On the use of neural networks to solve problems. From multilayer perceptron to deep learning and reservoir computing

On the use of neural networks to solve problems

Michel Salomon

HABILITATION À DIRIGER DES RECHERCHES

de l'Université de Franche-Comté

préparée au sein de l'Univ. Bourgogne Franche-Comté (UBFC)

Spécialité : **Informatique**

présentée par

MICHEL SALOMON

On the use of neural networks to solve problems. From multilayer perceptron to deep learning and reservoir computing

On the use of neural networks to solve problems

Soutenue publiquement le 26 octobre 2018 devant le Jury composé de :

CHRISTOPHE CERISARA	Reviewer	CNRS researcher (HDR) at LORIA, France
CHRISTOPHE GARCIA	Reviewer	Professor at INSA de Lyon, France
DAVID ELIZONDO	Reviewer	Professor at De Montfort University, UK
STÉPHANE CHRÉTIEN	Examinator	Senior research scientist (HDR) at NPL, UK
SYLVAIN CONTASSOT-VIVIER	Examinator	Professor at Université de Lorraine, France
RAPHAËL COUTURIER	Examinator	Professor at Université de Franche-Comté, France

Curriculum Vitæ and Overview of Activities

This Habilitation manuscript focuses on research works done in the field of neural networks and more particularly on the use of different kinds of architectures to solve problems. However, I have also addressed problems taking place in other domains such as wireless sensor networks. Therefore, to show that I have acquired a very different set of skills in different fields of study, to start this chapter gives a brief curriculum vitæ and overwiew of the activities completed since I became an assistant professor in computer science at the University of Franche-Comté. A chronological presentation of the different research works, past and present, is made, describing the covered topics and the resulting publications which are grouped by publication types at the end.

Curriculum vitæ

- Academic position
  - September 2002 - date → Assistant Professor in Computer Science at University Institute of Technology of Belfort-Montbéliard (IUT Belfort-Montbéliard), part of the University of Franche-Comté.
- Past-academic positions
  - 2001 - 2002 → Full-time Temporary Teaching and Research Assistant (ATER) at Louis Pasteur University of Strasbourg.
  - 2000 - 2001 → Half-time Temporary Teaching and Research Assistant (ATER) at Louis Pasteur University of Strasbourg.
- Education
  - 2001 → PhD in Computer Science at Louis Pasteur University of Strasbourg.
  - 1996 → MSc in Computer Science (Diplôme d'Études Approfondies - 5 year program) at Louis Pasteur University of Strasbourg.

External links

- Personal professional website
- Google Scholar
- ResearchGate

Institutional activity

- September 2011 - August 2014 → Head of the Computer Science Department at University Institute of Technology of Belfort-Montbéliard (IUT Belfort-Montbéliard).
- Since 2008 → Elected member of the Department Council (Conseil de Département), as teacher representative, during each academic year.

Research activity

My research activities have entirely been carried out as a member of the AND team (Algorithmique Numérique Distribuée), a part of the DISC department (Département Informatique des Systèmes Complexes) which is the Computer Science department of FEMTO-ST Institute. Over the past 15 years I have worked on various topics, going back and forth between some of them:

- I begun to work on discrete dynamical systems and high performance computing;
- Then I investigated the use of optimization methods, first to improve the lifetime of Wireless Sensor Networks (WSNs) and second to predict the evolution of genomes in bioinformatics;
- Finally, I studied the use of artificial neural networks to solve different prediction and classification tasks. From 2008 up to 2012, the considered network architecture was the multilayer perceptron, while since 2015 I am interested in deep learning ones and Reservoir Computing.

More details on the different research works, presented in their chronological order, are given thereafter. When a work has been completed during a PhD thesis, or a Master of Science internship, I precise it as well as any resulting publication.

1/ 2002 - 2005 / Discrete dynamical systems and high performance computing

I studied from a theoretical point of view the asynchronous mode of discrete-state discrete-time dynamic networks. The asynchronous mode allows to obtain smaller global computation time on clusters that communicate together through dynamic links, typically distant, but it can lead to unstable behaviors due to cycles in the connection graph. The main result, issued from the internship of a Master Science student is the proposal, published in [19], of a new class of discrete dynamic networks mixing synchronism and asynchronism.

