resnet-50), SANet (the proposed method) , and some other pruning methods on CIFAR100. Note that the number between () refers to the result obtained by the baseline used for each method.

		CIFAR100		
		Accuracy	Params (M)	FLOPs (M)
Pruning	Pruned-B [129]	73.6%(74.46)	7.83(17.1)	616(1409)
	PCAS [133]	73.84%(74.46)	4.02(17.1)	475(1409)
	SANet (ours)	**77.6**%(78)	**3.9** (16.9)	**251**(1308)

Table II.4: Comparison of accuracy and memory usage when training CIFAR10 on ResNet20, for the baseline architecture, the pruned one and the combination of pruning and quantization.

	Accuracy (%)	Memory usage (Mb)
Baseline	94.66	39.04
[17]	95.52	31.36
[17] + [134]	94.00	6.87

II.4.1.2 Quantization and Pruning

Given an architecture, another simple way to reduce its memory size is to quantize its parameters, for example using binary weights instead of float-precision ones. For example in [134] the authors propose to binarize parameters using a straight-through method: they perform the forward pass to compute the outputs using the binarized version of the weights, but then update the float-precision version of the weights during the backward update.

In [48], we proposed to combine this quantization scheme with the pruning method introduced in [17]. The interest is double: by using the pruning method in [17], it is possible to replace the convolution with a simple multiplication (and a shift of the input). By using the quantization method in [134], the multiplication is then replaced by a low-cost multiplexer. As a result, the memory usage and energy consumption of the architecture are greatly reduced. For example, in Table II.4 we observe the effect of combining the methods on both the accuracy of the system and the memory usage.

II.4.1.3 Varying-bit precision

In [50], we introduced a regularizer that aims at reducing the bit precision of weight values during the learning phase. The idea is to push the network into finding a good compromise between bit precision and accuracy.

An example result is depicted in Figure II.22, where we show the result of the proposed method when training a ResNet18 architecture on the ImageNet 2012 challenge. On the left we see the number of bits that has been found for each layer, and on the right we see how the number of bits evolved with the number of epochs of training.

II.4.1.4 Quantization and Factorization

Quantization can also be performed at the scale of small sets of parameters (subvectors) [135]. In [136] we proposed to combine both quantization at the scale of individual weights and groups of weights for increased reduction in memory size. We

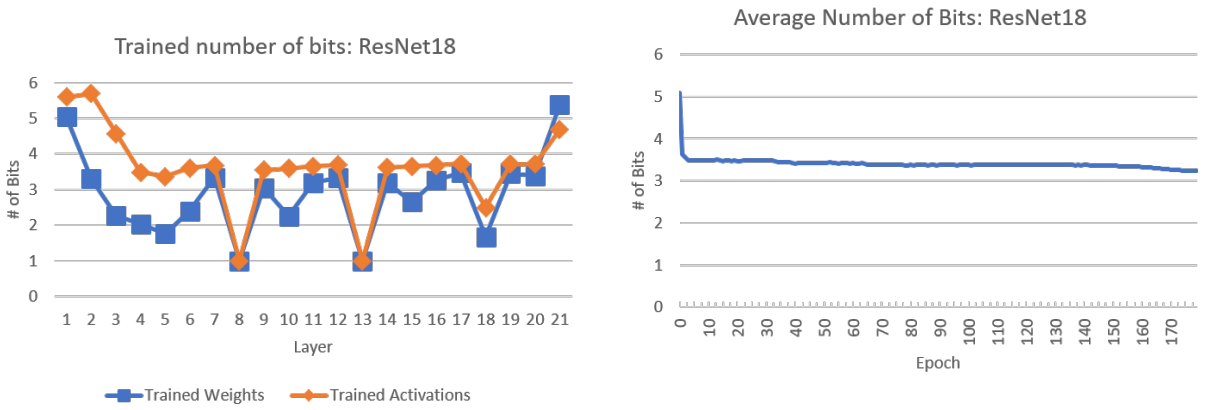


Figure II.22: Depiction of the number of bits per layer found using the method in [50] (left). Depiction of the evolution of the average number of bits per parameter in the architecture, as a function of the number of epochs of training (right). In both cases the architecture was trained on the ImageNet 2012 challenge.

performed experiments on toy datasets (CIFAR10 and MNIST) and demonstrated an ability to reduce by 40 to 50 times the memory consumption compared to the naive full-precision baseline, without any significant drop in the corresponding accuracy.

To this end, we introduced a soft binarization term during the learning phase, that pushed weights to become binary. We made use of a smoothing method that consists in increasing the importance of this regularization throughout the learning phase. Indeed, starting with a strong constraint of being binary would have the effect of creating inescapable local minima in the loss function. We then combined this method with product quantization to obtain the best reductions in size. If Figure II.23 we depict the evolution of the classification error on MNIST (left) or the test accuracy on CIFAR10 (right) as a function of the compression rate (expressed here as a multiplicative factor, so 20 means the size of the architecture has been reduced 20 times).

II.4.2 Applications and Methods

Deep Learning methods can be used for a variety of tasks in the context of machine learning. In the coming subsections we present various use cases we considered in the past.

II.4.2.1 Neural Networks as Accelerators

In [92] we proposed to use neural networks to solve the feature correspondence problem in image matching. The problem can be expressed as follows: given two images A and B, we first extract a set of local feature vectors. Then, the question is to find the

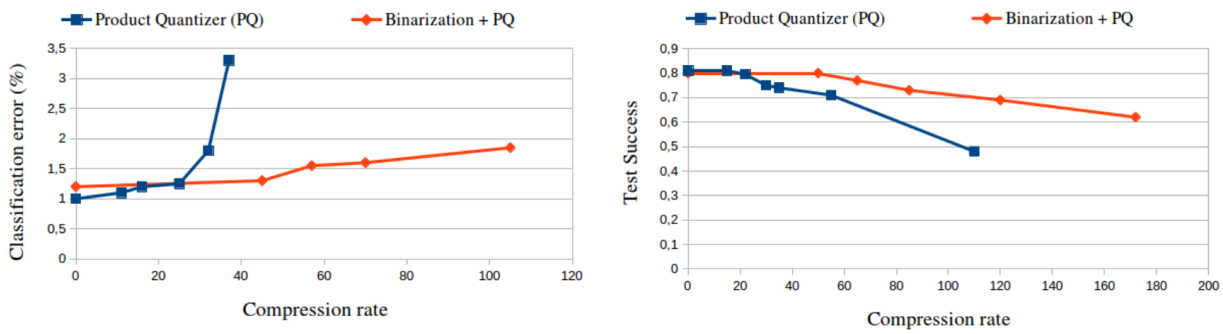


Figure II.23: Evolution of the classification error on MNIST (left) and the test accuracy on CIFAR10 (right) depending on the compression rate.

best one-to-one mapping between feature vectors of both images.

This problem is celebrated in computer science under the name of the assignment problem. It is usually solved using the Hungarian method, which complexity can be made $\mathcal{O}(n^3)$, where n is the number of feature vectors in each image (or the maximum of the two, if they are different). The problem is that this complexity can be too large for some applications, where the time required to find the best matching is constrained.

This is why we proposed a specific architecture of neural networks in [92] to approximately solve the feature correspondence problem with a low complexity. We applied our proposed architecture to finding matches in synthetic datasets, and observed high precision with greatly reduced computational costs.

II.4.2.2 Transfer

Training Deep Learning architectures can prove to be very costly in practice. For many reasons, this procedure is poorly fitted to be implemented on resource-limited hardware devices, for example in the context of embedded systems.

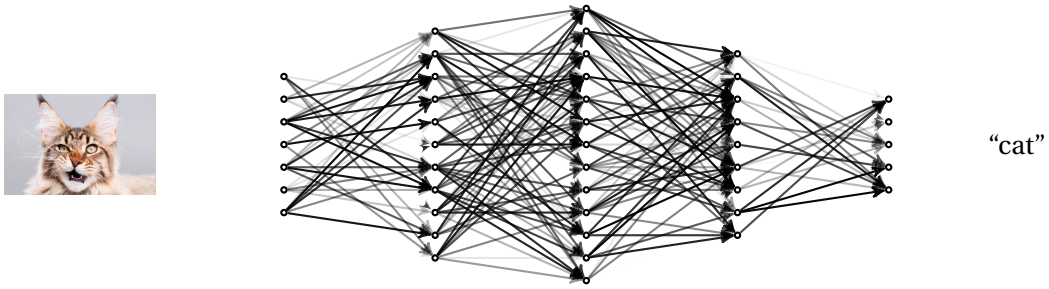
As a matter of fact, there are multiple reasons why deep learning training is demanding:

1. The training dataset has to be processed several times, and as such must be stored in memory,

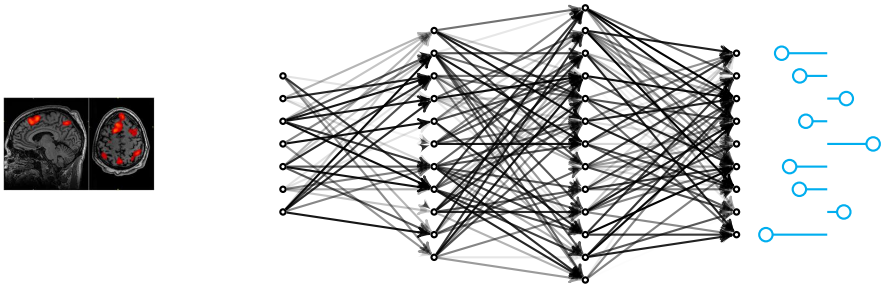
2. During the feedforward pass, all intermediate activations must be stored, as they are used during the backward pass,

3. Some operations are costly to implement, such as Softmax or Cross-entropy,

1. Train a CNN using massive *generic* datasets:



2. Transform your signals into feature vectors using an intermediate representation in the CNN:



3. Use your favorite method on obtained feature vectors:

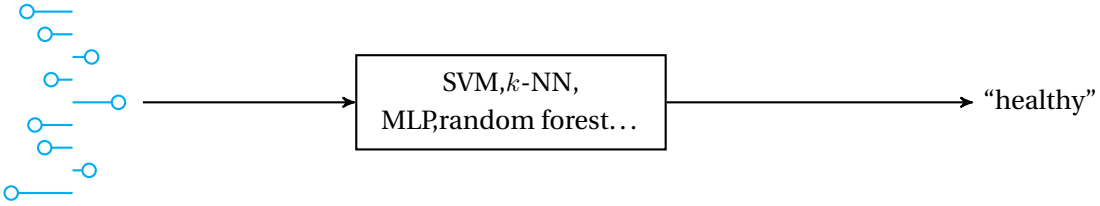


Figure II.24: Illustration of the principle of transfer learning.

4. The use of batch-norm layers, very useful to improve accuracy and training speed of deep architectures, requires heavy computations.

Instead of trying to learn the deep learning architecture on an embedded device, a possible workaround is to rely on transfer learning instead. In Figure II.24 we summarize the principle of transfer learning: first a deep learning architecture is trained on a big dataset, then this deep learning architecture is truncated to act as a feature extractor for another dataset.

In [15] we were interested in showing that transfer learning features have specific distributions that can be leveraged to improve the accuracy of downstream classifiers. In particular it is expected that many of the dimensions of transfer learning feature vectors are meaningless for the task and as such only convey noise. A simple strategy

Inception V3, 1-NN

p	1	4	16	64	256
CIFAR10	0.8519	0.8652	**0.8781**	0.8651	0.8347
ImageNet1	0.9328	0.9354	0.9424	**0.9439**	0.9081
ImageNet2	0.9438	0.9451	**0.9524**	0.9464	0.9171

SqueezeNet, 1-NN

p	1	5	20	100	200
CIFAR10	0.6839	0.7069	**0.7472**	0.6890	0.6225
ImageNet1	0.8854	0.8900	**0.9001**	0.8784	0.8466
ImageNet2	0.8737	0.8802	**0.8926**	0.8669	0.8267

AudioSet

p	1	10	20	40	160
1-NN	0.605	0.704	0.698	**0.724**	0.660
5-NN	0.564	0.704	0.718	**0.727**	0.668

Table II.5: Accuracy of classification, depending on the feature extractor used, the dataset and the number of segments p.

to mitigate the effect of this noise consists in segmenting the transfer learning feature vectors into subparts that can then be processed independently. In Table II.5 we show the effect of segmenting feature vectors on various architectures and datasets, as a function of the number of segments p. The classifier used is Nearest Neighbor (NN) with either 1 considered neighbor (1-NN) or 5 considered neighbors (5-NN). ImageNet1 and ImageNet2 refer to datasets created from the ImageNet dataset that are disjoint from the 2012 challenge used to train the deep learning architectures.

II.4.2.3 Incremental Learning

One of the main interest of transfer learning is that it usually provides features that are so good that even very simple downstream classifiers can achieve outstanding performance. Among others, incremental learning classifiers can be of interest in applications where data is streamed over time. Incremental refer to the ability of a classifier to deal with new classes or new examples over time, without the need to retrain the whole system.

For example in [137] we introduced an incremental classifier using sparse associative memories, to be used on top of pretrained deep learning architectures. In [6, 16, 138], we proposed algorithms and implementations of transfer incremental learning methods, able to compete with state-of-the-art nonincremental methods on off-the-shelf datasets. In Figure II.25, we illustrate the principle of the method introduced in [6]. In Figure II.26, we depict the performance of the method in a class-incremental scenario (top) and in an example-incremental scenario (bottom). In both cases we

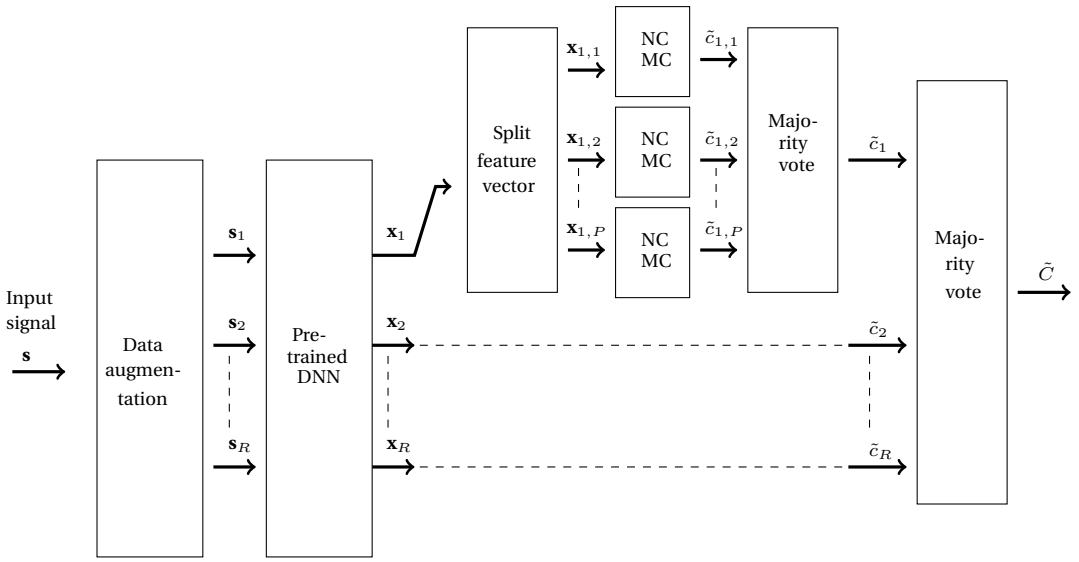


Figure II.25: Illustration of the principle of the method proposed in [6]. First input signals are augmented using standard data-augmentation techniques. Then, these augmented signals are transformed into feature vectors thanks to the use of a pretrained deep learning architecture. The corresponding feature vectors are segmented, and classified independently using Nearest-Class Mean Classifiers (NCMC). A majority vote is first performed on various segmented parts, then on all augmented input signals.

compare the proposed method in [6], denoted TILDA-DA, with popular alternatives. Note that we removed the data-augmentation when performing comparisons to be fair with other methods.

II.4.2.4 Robustness

In [139], authors have shown that deep learning architectures can easily be fooled when facing unconventional perturbations. More recently in [140], a standardized benchmark has been proposed to stress the abilities of deep learning architectures to resist common perturbations in the context of computer vision: blur, change of constrast, environmental conditions...

To improve the robustness of trained deep learning architectures to perturbations, numerous contributions have been proposed. For example in [141], the authors propose to increase robustness of the decision process by using an ensemble composed of k-nearest neighbor classifiers at each layer of the architecture. In [142, 143], the idea is to constrain the network function to be Lipschitz, so that small perturbations of the input cause small perturbations of the output. In [144, 145, 146, 147], the authors suggest to augment the training dataset with perturbated versions.

In [20], we proposed a new formal definition of robustness for deep learning ar-

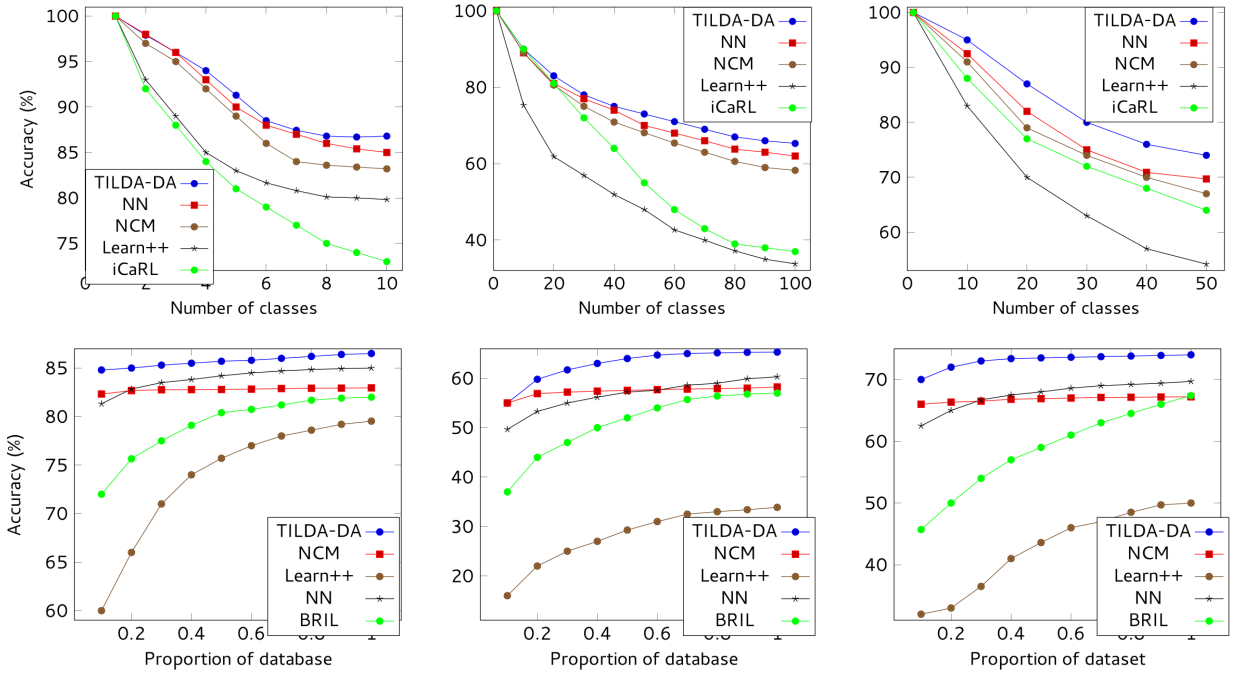


Figure II.26: Evolution of the accuracy of incremental learning methods on various datasets (left:CIFAR10, middle:CIFAR100, right:ImageNet1).

chitectures. Contrary to the usual Lipschitz constraint, our proposed definition takes into account locality. As a matter of fact, the Lipschitz constraint is applied to all the input domain. But we advocate that network functions should be allowed sharp transitions between class domains, far away from the training examples. By conducting experiments, we show that our proposed robustness criterion can help predict the robustness of trained architectures.

II.4.3 Graph Neural Networks

A very active field of research is that of graph neural networks. In this area of research, authors are interested in various problems, ranging from link prediction for recommender systems, to node embedding, semi-supervised learning or extension of convolutional neural networks to irregular domains.