In the high performance computing topic, I took part in the development of parallel synchronous and asynchronous Newton multisplitting algorithms, in order to efficiently solve a 3D transport model using geographically distributed computers. In fact, thanks to the multisplitting method and overlapping techniques, the number of iterations may be reduced and consequently lead to reduced computation times. Algorithms were implemented with a grid enabled Java Asynchronous Computing Environment and evaluated on a heterogeneous grid environment. [18] and [20] are the resulting publications.

The first year of this period of time I also completed two publications presenting the main contributions of my PhD thesis. In this dissertation, I addressed the parallelization of several metaheuristics for combinatorial optimization in the context of medical image registration, showing the suitability of the data parallel programming paradigm. First, for the rigid matching in chapter book [2] is showed that a data parallel implementation of differential evolution can yield subvoxel accuracy and exhibit an almost linear speedup. Second, for the deformable matching in [19] is described a massively parallel approach via stochastic differential equations giving computation times compatible with a clinical use. In both cases the addressed problem was 3D Magnetic Resonance Image (MRI) monomodal registration.

2/ 2003 - 2007 / First work on wireless sensor networks

During these years, I co-supervised with Jacques M. Bahi a PhD thesis about the application of concepts of asynchronous computing framework to WSNs. An approach allowing to improve the lifetime of heterogeneous WSNs and Mobile Ad hoc NETworks (MANETs) by distributing the computing tasks according to the remaining energy in the different nodes has been proposed and evaluated using the discrete-event simulator OMNeT++. The background idea was to formulate the problem as an energy-controlled load balancing problem so that each node would cooperate in proportion to its remaining energy. Three publications are issued from this work [17, 17, 18].

3/ 2008 - 2012 / Solving problems with multilayer perceptrons

In 2008 I chose to focus my research activity on the multilayer perceptron to extend a collaboration with the IRMA (Informatics and Radiation physics Medical Applications) team from the Chrono-Environnement laboratory of the University of Franche-Comté. This team works more particularly on dosimetry [2, 13], radiophysics, with external radiation therapy as case study. The latter is the most common type of radiation therapy (or radiotherapy) used for cancer treatment. As a result, I co-supervised a PhD thesis on the use of the multilayer perceptron to predict lung respiratory motion. Indeed, being able to manage respiratory motion is very important in lung cancer radiotherapy. Numerous publications [5, 9, 10, 13, 14, 15, 1, 16] and a software patent [1] assess the high quality of this work. Furthermore, other publications [7, 16] dealing with the use of a neural network for case-based reasoning and another patent [2] are directly linked to it. Considering still this medical physics context, I have investigated in [1, 15, 16] different approaches to improve the building and training of a multilayer perceptron.

During the last year of the PhD thesis and after its defense in october 2011, other kinds of prediction problems have received my attention. The construction and evaluation of chaotic networks, just as the prediction of chaos with multilayer networks, were the subject of research works published in [12, 14]. A case study was protein folding in the 2D model that has been shown to be chaotic in [11]. In [8, 12], I examined with other colleagues whether a neural network is a relevant solution for active Micro-Electro-Mechanical Systems (MEMS) based aerodynamic flow control, and more precisely if a multilayer perceptron is able to predict the evolution of the force acting on a backward-facing step (a rough model of the rear of a car) due to the MEMS effect on the turbulent airflow over the step.

4/ 2012 - 2015 / Second work on wireless sensor networks and bioinformatics

In late 2012 I worked once again on wireless sensor networks and began an activity in bioinformatics due to my involvement in two PhD thesis as co-supervisor.

The first one dealt with the design of distributed coverage optimization techniques for improving lifetime of wireless sensor networks. Different distributed protocols based on a partitioning of the area of interest to be monitored in sub-regions have been thus proposed [1, 4, 6, 11]. The main idea in each of these protocols was to periodically elect in a sub-region a leader node that is in charge of computing disjoint cover sets for one or several monitoring rounds, where the cover sets are obtained by solving an integer programming problem. From a global area of interest point of view, the different protocols are distributed, but locally in each sub-region they are centralized. As in the previous works on WSNs, the performance of each protocol is assessed with the simulator OMNeT++, using the GLPK solver to find a solution to the optimization problem.

The second thesis faced the problem of the reconstruction of ancestral DNA sequences, using as case study 450 chloroplast genomes. These organelles are nucleotide chains of intermediate size that have undergone very little recombination over time. To solve this problem, a pipeline of methods allowing to obtain a well-supported phylogenetic tree, in accordance with the largest number of genes, from a set of unannotated genomes has been proposed. These methods combined tools from high performance computing, nature-inspired optimization, and bio-statistics. For the phylogenetic prediction stage that tries to build the most supported tree from a set of core genomes, genetic algorithm and particle swarm optimization were applied. The large number of papers presented in international conferences, directly issued from this thesis [4, 6, 7, 8, 9, 10], or from ongoing research works [2, 2, 3] in which I have a minor activity, shows the relevance of the proposal.

5/ Since 2015 / Reservoir computing and deep learning

A couple of years ago I had a renewed interest in neural network architectures, mainly for two reasons. The first one is a novel brain-inspired computing concept called Reservoir Computing, which has recently been explored experimentally through time delayed feedback nonlinear dynamics by colleagues of the OPTO team from FEMTO-ST Institute's OPTICS department. Indeed, first experiments showed that such a hardware implementation was able to recognize simple speech (spoken digits) at the rate of 1 million words per second. Therefore, I decided to study the Reservoir Computing paradigm, focusing on its representative in machine learning: the Echo State Network model. The second reason is the rise of deep learning over the past few years, with an increasing number of architectures exhibiting impressive performances on various classification and prediction problems. More specifically, I have investigated architectures in order to solve problems already studied in the AND team, namely image steganalysis in spatial domain and more recently image denoising.

The research work on Reservoir Computing started at the beginning of 2015 thanks to two Master internships. The first student has thus developed a multicore version of a Matlab program (designed by Pr. Laurent Larger, a colleague of the OPTO team) simulating an electro-optical hardware implementation of Reservoir Computing in order to reduce the computation time, while the second studied the detection performance of Echo State Networks on a well-known image recognition problem: the MNIST handwriting benchmark.

The results obtained at the end of these internships have been presented, respectively, in the Workshop Dynamical Systems and Brain-inspired Information Processing organized by the Laboratory of Mathematics of Besançon in November 2015 [5] and at an international conference in august 2016 [5].

Today, deep learning is known to be the provider of state-of-the-art solutions for many image problems and, among the available architectures, convolutional neural networks usually represent the best choice for image classification. Therefore, as image steganalysis is such a problem, since we want to distinguish images embedding a hidden message from innocent ones, considering embedding schemes working in spatial domain, the objective was to find an approach allowing to obtain better detection performance than the currently available conventional Spatial Rich Models (SRM)+Ensemble Classifier (EC) steganalyzers and CNN-based ones. The first idea was to design a new convolutional neural network, unfortunately we did not succeed [3, 4]. In fact, like all previous research works on CNNs for image steganalysis in spatial domain, the proposal was only competitive when the stego key controlling the embedding process remains the same. Obviously, having a single key is not desirable, since it weakens the security of the steganographic method performing the embedding. Despite this negative result, we were then able to obtain a major result [1] by paying a particular attention to the characteristic failure of the first competitive CNN-based steganalyzer with the conventional ones proposed by Xu *et al.* in 2016. A careful study of the detection performance of this CNN that does not suffer from the single key limitation allowed to propose a criterion to choose either the CNN or the combination SRM+EC for an input image. This very satisfactory result has been presented in a international conference [1] and two invited national conferences [2, 3], it has also encouraged us to continue with the exploration of deep learning methods for image denoising and medical image analysis. I have also been invited in march 2018 by the students of the CMI Bachelor of Computer Science from the University of Franche-Comté to give an oral presentation introducing the field of artificial intelligence during their R&D Day [1].

PhD and Master thesis supervision

- Co-supervisor of 4 PhD students
  - 2012.12 - 2015.12 / Bassam Alkindy
    Dissertation: “Combining Approaches for Predicting Genomic Evolution”
    Placement: Assistant Professor at University of Mustansiriyah, Baghdad, Iraq
  - 2012.09 - 2015.10 / Ali Kahadum Idrees
    Dissertation: “Distributed Coverage Optimization Techniques for Improving Lifetime of Wireless Sensor Networks”
    Placement: Assistant Professor at University of Babylon, Iraq
  - 2008.09 - 2011-09 / Rémy Laurent
    Dissertation: “Simulation du mouvement pulmonaire personnalisé par réseau de neurones artificiels pour la radiothérapie externe”
    Placement: Works in a private company
  - 2003.09 - 2007.06 / Amine Abbas
    Dissertation: “Optimisation de la durée de vie d’un réseau de capteurs”
    Placement: Works in a private company