II.4.3.1 Node Embedding

In [96], we introduced a method to project vertices of a given graph to a grid, so that the topology is essentially preserved. As a consequence, it becomes possible to treat graph signals as signals on a grid and to rely on classical convolutional neural networks afterwards. To obtain such a projection, we use an optimization procedure that

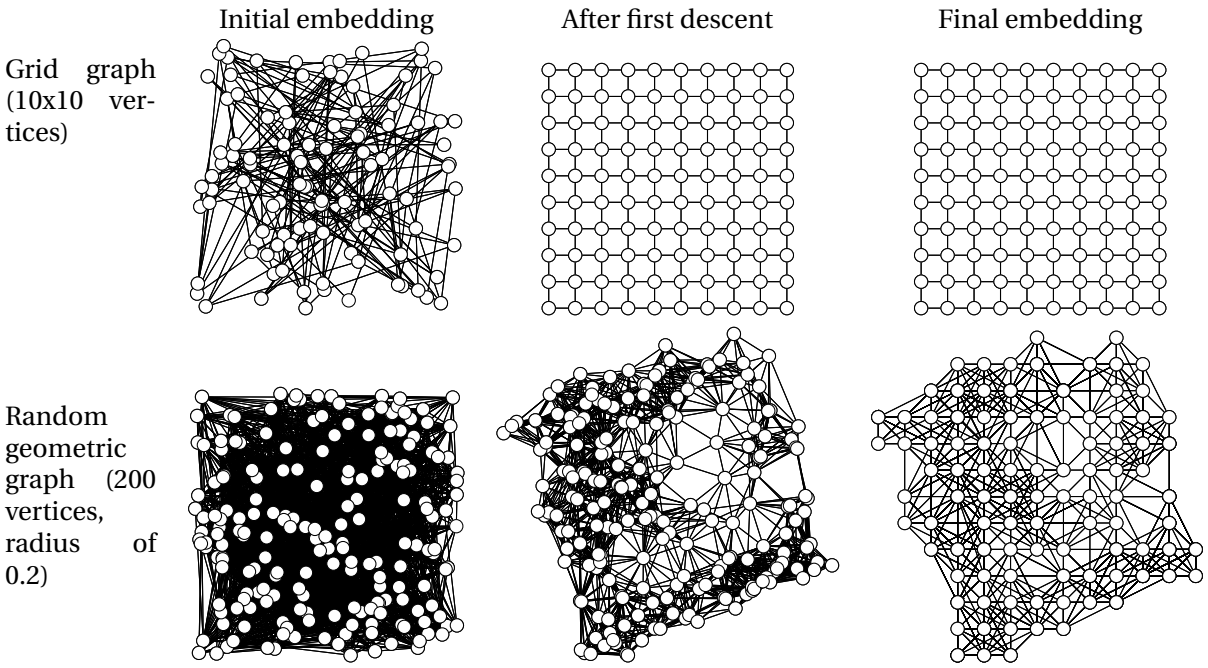


Figure II.27: Illustration of successive steps in the proposed optimisation procedure in [96]. Left column is initial random embedding, middle column is after the first gradient descent, and right column at the end of the process.

first aims at projecting vertices to $\mathbb{R}^2$, and then gradually increases the importance of pushing vertices to integer coordinates. This principle is known as smoothing in the context of optimization. An illustration of the procedure is depicted in Figure II.27.

II.4.3.2 Visualization of Deep Architectures

In [18], we proposed for the first time to use Graph Signal Processing to visualize intermediate representations of deep learning architectures. This visualization tool allows to monitor the training process and even to detect overfitting in some cases. An illustration of the the monitoring of label smoothness at various layers of a PreActResNet18 architecture is depicted in Figure II.28. Note that the sudden changes in smoothness are due to a change in the learning rate.

II.4.3.3 Graph Smoothness Loss

In [19], we proposed to go one step further and to use label smoothness as a way to train deep learning architectures. As a matter of fact, most trained architectures use cross-entropy on top of one-hot-bit encoded outputs. This has important consequences:

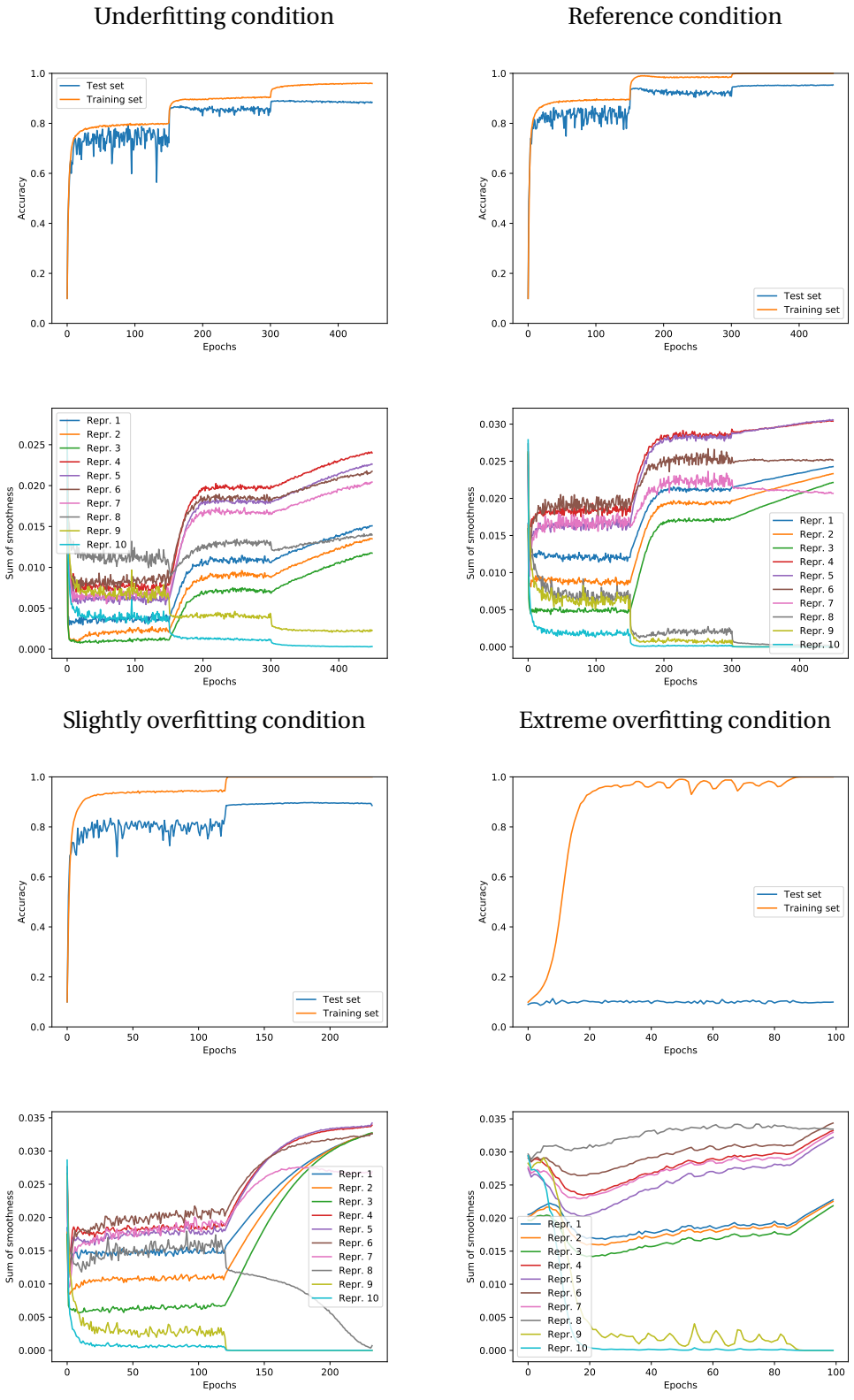


Figure II.28: Comparison of accuracy and label smoothness for PreActResNet18 under different conditions.

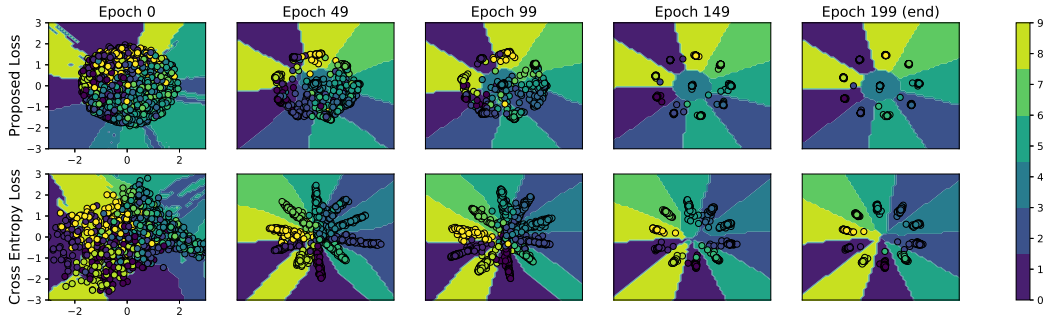


Figure II.29: Illustration of the smoothness loss effect on the output distribution when training ResNet18 on CIFAR10. Comparison with a standard cross-entropy when using a bottleneck layer.

1. Both the initialization of the parameters and the dataset are ignored when choosing which class is associated with which one-hot-bit vector. As a result, the training process may have to strongly deform the topological space during training,
2. The number of dimensions in the output vector is determined by the number of classes, making it harder to adapt in a class-incremental learning scenario,
3. The classifier is not only forced to separate inputs of distinct classes, but also to merge those of a same class, even if classes are actually formed of well-defined disjoint subclasses.

By using label smoothness as a loss, we can avoid these shortcomings. We empirically demonstrated the effectiveness of the proposed method in reaching state-of-the-art accuracy in [19]. An illustration of the process is depicted in Figure II.29, where we specifically chose the output dimension to be 2 for easy visualization.

II.4.3.4 Deep Learning with Irregular Signals

We also made several contributions to the problem of extending convolutions to irregular domains. As a matter of fact, we saw earlier in this section that when dealing with regular input signals, convolutional neural networks can be used. They offer outstanding gains in performance, specifically for vision datasets. For example, with the CIFAR10 dataset, the state-of-the-art performance while disregarding the input structure is 31% error rate [148]. When using state-of-the-art convolutional neural networks, the error rate drops at 2% [149].

Of course there has been numerous works in the literature in this domain. As most reviews focused on an experimental comparison of methods, we proposed in [8] a unified formalism where pros and cons of each methods can be made apparent.

In [150, 151], we introduced methods to extend convolutions to slightly irregular domains, by building on analogies with the regular case. In [152], we proposed a generalization of convolutions where weight-sharing can be learned jointly with the parameters for solving the considered task. In Figure II.30, we depict an illustration of the methodology proposed in [150] to extend convolutions to irregular domains. The methodology builds upon 5 steps: inferring a graph, inferring translations on the graph, designing weight-sharing, designing data-augmentation and finally designing subsampling.

II.4.4 Summary of Contributions of the Section

In the field of neural networks for learning, the main contributions are:

1. Fundamental questions:
   (a) Contributing to the extension of convolutional neural networks to irregular domains [96, 152, 150, 151, 8],
   (b) Analysing transfer learning features for better downstream classification [15],
   (c) Proposing formal definitions of neural network robustness [20],
   (d) Defining new loss using graph representations of the output space [19].
2. Applications:
   (a) Compression of Deep Neural Networks [17, 48, 17, 50, 136],
   (b) Incremental learning on chip [137, 6, 16, 138],
   (c) Solving the feature correspondence problem [92],
   (d) Using graph signal processing for monitoring of deep neural network learning process [18].