- Co-supervisor of 7 Master students
  - 2018.02 - 2018.09 / David Roche

    Dissertation: “Deep learning pour la conception d’une intelligence artificielle pour un jeu abstrait”
  - 2018.02 - 2018.08 / Pierre Feilles

    Dissertation: “Étude et mise en œuvre d’une approche de type deep learning pour segmenter automatiquement le myocarde”
  - 2017.02 - 2017.09 / Pierre Primet

    Dissertation: “Étude et mise en œuvre d’une approche de type deep learning comme outil d’aide à la détection d’une pathologie”
  - 2015.02 - 2015.09 / Nils Schaetti

    Dissertation: “Reservoir Computing : Étude théorique et pratique en reconnaissance de chiffres manuscrits”

    Placement: PhD student at Computer Science department (IIUN) of the Université de Neuchâtel, Switzerland
  - 2015.02 - 2015.09 / Jad Nassar

    Dissertation: “Parallelization of a Simulation Code for Neuromorphic Computing”

    Placement: PhD student at INRIA FUN Team, Lille
  - 2010.02 - 2010.10 / Arnaud Aubert

    Dissertation: “Contrôle actif d’écoulements aérodynamiques par réseaux de neurones”
  - 2003.02 - 2003.09 / Amine Abbas

    Dissertation: “Mixage synchronisme/asynchronisme dans les réseaux d’automates finis discrets”

Research projects

Over the past years I was involved in writing proposals for several calls, in particular calls for proposals from the French National Research Agency (ANR). Unfortunately in this latter case none has been accepted.

- ANR Calls for proposals
  - Coordinator of the APACHE project which was submitted to the 2013 Specific Support for Defence Research Projects and Innovation (ASTRID) program.

    The objective of this project was to design an approach using Acoustic PUFs (Physical Unclonable Functions) to Authenticate Computer Hardware and Electronics. APACHE was labelled by the Pole Véhicule du Futur, French Pôle de Compétitivité.
  - Member of the JCJC (coordinated by a young researcher) MAPGEN project which was submitted to 2012 call for proposals.

    This project was planned to study mathematical and computing models able to simulate genome dynamics in evolution, taking into account gene mutations and genomic rearrangements.

  - Member of the CrypSKOD project which was submitted to 2006 SETIN call.

    The CrypSKOD project, in which was involved the OPTICS Department of the FEMTO-ST Institute and the R2D2 team from the IRISA, was a research project intended to propose robust software and hardware cryptographic systems based on chaos.

- ISITE-BFC / Industry Calls for projects

  - Member of the ADVANCES project which was successfully submitted to the Second ISITE-BFC call for proposals in november 2017.

    The Automatic Detection of Viable myocArdiac segmeNts Considering dEep networkS (ADVANCES) project, which will have a three-year duration, has been awarded 309,000 euros in february 2018.

Organization of conference

Member of the organizing committee for the 9^{th} International Meeting Statistical Implicative Analysis (ASI9), which took place at the University Institute of Technology of Belfort-Montbéliard (IUT Belfort-Montbéliard) from october 4 to 7 in 2017.

Reviewer activity

My peer review contributions for academic journals can be tracked and verified on Publons website.

- Reviewer for international journals

  - Algorithms in 2018
  - Symmetry in 2018 (SJR Q2 in Computer Science)
  - IEEE Journal of Microelectromechanical Systems (IF Web of Science JCR 2016/2017 2.124) in 2018
  - Mathematics in 2018
  - IEEE/ACM Transactions on Computational Biology and Bioinformatics (IF Web of Science JCR 2016/2017 1.955) in 2018
  - Journal of Computational Science (IF Web of Science JCR 2016/17 1.748) in 2015, 2016, 2017, and 2018
  - Journal of Supercomputing (IF Web of Science JCR 2016/2017 1.326) in 2016
  - Engineering Applications of Artificial Intelligence (IF Web of Science JCR 2015 2.368) in 2014
  - Special issue of IEEE Computational Intelligence Magazine in 2012

    (Topic: Computational Intelligence in Computer Vision and Image Processing)

- Reviewer for international conferences

  - MENACOMM 2018, IEEE Middle East North Africa COMMunications Conference, Jounieh, Lebanon