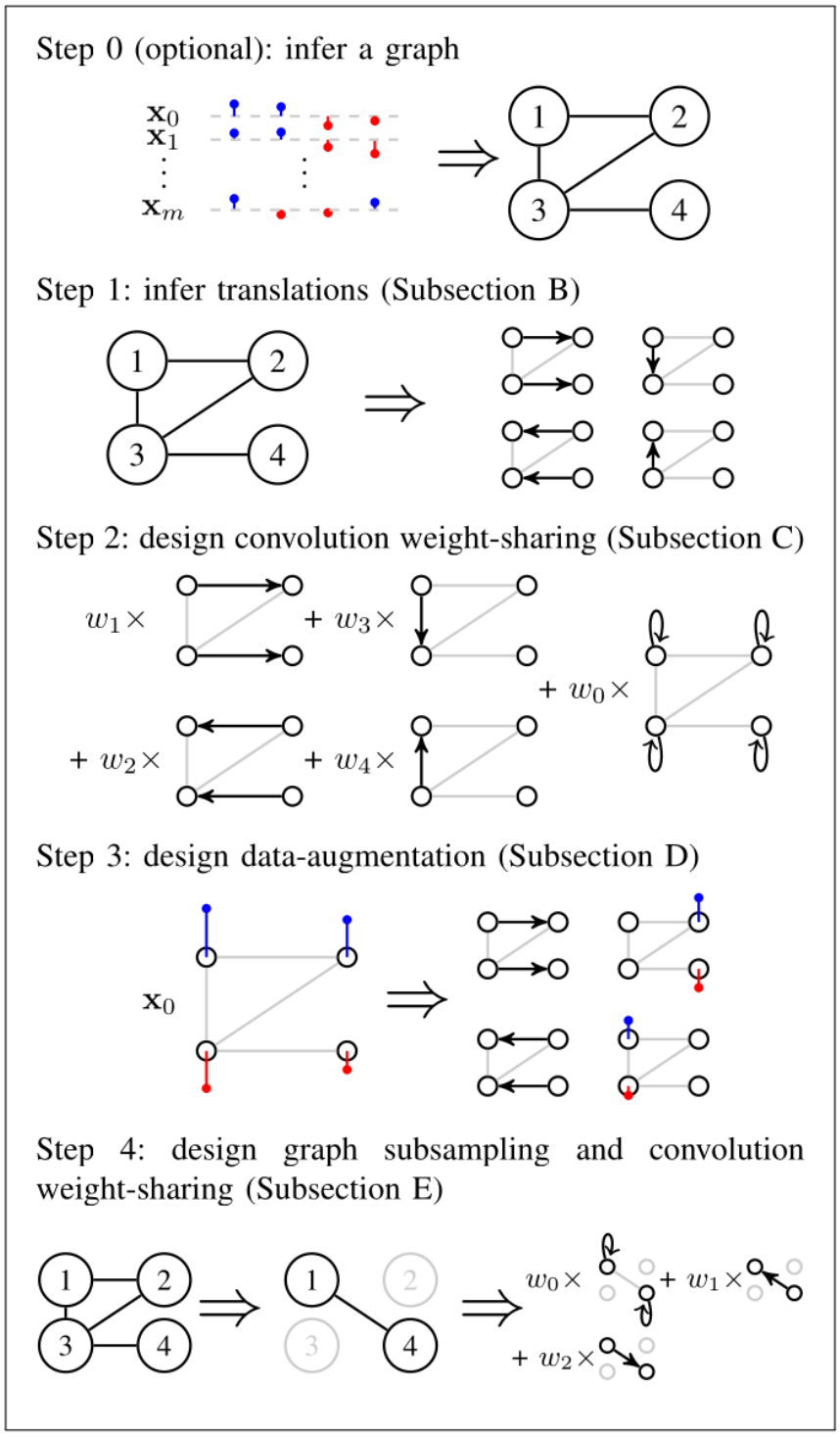


Figure II.30: Illustration of the method introduced in [150] to extend convolutional neural networks to irregular domains.

Chapter III

Future Work and Directions

Throughout the past sections of this document, I hope it appeared clearly how enthusiast I am about interdisciplinarity. The main question that drove my research so far is that of the abilities of machines to learn and gain in autonomy, and I looked at the question using multiple paradigms: information theory, machine learning, signal processing... I have great hopes about how this field of research could be beneficial to most in the coming decades. This is why this section mainly discusses these points.

I have thought of three main directions to take for my future research, in the continuity of what I did in the past few years. The first axis is dedicated to contributing to the rise of reusable artificial intelligence, where methods are thought of outside the bounds of a constrained problem. I have been greatly influenced by Yoshua Bengio for this topic, who promoted curriculum learning during the past few years. The second and third axis are dedicated to making AI systems accessible to most. The second axis focuses on the problem of data-thrifty settings where either few data is available, or few labels can be obtained. The third and last axis focuses on the problem of AI computing at the edge.

III.1 Compositional AI

Most of the literature in machine learning focuses on well separated tasks: naming objects in natural images [153], identifying faces [154], recognizing characters [155], analyzing sentiments in texts (e.g. IMDB), classifying short sounds [156]... and bypass the key problem of extracting common ground knowledge. However, human intelligence is different: each newly processed data has the ability to enhance past, present and future acquired knowledge, building increasingly adequate and adaptable representations of the environment and inputs. Consequently, humans are much more effective at dealing with small datasets through continual [157] and curriculum learn-

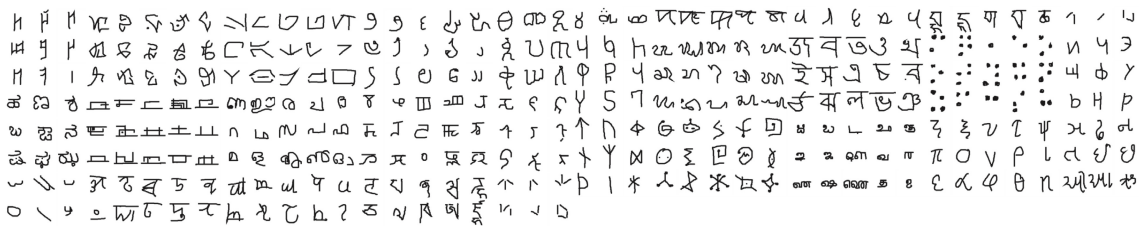


Figure III.1: Example of characters extracted from the Omniglot dataset, illustrating the difficulty of learning a new alphabet with only one example per character.

ing [158], leading to better performance than machines when facing few-shot or zero-shot learning. Understanding how to efficiently transfer intelligence between problems is arguably one of the most important locks to be lifted to access the next level of machine intelligence.

The machine learning literature has been interested in transferring previously acquired knowledge to newly considered tasks. For example transfer learning methods [159, 160, 57, 161] propose to use representations optimized for a first task as features for a second task. In meta learning [162, 163], representations or parameters of systems are optimized on preavailable large datasets, to be fine tuned on smaller ones. But in both cases the idea remains that newer tasks cannot help in finding better representations for older ones or subsequent ones. The scope of the problem is indeed most of the time narrowed to using a large dataset to increase accuracy on a smaller one. These transfer methods consist more of a data augmentation than a real ability to uncover and exploit the common ground. As a consequence, training over sequential tasks and adaptability remain open scientific challenges, despite being applicable to a vast number of practical problems [164].

There is one main reason to explain this shortcoming, and it is strongly related to the supremacy of deep learning models in the literature. Indeed, deep learning models are usually trained through stochastic gradient descent algorithms, requiring to cycle through datasets a large number of times. When processing streaming data, the consequence is catastrophic forgetting [157] (i.e. the fact the procedure specializes on lastly considered data elements [165], at the cost of losing previously acquired knowledge). Some alternative exists, but they usually achieve poorer accuracy. For example, we proposed a technique at the 2020 CVPR Challenge on Incremental Learning and won the first prize for the New Instances and Classes subchallenge `https://sites.google.com/view/clvision2020/challenge/challenge-winners`.

If stochastic gradient descent remains (by far) the most efficient method to train deep architectures, the fact it is deployed without explicit control of what exactly is

being learned at each layer of the architecture makes it impractical to preserve previously acquired knowledge.

Deep learning systems are usually designed as a composition of layers – i.e. very simple mathematical functions –, containing parameters to be trained. But in most cases these layers are initialized randomly (i.e. they are only containers) when facing a new task, or updated without taking into account the impact on other tasks. What if, instead of assembling containers, we would consider assembling contents, in a way that both previous and future tasks can benefit from the current training?

A key question I would like to investigate is that of training machines able to enrich their representations of inputs and contexts with each newly processed data, being able to adapt and transfer knowledge efficiently. The ambition is to open the way to a plethora of new usages of AI in domains where data acquisition is expensive or changing, as well as contributing to the invention of sustainable artificial intelligence. Such a technology would lead to a breakthrough in both fundamental and applied sciences, by democratizing AI to small businesses and small datasets.

To offer this new generation of intelligent machines, several subproblems must be addressed:

Subproblem 1: setting up systems able to semantically define and maintain the purpose of a layer in a deep learning architecture. This step is necessary to ensure that knowledge can be enriched and adapted without the downfalls of catastrophic forgetting. This raises new question in the field of transfer learning, where the transfer is not considered as a one-time forgettable adaptation, but instead as a sustainable binding between multiple problems.

Subproblem 2: proposing learning methods able to uncover common ground between tasks. Indeed, it is by optimizing common ground that multiple tasks can be mutually beneficial. Instead of an implicit reuse of previously acquired intelligence that would risk obfuscating more and more the common ground, the idea here is to explicitly identify and segregate reusable and specific knowledge for each newly considered task. This raises new questions in the field of transfer learning where the ultimate goal would not be the performance on individual tasks, but the ability to factorize most of the common ground knowledge for better adaptability and reusability. We call this ability the disentangling transfer.

Subproblem 3: targeting the disclosure of atomic (fundamental) pieces of knowledge allowing to solve a given problem. Only through the exposition of these elements can a truly compositional AI system emerge, in the sense that it allows to express the solution of any task as a combination of these atoms, along with some domain adaptation. The problem of finding causal variables is central in the field [166, 167], but

remains a very difficult challenge. The idea would be to tackle this question using continual and transfer learning as a mean instead of a constraint: it is because humans are forced to experience lifelong learning that they must acquire this ability to find reusable and adaptable representations of the world. We call this objective the atomization learning.