- MMSys 2018 (IoT and Smart Cities track), ACM Multimedia Systems Conference, Amsterdam, Netherlands
- ISPA 2017, 15th IEEE International Symposium on Parallel and Distributed Processing with Applications, Guangzhou, China
- CSE 2016, 19th IEEE Int. Conf. on Computational Science and Engineering, Paris, France
- ICANN 2009, 19th International Conference on Artificial Neural Networks, Limassol, Cyprus
- LCN 2007, 32nd IEEE Int. Conf. on Local Computer Networks, Dublin, Ireland

Courses taught

Since september 2002, I have mainly taught courses in Computer Architecture, Operating systems, and Networks at University Institute of Technology of Belfort-Montbéliard (IUT Belfort-Montbéliard) to students studying for a two-year university degree in technology (DUT) in Computer Science. During the years 2011 through 2016, my teaching load was in average of 280 hours. In addition to these courses, I also give some lectures at Bachelor's level and I supervise interns, in particular at the Master's level.

List of publications

In the following list of publications, apart from publications [19] and [2], only the ones issued from research works done since I became an assistant professor are listed. A publication with a star $\star$ in superscript at the end of the authors' list means that they are given in alphabetical order. The journal impact factor given for publications in 2016 and 2017 are provided by the 2016/2017 JCR release.

Peer reviewed international journals

[1] Ali Kadhum Idrees, Karine Deschinkel, Michel Salomon, and Raphaël Couturier. "Multiround Distributed Lifetime Coverage Optimization Protocol in Wireless Sensor Networks". In: *The Journal of Supercomputing* 74.5 (2018), pp. 1949–1972. ISSN: 1573-0484. DOI: 10.1007/s11227-017-2203-7. IF Web of Science JCR 1.326, SJR Q2 in Computer Science.

[2] Régis Garnier et al. "Comparison of Metaheuristics to Measure Gene Effects on Phylogenetic Supports and Topologies". In: *BMC Bioinformatics* x (2017), x–y. IF Web of Science JCR 2.448, SJR Q1 in Computer Science Applications.

[3] Michel Salomon, Raphaël Couturier, Christophe Guyeux, Jean-François Couchot, and Jacques M. Bahi. "Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine". In: *European Research in Telemedicine / La Recherche Européenne en Télémédecine* 6.2 (2017), pp. 79–92. ISSN: 2212-764X. DOI: 10.1016/j.eurtel.2017.06.001. SJR Q3 in Health Informatics.

[4] Ali Kadhum Idrees, Karine Deschinkel, Michel Salomon, and Raphaël Couturier. "Perimeter-based coverage optimization to improve lifetime in wireless sensor networks". In: *Engineering Optimization* 48.11 (2016), pp. 1951–1972. DOI: 10.1080/0305215X.2016.1145015. IF Web of Science JCR 1.728, SJR Q2 in Computer Science.

[5] Pierre-Emmanuel Leni et al. "Development of a 4D numerical chest phantom with customizable breathing". In: *Physica Medica: European Journal of Medical Physics* 32.6 (2016), pp. 795–800. DOI: doi:10.1016/j.ejmp.2016.05.004. IF Web of Science JCR 1.99, SJR Q2 in Radiology, Nuclear Medicine and Imaging.

[6] Ali Kadhum Idrees, Karine Deschinkel, Michel Salomon, and Raphaël Couturier. "Distributed lifetime coverage optimization protocol in wireless sensor networks". In: *The Journal of Supercomputing* 71.12 (2015), pp. 4578–4593. DOI: 10.1007/s11227-015-1558-x. IF Web of Science JCR 1.088, SJR Q2 in Computer Science.

[7] Julien Henriet, Pierre-Emmanuel Leni, Rémy Laurent, and Michel Salomon. "Case-Based Reasoning adaptation of numerical representations of human organs by interpolation". In: *Expert Systems with Applications* 41.2 (2014), pp. 260–266. DOI: 10.1016/j.eswa.2013.05.064. IF Web of Science JCR 2.981, SJR Q1 in Computer Science.

[8] Jean-François Couchot, Karine Deschinkel, and Michel Salomon*. "Active MEMS-based flow control using artificial neural network". In: *Mechatronics* 23.7 (2013), pp. 898–905. DOI: 10.1016/j.mechatronics.2013.02.010. IF Web of Science JCR 1.871, SJR Q1 in Computer Science.

[9] Julien Henriet et al. "EQUIVOX: An example of adaptation using an artificial neural network on a case-based reasoning platform". In: *Biomedical Engineering: Applications, Basis and Communications* 25.02 (2013), p. 1350027. DOI: 10.4015/S1016237213500270. IF Scopus 0.233, SJR Q4 in Bioengineering.

[10] Rémy Laurent, Pierre-Emmanuel Leni, Michel Salomon, Julien Henriet, and Régine Gschwind. "Integration of the lung motion into 3D phantoms". In: *Physica Medica: European Journal of Medical Physiscs* 29, Supplement 1 (2013), e25–. DOI: 10.1016/j.ejmp.2013.08.081. IF Web of Science JCR 1.763, SJR Q2 in Radiology, Nuclear Medicine and Imaging.

[11] Jacques M. Bahi, Nathalie Coté, Christophe Guyeux, and Michel Salomon*. "Protein folding in the 2D Hydrophobic–Hydrophilic (HP) square lattice model is chaotic". In: *Cognitive Computation* 4 (1 2012), pp. 98–114. DOI: 10.1007/s12559-011-9118-z. IF Web of Science JCR 1.933, SJR Q2 in Computer Science Applications.

[12] Jacques M. Bahi, Jean-François Couchot, Christophe Guyeux, and Michel Salomon*. "Neural networks and chaos: Construction, evaluation of chaotic networks, and prediction of chaos with multilayer feedforward networks". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1 (2012), p. 013122. DOI: 10.1063/1.3685524. IF Web of Science JCR 2.049, SJR Q2 in Applied Mathematics.

[13] Rémy Laurent, Régine Gschwind, Michel Salomon, Julien Henriet, and Libor Makovicka. "Perspective de la plate-forme NEMOSIS dans le cadre d´une réduction de doses en imagerie". In: *Radioprotection* 47.4 (2012), pp. 599–617. DOI: 10.1051/radiopro/2012030. IF Web of Science JCR 0.508, SJR Q3 in Nuclear Energy and Engineering.

[14] Rémy Laurent et al. “Data Processing using Artificial Neural Networks to Improve the Simulation of Lung Motion”. In: *Biomedical Engineering: Applications, Basis and Communications (BME)* 24.06 (2012), pp. 563–571. DOI: 10.4015/S1016237212500524. IF Scopus 0.233, SJR Q4 in Bioengineering.

[15] Rémy Laurent et al. “Respiratory lung motion using an artificial neural network”. In: *Neural Computing and Applications* 21.5 (2012), pp. 929–934. DOI: 10.1007/s00521-011-0727-y. IF Web of Science JCR 1.492, SJR Q2 in Artificial Intelligence.

[16] Jad Farah et al. “Development of a new CBR-based platform for human contamination emergency situations”. In: 144.1-4 (2011), pp. 564–570. DOI: 10.1093/rpd/ncq440. IF Web of Science JCR 0.894, SJR Q3 in Radiology, Nuclear Medicine and Imaging.

[17] Jacques M. Bahi and Michel Salomon*. “A decentralized energy-based diffusion algorithm to increase the lifetime of MANETs”. In: *Computer Networks* 54.16 (2010), pp. 2887–2898. DOI: 10.1016/j.comnet.2010.07.021. IF Web of Science JCR 1.446, SJR Q2 in Computer Networks and Communication.

[18] Jacques M. Bahi, Raphaël Couturier, Kamel Mazouzi, and Michel Salomon*. “Synchronous and asynchronous solution of a 3D transport model in a grid computing environment”. In: *Applied Mathematical Modelling* 30.7 (2006), pp. 616–628. DOI: 10.1016/j.apm.2005.06.017. IF Web of Science JCR 2.291, SJR Q1 in Applied Mathematics and Modeling and Simulation.

[19] Michel Salomon, Fabrice Heitz, Guy-René Perrin, and Jean-Paul Armspach. “A massively parallel approach to deformable matching of 3D medical images via stochastic differential equations”. In: *Parallel Computing* 31.1 (2005), pp. 45–71. DOI: 10.1016/j.parco.2004.12.003. IF Web of Science JCR 1.000, SJR Q2.

Peer reviewed national journals

[1] Rémy Laurent et al. “Utilisation d’un réseau de neurones artificiels pour la simulation des mouvements pulmonaires”. In: *Cancer/Radiothérapie* 15.2 (2011), pp. 123–129. DOI: 10.1016/j.canrad.2010.07.636.