III.1.1 Congruent Updates

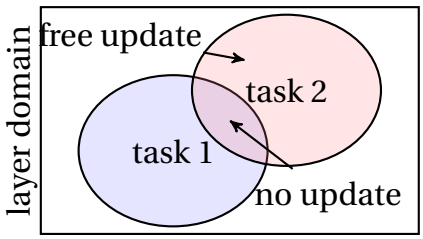
In computer science, a congruent update usually refers to the fact a library has been updated while remaining fully compatible (backcompatible) with previous versions. It can be ensured by looking at the semantics of the library, and checking that previous usages remain unchanged. In deep learning, layers of an architectures are jointly optimized on a given task. As a consequence, there is no controlled or even explicit semantics about them.

In the context of distillation [168], authors have shown that it is possible to train an architecture by mimicking the outputs at each layer, thus showing that the semantics of a layer can be fully captured by the associated pairs of inputs/outputs of each layer on the training set. Despite exploratory, these ideas show it is possible to capture the semantics of a layer. Once the semantics is defined, it is then possible to propose a formal definition of congruent updates in the context of deep learning layers.

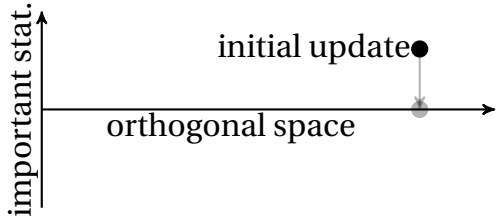
In order to better understand the transforms induced by a given layer in the manifold of the input domain it has been trained on, we think that Graph Signal Processing [169] (GSP) representations is a promising line of research. Indeed, we showed in a previous work that GSP can successfully be used to visualize the effect of individual layers in the dynamics of deep learning architectures [18, 19]. Graphs have the interest of capturing higher order statistics than the mere input-output associations previously mentioned, and as such can precisely model the mathematical transform associated with a layer.

Based on recent works in the field of distillation and graph signal processing, we thus propose to define the semantics of a layer for a given task so that congruent updates (updates maintaining the semantics) can be performed. We split this work in two parts:

1. In a first part we aim at characterizing which portions of space are used on the given task and at the given layer. Thus we can allow updates in all unused directions of space without compromising the performance on previous tasks.



2. In a second part, we aim at identifying which statistical characteristics of the input and output domain (evaluated on training samples) must be preserved to maintain performance on previous tasks. Again, the idea is to allow updates that preserve these important statistical characteristics.



III.1.2 Disentangling Transfer

In most cases in transfer learning or meta learning, the main idea is to exploit the knowledge acquired on previously considered datasets (called support) in order to increase performance on a newly considered dataset (called query). As a consequence of this asymetric paradigm, it is usually the case that the support is much larger than the query, and therefore the query is not considered helping the performance on the support. For example, in [6, 137], we look at strong versions of transfer in continual learning settings.

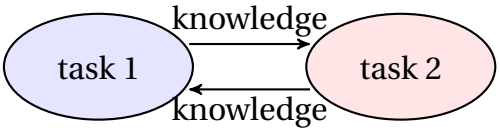
In this work, we consider introducing disentangling transfer. Basically transfer learning hypothesises that there are two types of information that help solve a task. The first type of information is reusable knowledge, that is to say knowledge that is inherently more generic than the task. For example, when learning to classify cats and dogs in images, it is expected that part of the knowledge learns to ignore the background of the image, which can be considered reusable knowledge for most tasks of recognizing objects in natural images. Then, the second type of information is specific knowledge. Continuing the previous example, learning to distinguish between the nose of a cat and the nose of a dog is a very specific feature of the dataset, and there is little chance it can be of use to other tasks.

When performing transfer (or meta-learning) the usual way, it is expected that training on the query set will implicitly update the reusable features and disregard the specific ones. But such a strategy only makes sense for very simple applications of transfer, where the goal is to use previously acquired knowledge to the sole purpose of solving a new one. In the more general setting of transfer across multiple sequentially treated and mutually profitable problems, it is essential to process reusable and specific knowledge as heterogeneous features. Following this idea, in a previous work we already showed that transfer features have specific distribution traits that can be

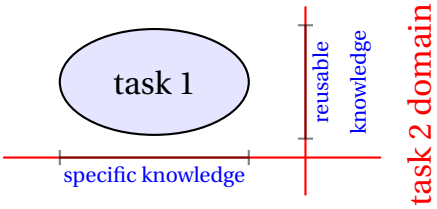
leveraged to enhance transfer performance [15].

Disentangling transfer refers to the idea that it is possible to perform transfer with explicit identification of reusable and specific knowledge. The idea is that congruently fine tuning reusable knowledge while learning specific knowledge from scratch should considerably help in achieving transfer in competitive lifelong learning settings. To this end, we split the work in two parts:

1. First we want to introduce and promote a new look at transfer learning in the community, where the objective is not simply to exploit data available in a large training set to perform better on a new, smaller one, but to establish sustainable bindings between problems, where improvement on a problem is leading to improvement in other problems. We consider introducing meaningful benchmarks and reviews of the existing methods in the fields of transfer learning, meta learning and domain adaptation on these benchmarks, to convince the community of the interest to look at *strong* versions of transfer, that would benefit to all research with small datasets.



2. Then we aim at introducing ways to identify and segregate reusable and specific knowledge, in the context of transfer learning. The idea is to introduce invertible transforms that project reusable and specific knowledge in disjoint dimensions of space (disjoint feature maps in the case of convolutional neural networks). This process can be performed during fine tuning. By using previously introduced benchmarks, we aim at showing the specific interest of looking for meaningful common ground features.



III.1.3 Atomization Learning

Fundamentally, the purpose of machine learning is to uncover causal (explaining) variables from data. Indeed, the rational behind most machine learning literature is that data can be seen as a (stochastic and noisy) process, which associates causal variables to a realization. Understanding how data can help in retrieving the procedure and the causal variables is the grail of the field.

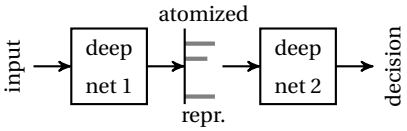
Classification can be seen as a partial answer to the problem, where data is associated with part of the content. But classification often looks at only a portion of the problem, since the content of interest is only part of the causal variables that helped in creating the data. In most deep learning architectures, the procedure (i.e. the mathematical formulation of how inputs are associated with outputs) is fixed, and only latent variables (i.e. parameters) are trained.

When it comes to solving multiple tasks and creating reusable knowledge, the problematic comes from the fact the most efficient architectures, in terms of accuracy on the task, are not necessarily the most efficient in terms of reusability.

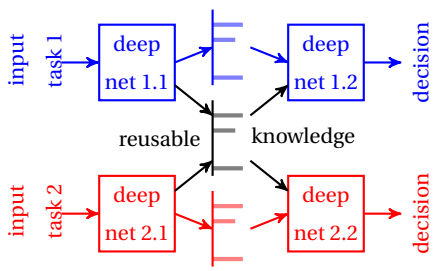
Instead of fixing the architecture and looking for the best parameters, the idea of atomization learning is to reverse the problem: what if we looked for the best architecture to solve the problem, knowing that we want to obtain a latent representation that is aligned with our objectives: independent coordinates, sparse response to environment changes, no loss of information and reusability?

There are thus two sub-challenges to be addressed:

1. The first step is to find atomization methods through constrained latent representations. The idea is to build upon the emerging literature on generative adversarial networks [166] and sparse and variational autoencoders [167, 161] in order to find new ways to perform both classification and atomization at the same time.



2. Then, the idea is to find atomization methods in the specific context of learning multiple sequential tasks. Indeed, while atomization learning can be seen as an additive constraint when learning to solve a single task from an agnostic starting point, it becomes an asset in the context of lifelong learning, where finding an atomic common decomposition of each task with a common base alphabet could dramatically improve the accuracy of systems facing highly challenging problems, such as the ones introduced in the previous subproblem.



III.2 Few-Label and Few-Shot Learning

In many cases, data labelling can be expensive despite its acquisition is cheap (i.e. few-label). In other cases, data acquisition can be complex or expensive (i.e. few-shot). In such cases, deep learning systems are likely to be hard to train, due to the discrepancy between the number of parameters to tune and the available information in the dataset to use. In the past few months, we have worked on solutions to address few-shot learning [170, 171, 172, 173]. We are currently ranked first on `papers-with-code` on the related benchmarks using the miniImageNet dataset[1].

More generally, authors have looked at many options to address these problems:

1. Transfer learning and meta learning have already been presented in the previous section. They both consists in using another dataset to enrich representations for the newly considered one.

2. Generative methods, including GANs [166] and sparse and variational autoencoders [167, 161], can help in artificially enriching the labelled data.

3. Semi-supervised graph methods [174] propose to use two types of information: feature vectors (typically obtained through transfer), with a similarity additional information represented through edges of a graph.

4. Finally, active learning [175] is a field where data labelling is performed interactively with the learning process.

There are many ways graph methods could be used to enhance the quality of few label or few shot learning performance.

III.2.1 Improved Transfer

Most often, transfer refers to the idea of truncating a pretrained neural network right before the ending fully connected layer in order to exploit the corresponding activations, computed on the considered dataset inputs, as feature vectors. This idea is largely based on the fact deep learning methods are often considered as the concatenation of a representation extraction function with a classifier (here: a simple logistic regression).

Instead, we could use graph signal processing, where vertices would correspond to data points in the query set, to detect which layers (or combination of layers) offer the best representation. For example, this could be performed by looking at the

[1] `https://paperswithcode.com/sota/few-shot-image-classification-on-mini-2` and `https://paperswithcode.com/sota/few-shot-image-classification-on-mini-3`

smoothness of the label signal [18, 171]. This could interestingly help in distinguishing between layers of the architecture that are good feature vectors for classification in general and which one are overspecialized to the support set.

In a second time, this would open the way to specific training techniques for the support set, where the training is not agnostic of the task to be processed afterwards.

III.2.2 Controlled Data-Augmentation

Data augmentation is a standard procedure when training deep learning systems. It consists in artificially increasing the size of the training set by considering combinations of transforms that are known to preserve the class of represented objects. For example, in the context of vision, data augmentation techniques usually consists of small shifts, flips, rotations of input images.

In the field of deep learning architecture robustness, data augmentation is used to train architectures to be robust to small deviations of inputs. In this area, transforms can be in any direction of space, but with a limited radius. Again, the rational is that a small deviation of the input is unlikely to change the nature of what is being represented in the image.

One can say that the inherent objective of classification is to find the domain of each class. A simpler, yet challenging problem, consists in finding the class domains in the vicinity of training examples. If we disposed of a system able to do so, this would open the way to a whole new level of data augmentation, where inputs no longer be points of space but large portions of space.

Determining the local structure of the manifold of a class domain is something that could be performed using graph techniques. For example, graphs could be generated as a proxy to represent the structure of a class domain. New points generated through data augmentation could be compared against this graph to be tested for belonging to the corresponding manifold or not. This first step could then be used to feed classification procedures.

III.2.3 Learnable Semi Supervision

In the field of Graph Signal Processing, many techniques have been proposed to infer graphs based on signal observations [83, 84, 80, 23, 176]. But in these examples, graphs are inferred agnostically of the the task being solved. As a consequence, possible disposable information, such as weak supervision, is disregarded at this step of the process.

Instead, we could propose techniques to infer graph structures with weak supervi-

sion. This could take the form of the method described in [177], where supervision is used to determine the low spectrum of graphs, that are then inferred using some prior desired property about them (e.g. sparsity). The obtained graphs could then be used in order to apply graph supported semi-supervised techniques on the considered problems.

As a further step, graphs could be learned jointly with parameters while solving the task, using some regularization to avoid overfitting, in a continuation of what we tried in [152].

III.3 Training Deep Learning on Chip

III.3.1 Context

There are many reasons to think that deep learning on chip is a key challenge for the coming years.

From the technical perspective, deep learning faces challenges where it has to be deployed on embedded resource-constrained devices, sometimes for realtime processing (e.g. assisted surgery), sometimes for memory-limited or energy-limited devices (e.g. smartphones). In both cases, the trend is to perform learning partially on chip as a way to accommodate for the specific environment of the user.

From the scientific perspective, reducing the size of architectures while training could open the way to better visualization and understanding of the core principles that allow these systems to reach state-of-the-art performance. Understanding and interpretation is extremely important in domains where decision must be explainable (e.g. automatic diagnostic). Also, Deep Learning is a field of research that is mostly experimental, and thus such that progress is limited by the ability to train fast. Accelerating training could dramatically improve the speed of research in the domain.

From the societal perspective, it becomes more and more clear that AI is going to be one of the most significant source of energy consumption in the coming years, leading to an environmental impact. There is thus urgency in reducing power consumption of these systems. On a related note, deep learning requires expensive hardware to reach state-of-the-art performance on challenging datasets, thus preventing small businesses from being able to compete with major companies. A possibility to approach this performance with limited hardware could help in reaching a better balance between small and big companies in the field.

The main problem is that deep learning systems trained through gradient descent are very poorly fitted to existing hardware. Most of the literature in the field of com-

pression (c.f. Chapter II) is focused on reducing architectures that were trained offline.

Inventing new techniques to replace gradient descent, adapting compression methods to the learning phase, and improving transfer learning are thus very promising directions of research (already mentioned in the two previous sections). We develop these ideas in the next sections.

III.3.2 Replacing Gradient Descent

Stochastic Gradient Descent is a key feature of deep learning systems. For many reasons it is believed to be the most adapted learning procedure to find adequate minima of the loss function [178, 179]. However, it is also the main reason why training deep learning architecture on chip is a challenging problem. In [180], the authors show that it is possible to express the training objective as the minimum of an energy function, which is a very promising direction of research.

Instead of searching ways to implement the exact gradient descent through backpropagation, it could also be promising to look at approximate methods. For example, the gradient of a function can be approximated by looking at its variation when suggest to a small perturbation. Given a very fast implementation of the forward pass, an alternative to backpropagation would be to estimate the gradient by looking at the effect at the output of small random perturbations. Random descents are another possible alternative, where directions to explore would be randomly chosen, instead of focusing on the gradient.

Another line of research is layer-wise training. Given adapted layer-wise representations of the loss function, an architecture can be trained layer-wise, with potential significant drops in the performance [181]. Considering our recent contributions to the monitoring of the training of deep learning architectures using Graph Signal Processing [18, 19], it would be interesting to see if the old ideas of layer-wise training can benefit from the recent advances in the field. In particular, residual architectures have the interest of directly propagating the input representation to intermediate layers, so that there is no information loss when performing layer-wise training.

Finally, in [182], the authors show that at initialization (after random generation of the parameters of the architecture), there exists subnetworks of the given architecture that can achieve the same performance as the complete initial one. These ideas suggest that initialization of parameters is key to the success of deep learning, and that randomness somehow create good starting points that are obfuscated with a lot of superfluous unimportant weights. Finding ways to identify which subnetworks are promising could significantly reduce the cost of training.

III.3.3 Compressing the Learning Phase

Most of the compression methods introduced so far in the literature, including quantization [134], clustering [183], pruning [127], distillation [168], are only fully effective if they are optimized during training, so that only the result architecture can fully benefit from the reduction.

As a way to benefit from reduction during training, a solution would consist in using progressive compression, where part of the compression would be performed while the training process is still unfinished. This would imply to revisit the proposed methods and to stress whether they can be beneficial when applied at early stages of the training process. For example, pruning could be performed gradually, starting from 0% to the targeted value at the end of the training. Finding the best profile of pruning during training would then be a fair question.

Instead of revisiting existing methods, signal processing tools could be invented to prematurely identify excessive parts of the trained architectures, along the training process. Based on statistics of the gradient or the parameters, these tools could help in improving the quality of compression, both in terms of time and quantity.

Finally, the fact it is possible to considerably reduce the size of trained architecture with no impact on accuracy suggest that gradient descent is not as effective as one could think, particularly when dealing with efficiency of the obtained parameters. Being able to directly efficiently train smaller architectures, without the need to start from a larger one, could be investigated by looking at modifications of the back-propagation algorithm or the random initialization of parameters.

III.3.4 Privacy and Edge Computing

Jointly with the rise in popularity of deep learning methods, concerns grow about how these methods could interfere with privacy. As a matter of fact, since deep learning models are best trained with large datasets, it is often more efficient to gather lots of data in cloud datacenters, where models are optimized.

When privacy is a major concern, training at the edge becomes necessary. It brings multiple challenges: *compression of models*, so that they can run with limited computational power, *lack of data diversity*, since the data used for training is mainly acquired locally, and *lack of supervision*, as in many contexts it is not reasonable to ask for a local user to perform labeling.

These three challenges open exciting questions that are related to the previous two sections: reusable AI and few-shot solutions, with the added difficulty of real-world streaming acquisition of data. Conducting research with the aim of making edge sys-

tems competitive in terms of accuracy with centralized solutions could become a major step towards the rise of trustworthy AI.

III.4 Ethical and Societal Discussion

I see it as my job to motivate and set an example for the students I mentor. This includes promoting the scientific method, remaining sensitive to issues of integration and equity, and putting the common interest ahead of personal ambition.

I strongly believe that developing a trustworthy AI is one of the key issues for the coming years. It is in any case necessary if its development is to take place in a democratic context. As a civil servant, participating in creating this trust between citizens and technology is a primary objective that I wish to follow in the years to come.

This is all the more true in a scientific context that is constantly under pressure from funding sources guided by simplistic indicators, which may ultimately question the meaning of the researcher's profession. The field of AI is developing fast, maybe too fast sometimes to ensure reliability or reproducibility. This is why since I obtained my position in 2013, I have always sought to push open science by publishing codes on github, early preprints on ArXiv, and sharing the expertise I acquired through popularization communications.

I have been very fortunate to have bright and motivated students, and it is with immense pride that I see them taking on responsibilities in the companies and academic circles they work with. I hope that by obtaining my HDR, I will be able to continue this exciting work and give it an even greater impact.

Bibliography

[1] Claude Berrou, Olivier Dufor, Vincent Gripon, and Xiaoran Jiang. Information, noise, coding, modulation: What about the brain? In *Proceedings of the 8th symposium on Turbo Codes and Iterative Information Processing*, pages 167–172, August 2014.

[2] A. Mheich, M. Hassan, V. Gripon, O. Dufor, M. Khalil, C. Berrou, and F. Wendling. A novel algorithm for measuring graph similarity: application to brain networks. In *Proceedings of the IEEE EMBS Neural Engineering Conference*, pages 1068–1071, April 2015.

[3] Ahmad Mheich, Mahmoud Hassan, Mohamad Khalil, Vincent Gripon, Olivier Dufor, and Fabrice Wendling. Siminet: a novel method for quantifying brain network similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2238–2249, September 2018.

[4] A. Mheich, M. Hassan, F. Wendling, M. Khalil, O. Dufor, V. Gripon, and C. Berrou. Simnet: A new algorithm for measuring brain networks similarity. In *Proceedings of the ICABME international conference*, pages 119–122, 2015.

[5] Mathilde Ménoret, Nicolas Farrugia, Bastien Pasdeloup, and Vincent Gripon. Evaluating graph signal processing for neuroimaging through classification and dimensionality reduction. In *Proceedings of GlobalSip*, 2017. To appear.

[6] Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Transfer incremental learning using data augmentation. *Applied Sciences*, 8(12), 2018.

[7] Myriam Bontonou, Carlos Lassance, Vincent Gripon, and Nicolas Farrugia. Comparing linear structure-based and data-driven latent spatial representations for sequence prediction. In *Wavelets and Sparsity XVIII*, San Diego, USA, August 2019. To appear.

[8] Myriam Bontonou, Carlos Lassance, Jean-Charles Vialatte, and Vincent Gripon. A unified deep learning formalism for processing graph signals. In *SDM Special Session on Graph Neural Networks*, May 2019.

[9] Zhe Yao, Vincent Gripon, and Michael Rabbat. A gpu-based associative memory using sparse neural networks. In *Proceedings of the PCNN-14 conference*, pages 688–692, July 2014.

[10] Philippe Tigréat, Carlos Rosar Kos Lassance, Xiaoran Jiang, Vincent Gripon, and Claude Berrou. Assembly output codes for learning neural networks. In *Proceedings of the 9th International Symposium on Turbo Codes and Iterative Information Processing*, pages 285–289, September 2016.

[11] Eliott Coyac, Vincent Gripon, and Charlotte Langlais. Impact du bruit synaptique sur les performances des réseaux de cliques neurales. In *Proceedings of the GRETSI conference*, 2015.

[12] Eliott Coyac, Vincent Gripon, Charlotte Langlais, and Claude Berrou. Performance of neural clique networks subject to synaptic noise. In *Proceedings of Cognitive*, pages 4–9, February 2017.

[13] Eliott Coyac, Vincent Gripon, Charlotte Langlais, and Claude Berrou. Distributed coding and synaptic pruning. In *Proceedings of the 9th International Symposium on Turbo Codes and Iterative Information Processing*, pages 206–210, September 2016.

[14] Eliott Coyac, Vincent Gripon, Charlotte Langlais, and Claude Berrou. Robust associative memories naturally occuring from recurrent hebbian networks under noise. In *Arxiv Preprint*, September 2017.

[15] Vincent Gripon, Ghouthi Boukli Hacene, Matthias Löwe, and Franck Vermet. Improving accuracy of nonparametric transfer learning via vector segmentation. In *proceedings of ICASSP*, pages 2966–2970, April 2018.

[16] Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Incremental learning on chip. In *Proceedings of GlobalSip*, 2017. To appear.

[17] Ghouthi Boukli Hacene, Carlos Lassance, Vincent Gripon, Matthieu Courbariaux, and Yoshua Bengio. Attention based pruning for shift networks. In *Arxiv Preprint*, 2019.

[18] Vincent Gripon, Antonio Ortega, and Benjamin Girault. An inside look at deep neural networks using graph signal processing. In *Proceedings of ITA*, February 2018.

[19] Myriam Bontonou, Carlos Lassance, Ghouthi Boukli Hacene, Vincent Gripon, Jian Tang, and Antonio Ortega. Introducing graph smoothness loss for training deep learning architectures. In *Data Science Workshop*, pages 160–164, June 2019.

[20] Carlos Lassance, Vincent Gripon, Jian Tang, and Antonio Ortega. Structural robustness for deep learning architectures. In *Data Science Workshop*, pages 125–129, June 2019.

[21] Bastien Pasdeloup, Vincent Gripon, Réda Alami, and Michael Rabbat. *Uncertainty Principle on Graphs*. Vertex-Frequency Analysis of Graph Signals. Springer Nature, April 2019.

[22] Ahmet Iscen, Teddy Furon, Vincent Gripon, Michael Rabbat, and Hervé Jégou. Memory vectors for similarity search in high-dimensional spaces. *IEEE Transactions on Big Data*, pages 65–77, 2018.

[23] Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, Dominique Pastor, and Michael Rabbat. Characterization and inference of graph diffusion processes from observations of stationary signals. *IEEE Transactions on Signal and Information Processing over Networks*, 4(3):481–496, September 2018.

[24] Xiaoran Jiang, Vincent Gripon, Claude Berrou, and Michael Rabbat. Storing sequences in binary tournament-based neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 27(5):913–925, 2016.

[25] Francois Leduc-Primeau, Vincent Gripon, Michael Rabbat, and Warren J. Gross. Fault-tolerant associative memories based on c-partite graphs. *IEEE Transactions on Signal Processing*, 64(4):829–841, 2015.

[26] Bastien Pasdeloup, Michael Rabbat, Vincent Gripon, Dominique Pastor, and Grégoire Mercier. Graph reconstruction from the observation of diffused signals. In *Proceedings of the 53rd Allerton Conference*, pages 1386–1390, October 2015.

[27] Vincent Gripon Bastien Pasdeloup, Reda Alami and Michael Rabbat. Toward an uncertainty principle for weighted graphs. In *Proceedings of the 23rd European Signal Processing Conference*, pages 1496–1500, July 2015.

[28] Vincent Gripon, Vitaly Skachek, and Michael Rabbat. Sparse binary matrices as efficient associative memories. In *Proceedings of the 52nd Allerton conference*, pages 499–504, October 2014.

[29] Michael Rabbat and Vincent Gripon. Towards a spectral characterization of signals supported on small-world networks. In *ICASSP*, pages 4793–4797, May 2014.

[30] Francois Leduc-Primeau, Vincent Gripon, Michael Rabbat, and Warren Gross. Cluster-based associative memories built from unreliable storage. In *ICASSP*, pages 8370–8374, May 2014.

[31] Vincent Gripon, Vitaly Skachek, and Michael G. Rabbat. Sparse structured associative memories as efficient set-membership data structures. In *Proceedings of the 51st Allerton conference*, pages 500–505, October 2013.

[32] Vincent Gripon and Michael Rabbat. Maximum likelihood associative memories. In *Proceedings of Information Theory Workshop*, pages 1–5, September 2013.

[33] Vincent Gripon and Michael Rabbat. Reconstructing a graph from path traces. In *Proceedings of International Symposium on Information Theory*, pages 2488–2492, July 2013.

[34] Vincent Gripon, Michael Rabbat, Vitaly Skachek, and Warren J. Gross. Compressing multisets using tries. In *Proceedings of Information Theory Workshop*, pages 647–651, Lausanne, Switzerland, September 2012.

[35] Vincent Gripon, Vitaly Skachek, Warren J. Gross, and Michael Rabbat. Random clique codes. In *Proceedings of 7" International Symposium on Turbo Codes and Iterative Information Processing*, pages 121–125, Gothenburg, Sweden, August 2012.

[36] Hooman Jarollahi, Vincent Gripon, Naoya Onizawa, and Warren J. Gross. Algorithm and architecture for a low-power content-addressable memory based on sparse-clustered networks. *Transactions on Very Large Scale Integration Systems*, 27(2):375–387, 2016.

[37] Robin Danilo, Vincent Gripon, Philippe Coussy, Laura Conde-Canencia, and Warren J. Gross. Restricted clustered neural network for storing real data. In *proceedings of GLSVLSI conference*, pages 205–210, May 2015.

[38] Robin Danilo, Homman Jarollahi, Vincent Gripon, Philippe Coussy, Laura Conde-Canencia, and Warren J. Gross. Algorithm and implementation of an associative memory for oriented edge detection using improved clustered neural networks. In *Proceedings of ISCAS Conference*, pages 2501–2504, May 2015.

[39] Hooman Jarollahi, Naoya Onizawa, Vincent Gripon, Noboru Sakimura, Tadahiko Sugibayashi, Tetsuo Endoh, Hideo Ohno, Takahiro Hanyu, and Warren J. Gross. A non-volatile associative memory-based context-driven search engine using 90 nm cmos mtj-hybrid logic-in-memory architecture. *Journal on Emerging and Selected Topics in Circuits and Systems*, 4:460–474, 2014.

[40] Hooman Jarollahi, Naoya Onizawa, Vincent Gripon, and Warren J. Gross. Algorithm and architecture of fully-parallel associative memories based on sparse clustered networks. *Journal of Signal Processing Systems*, pages 1–13, 2014.

[41] Hooman Jarollahi, Naoya Onizawa, Vincent Gripon, Takahiro Hanyu, and Warren J. Gross. Algorithm and architecture for a multiple-field context-driven search engine using fully-parallel clustered associative memories. In *Proceedings of SiPS*, pages 1–6, October 2014.

[42] Hooman Jarollahi, Vincent Gripon, Naoya Onizawa, and Warren J. Gross. A low-power content-adressable-memory based on clustered-sparse-networks. In *Proceedings of 24th International Conference on Application-specific Systems, Architectures and Processors*, pages 642–653, June 2013.

[43] Hooman Jarollahi, Naoya Onizawa, Vincent Gripon, and Warren J. Gross. Reduced-complexity binary-weight-coded associative memories. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 2523–2527, May 2013.

[44] Hooman Jarollahi, Naoya Onizawa, Vincent Gripon, and Warren J. Gross. Architecture and implementation of an associative memory using sparse clustered networks. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 2901–2904, May 2012.

[45] Vincent Gripon, Matthias Löwe, and Franck Vermet. Associative memories to accelerate approximate nearest neighbor search. *Applied Sciences*, September 2018.

[46] Vincent Gripon, Judith Heusel, Matthias Löwe, and Franck Vermet. A comparative study of sparse associative memories. *Journal of Statistical Physics*, 164:105–129, 2016.

[47] Théo Giraudon, Vincent Gripon, Matthias Löwe, and Franck Vermet. Towards an intrinsic definition of robustness for a classifier. *arXiv preprint arXiv:2006.05095*, 2020.

[48] Ghouthi Boukli Hacene, Vincent Gripon, Matthieu Arzel, Nicolas Farrugia, and Yoshua Bengio. Quantized guided pruning for efficient hardware implementations of convolutional neural networks. In *Arxiv Preprint*, 2018.

[49] Ghouthi Boukli Hacene, Vincent Gripon, Matthieu Arzel, Nicolas Farrugia, and Yoshua Bengio. Quantized guided pruning for efficient hardware implementations of deep neural networks. In *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 206–209. IEEE, 2020.

[50] Miloš Nikolić, Matthieu Courbariaux, Ghouthi Boukli Hacene, Vincent Gripon, Yoshua Bengio, and Andreas Moshovos. Learning bit-widths for aggressive and accurate quantization. *arXiv preprint*, 2019.

[51] Carlos Lassance, Myriam Bontonou, Ghouthi Boukli Hacene, Vincent Gripon, Jian Tang, and Antonio Ortega. Deep geometric knowledge distillation with graphs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8484–8488. IEEE, 2020.

[52] Carlos Lassance, Vincent Gripon, Jian Tang, and Antonio Ortega. Robustesse structurelle des architectures d'apprentissage profond. In *GRETSI*, 2020.

[53] Yibo Yang, Zhisheng Zhong, Tiancheng Shen, and Zhouchen Lin. Convolutional neural networks with alternately updated clique. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2422, 2018.

[54] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[55] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[56] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311. IEEE Computer Society, 2010.

[59] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[60] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 370–377. IEEE, 2005.

[61] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[62] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[63] Diek W Wheeler, Charise M White, Christopher L Rees, Alexander O Komendantov, David J Hamilton, and Giorgio A Ascoli. Hippocampome. org: a knowledge base of neuron types in the rodent hippocampus. *Elife*, 4:e09960, 2015.

[64] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328, 1979.

[65] Pascal Fries, John H Reynolds, Alan E Rorie, and Robert Desimone. Modulation of oscillatory neuronal synchronization by selective visual attention. *Science*, 291(5508):1560–1563, 2001.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[67] John R Anderson and Gordon H Bower. *Human associative memory*. Psychology press, 2014.

[68] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[69] Vincent Gripon and Olivier Serre. Qualitative concurrent stochastic games with imperfect information. In Springer, editor, *Proceedings of 36th International*

Colloquium of Automata, Languages and Programming, Lecture Notes in Computer Science, pages 200–211, Rhodes, Greece, July 2009.

[70] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

[71] Vincent Gripon. Tropical graph signal processing. In *Proceedings of the Asilomar conference*, October 2017. To appear.

[72] Benjamin Girault, Antonio Ortega, and Shrikanth S Narayanan. Irregularity-aware graph fourier transforms. *IEEE Transactions on Signal Processing*, 66(21):5746–5761, 2018.

[73] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[74] Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. In *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pages 478–504. Springer, 1985.

[75] Ameya Agaskar and Yue M Lu. Uncertainty principles for signals defined on graphs: Bounds and characterizations. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3493–3496. IEEE, 2012.

[76] Ameya Agaskar and Yue M Lu. A spectral graph uncertainty principle. *IEEE Transactions on Information Theory*, 59(7):4338–4356, 2013.

[77] Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, and Dominique Pastor. Vers une caractérisation de la courbe d'incertitude pour des graphes portant des signaux. In *Proceedings of the GRETSI conference*, 2015.

[78] Bastien Pasdeloup. *Extending convolutional neural networks to irregular domains through graph inference*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique, 2017.

[79] Nicolas Grelier, Bastien Pasdeloup, Jean-Charles Vialatte, and Vincent Gripon. Neighborhood-preserving translations on graphs. In *Proceedings of GlobalSIP*, pages 410–414, October 2016.

[80] Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929, 2016.

[81] Arthur P Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

[82] Peter J Bickel, Elizaveta Levina, et al. Covariance regularization by thresholding. *The Annals of Statistics*, 36(6):2577–2604, 2008.

[83] Brenden Lake and Joshua Tenenbaum. Discovering structure by learning sparse graphs. 2010.

[84] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

[85] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):467–483, 2017.

[86] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[87] Tristan Stérin, Nicolas Farrugia, and Vincent Gripon. An intrinsic difference between vanilla rnns and gru models. In *Proceedings of Cognitive*, pages 76–81, February 2017.

[88] Olaf Sporns. The future of network neuroscience, 2017.

[89] Weiyu Huang, Leah Goldsberry, Nicholas F Wymbs, Scott T Grafton, Danielle S Bassett, and Alejandro Ribeiro. Graph frequency analysis of brain signals. *IEEE journal of selected topics in signal processing*, 10(7):1189–1203, 2016.

[90] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186, 2009.

[91] James V Haxby, M Ida Gobbini, Maura L Furey, Alumit Ishai, Jennifer L Schouten, and Pietro Pietrini. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430, 2001.

[92] Ala Aboudib, Vincent Gripon, and Gilles Coppin. A biologically inspired framework for visual information processing and an application on modeling bottom-up visual attention. *Cognitive Computation*, pages 1–20, September 2016.

[93] Ala Aboudib, Vincent Gripon, and Gilles Coppin. A model of bottom-up visual attention using cortical magnification. In *Proceedings of ICASSP*, pages 1493–1497, April 2015.

[94] Anthony Randal McIntosh. Towards a network theory of cognition. *Neural Networks*, 13(8-9):861–870, 2000.

[95] L da F Costa, Francisco A Rodrigues, Gonzalo Travieso, and Paulino Ribeiro Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in physics*, 56(1):167–242, 2007.

[96] Nicolas Grelier, Carlos Rosar Kos Lassance, Elsa Dupraz, and Vincent Gripon. Graph-projected signal processing. In *IEEE GlobalSIP*, 2018. To appear.

[97] T Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77, 1973.

[98] Abraham Lempel. On multiset decipherable codes (corresp.). *IEEE transactions on information theory*, 32(5):714–716, 1986.

[99] Lav R Varshney and Vivek K Goyal. Toward a source coding theory for sets. In *Data Compression Conference (DCC'06)*, pages 13–22. IEEE, 2006.

[100] Lav R Varshney and Vivek K Goyal. Ordered and disordered source coding. *sort*, 1:1, 2006.

[101] Eun Bae Kong and Thomas G Dietterich. Error-correcting output coding corrects bias and variance. In *Machine Learning Proceedings 1995*, pages 313–321. Elsevier, 1995.

[102] Claude Berrou and Vincent Gripon. Coded hopfield networks. In *Proceedings of 6" International Symposium on Turbo Codes and Iterative Information Processing*, pages 1–5, Brest, France, September 2010.

[103] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960, 1969.

[104] Vincent Gripon and Claude Berrou. A simple and efficient way to store many messages using neural cliques. In *Proceedings of IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, pages 54–58, Paris, France, April 2011.

[105] Vincent Gripon and Claude Berrou. Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks*, 22(7):1087–1096, July 2011.

[106] Vincent Gripon and Claude Berrou. Nearly-optimal associative memories based on distributed constant weight codes. In *Proceedings of Information Theory and Applications Workshop*, pages 269–273, San Diego, CA, USA, February 2012.

[107] Ala Aboudib, Vincent Gripon, and Xiaoran Jiang. A study of retrieval algorithms of sparse messages in networks of neural cliques. In *Proceedings of Cognitive 2014*, pages 140–146, May 2014.

[108] Stephen Larroque, Ehsan Sedgh Gooya, Vincent Gripon, and Dominique Pastor. Using tags to improve diversity of sparse associative memories. In *Proceedings of Cognitive*, pages 1–7, March 2015.

[109] Vincent Gripon and Xiaoran Jiang. Mémoires associatives pour observations floues. In *Proceedings of XXIV-th Gretsi seminar*, September 2013.

[110] Behrooz Kamary Aliabadi, Claude Berrou, Vincent Gripon, and Xiaoran Jiang. Storing sparse messages in networks of neural cliques. *IEEE Transactions on Neural Networks and Learning Systems*, 25:980–989, 2014.

[111] Bartosz Boguslawski, Vincent Gripon, Fabrice Seguin, and Frédéric Heitzmann. Twin neurons for efficient real-world data distribution in networks of neural cliques. applications in power management in electronic circuits. *IEEE Transactions on Neural Networks and Learning Systems*, 27(2):375–387, 2016.

[112] Robin Danilo, Vincent Gripon, Philippe Coussy, and Laura Conde-Canencia. RÃ©seaux de clusters de neurones restreints. In *Proceedings of the GRETSI conference*, 2015.

[113] Chendi Yu, Vincent Gripon, Xiaoran Jiang, and Hervé Jégou. Neural associative memories as accelerators for binary vector search. In *Proceedings of Cognitive*, pages 85–89, March 2015.

[114] Demetrio Ferro, Vincent Gripon, and Xiaoran Jiang. Nearest neighbour search using binary neural networks. In *Proceedings of IJCNN*, pages 5106–5112, July 2016.

[115] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[116] Ehsan Sedgh Gooya, Dominique Pastor, and Vincent Gripon. Automatic face recognition using sift and networks of tagged neural cliques. In *Proceedings of Cognitive*, pages 57–61, March 2015.

[117] Ala Aboudib, Vincent Gripon, and Baptiste Tessiau. Implementing relational-algebraic operators for improving cognitive abilities in networks of neural cliques. In *Proceedings of Cognitive*, pages 36–41, March 2015.

[118] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[119] Simon B Laughlin and Terrence J Sejnowski. Communication in neuronal networks. *Science*, 301(5641):1870–1874, 2003.

[120] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

[121] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[122] Claude Berrou and Vincent Gripon. *Petite mathÃ©matique du cerveau*. September 2012.

[123] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[124] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in neural information processing systems*, pages 892–900, 2016.

[125] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[127] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[128] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[129] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[130] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *arXiv preprint arXiv:1711.08141*, 2017.

[131] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. *arXiv preprint arXiv:1806.07370*, 2018.

[132] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[133] Kohei Yamamoto and Kurato Maeno. Pcas: Pruning channels with attention statistics. *arXiv preprint arXiv:1806.05382*, 2018.

[134] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[135] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[136] Guillaume Soulié, Vincent Gripon, and Maëlys Robert. Compression of deep neural networks on the fly. *Lecture Notes in Computer Science*, 9887:153–170, September 2016.

[137] Quentin Jodelet, Vincent Gripon, and Masafumi Hagiwara. Transfer learning with sparse associative memories. In *International Conference on Artificial Neural Networks*, 2019. To appear.

[138] Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Finding all matches in a database using binary neural networks. In *Proceedings of Cognitive*, pages 59–64, February 2017.

[139] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[140] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[141] Nicolas Papernot and Patrick D. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, abs/1803.04765, 2018.

[142] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[143] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863, 2017.

[144] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[145] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[146] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[147] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[148] Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks. *arXiv preprint arXiv:1511.02580*, 2015.

[149] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[150] Carlos Eduardo Rosar Kos Lassance, Jean-Charles Vialatte, and Vincent Gripon. Matching convolutional neural networks without priors about data. In *Proceedings of Data Science Workshop*, 2018. Submitted to.

[151] Jean-Charles Vialatte, Vincent Gripon, and Grégoire Mercier. Generalizing the convolution operator to extend cnns to irregular domains. *arXiv preprint arXiv:1606.01166*, 2016.

[152] Jean-Charles Vialatte, Vincent Gripon, and Gilles Coppin. Learning local receptive fields and their weight sharing scheme on graphs. In *Proceedings of GlobalSip*, 2017.

[153] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[154] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[155] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[156] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[157] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[158] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[159] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[160] Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Transfer incremental learning using data augmentation. *Applied Sciences*, 8(12):2512, 2018.

[161] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[162] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[163] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135, 2017.

[164] Vincenzo Lomonaco. Why Continuous Learning is the Key Towards Machine Intelligence. `https://medium.com/continual-ai/why-continuous-learning-is-the-key-towards-machine-intelligence-1851cb57c308`, 2017.

[165] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[166] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[167] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[168] Animesh Koratana, Daniel Kang, Peter Bailis, and Matei Zaharia. Lit: Learned intermediate representation training for model compression. In *International Conference on Machine Learning*, pages 3509–3518, 2019.

[169] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[170] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based few-shot learning. *arXiv preprint arXiv:2006.03806*, 2020.

[171] Myriam Bontonou, Louis Béthune, and Vincent Gripon. Predicting the accuracy of a few-shot classifier. *arXiv preprint arXiv:2007.04238*, 2020.

[172] Lyes Khacef, Vincent Gripon, and Benoit Miramond. Gpu-based self-organizing maps for post-labeled few-shot unsupervised learning. *arXiv preprint arXiv:2009.03665*, 2020.

[173] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Exploiting unsupervised inputs for accurate few-shot classification. *arXiv preprint arXiv:2001.09849*, 2020.

[174] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[175] Kwang-Sung Jun and Robert Nowak. Graph-based active learning: A new look at expected error minimization. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1325–1329. IEEE, 2016.

[176] Carlos Lassance, Vincent Gripon, and Gonzalo Mateos. Graph topology inference benchmarks for machine learning. *arXiv preprint arXiv:2007.08216*, 2020.

[177] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. Graph topology inference based on sparsifying transform learning. *IEEE Transactions on Signal Processing*, 67(7):1712–1727, 2019.

[178] Quoc V. Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 265–272, 2011.

[179] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[180] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11, 2017.

[181] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[182] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018.

[183] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.