[2] Libor Makovicka et al. “Avenir des nouveaux concepts des calculs dosimétriques basés sur les méthodes de Monte Carlo”. In: *Radioprotection* 44.1 (2009), pp. 77–88. DOI: 10.1051/radiopro/2008055.

Book chapters

[1] Marc Sauget, Sylvain Contassot-Vivier, and Michel Salomon. “Parallelization of neural network building and training: an original decomposition method”. In: *Advances in Mathematics Research*. Ed. by Albert R. Baswell. Vol. 17. Nova Publishers, 2012, pp. 193–223. URL: http://hal.inria.fr/hal-00643920/en.

[2] Michel Salomon, Guy-René Perrin, Fabrice Heitz, and Jean-Paul Armspach. “Parallel differential evolution: application to 3-D medical image registration”. In: *Differential Evolution*. Springer, 2005, pp. 353–411.

PEER REVIEWED INTERNATIONAL CONFERENCES

[1] Jean-François Couchot, Raphaël Couturier, and Michel Salomon*. “Improving Blind Steganalysis in Spatial Domain using a Criterion to Choose the Appropriate Steganalyzer between CNN and SRM+EC”. In: *ICT Systems Security and Privacy Protection: 32nd IFIP TC 11 International Conference, SEC 2017, Rome, Italy, May 29-31, 2017, Proceedings, IFIP Advances in Information and Communication Technology*. Ed. by Sabrina De Capitani di Vimercati and Fabio Martinelli. Vol. 502. Springer International Publishing, 2017, pp. 327–340. DOI: 10.1007/978-3-319-58469-0. Rank B.

[2] Christophe Guyeux, Bashar Al-Nuaimi, Bassam AlKindy, Jean-François Couchot, and Michel Salomon. “On the ability to reconstruct ancestral genomes from mycobacterium genus”. In: *Bioinformatics and Biomedical Engineering: 5th International Work-Conference, IWBBIO 2017, Granada, Spain, April 26–28, 2017, Proceedings, LNCS Part I*. Ed. by Ignacio Rojas and Francisco Ortuño. Vol. 10208. Cham: Springer International Publishing, 2017, pp. 642–658. DOI: 10.1007/978-3-319-56148-6_57.

[3] Bashar Al-Nuaimi, Christophe Guyeux, Bassam Alkindy, Michel Salomon, and Jean-François Couchot. “Relation between Gene Content and Taxonomy in Chloroplasts”. In: *International Journal of Bioscience, Biochemistry and Bioinformatics* 7 (1 2017). Selected papers of ICBSB 2016, pp. 41–50. DOI: 10.17706/ijbbb.2017.7.1.41-50.

[4] Bassam Alkindy et al. “Binary particle swarm optimization versus hybrid genetic algorithm for inferring well supported phylogenetic trees”. In: *Computational Intelligence Methods for Bioinformatics and Biostatistics: 12th International Meeting, CIBB 2015, Naples, Italy, September 10-12, 2015, Proceedings Revised Selected Papers*. Ed. by Claudia Angelini, Paola MV Rancoita, and Stefano Rovetta. Cham: Springer International Publishing, 2016, pp. 165–179. DOI: 10.1007/978-3-319-44332-4_13.

[5] Nils Schaetti, Michel Salomon, and Raphaël Couturier. “Echo state networks-based reservoir computing for MNIST handwritten digits recognition”. In: *19th IEEE International Conference on Computational Science and Engineering, CSE 2016, Paris, 24-26 August 2016, Proceedings*. 2016, pp. 484–491. DOI: 10.1109/CSE-EUC-DCABES.2016.229.

[6] Bassam AlKindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M. Bahi. “Using genetic algorithm for optimizing phylogenetic tree inference in plant species”. In: *2015 Mathematical and Computational Evolutionary Biology (MCEB), Proceedings*. June 2015.

[7] Bassam AlKindy et al. “Hybrid genetic algorithm and lasso test approach for inferring well supported phylogenetic trees based on subsets of chloroplastic core genes”. In: *Algorithms for Computational Biology: Second International Conference, AlCoB 2015, Mexico City, Mexico, August 4-5, 2015, Proceedings, LNCS/LNBI*. Ed. by Adrian-Horia Dediu, Francisco Hernández-Quiroz, Carlos Martín-Vide, and David A. Rosenblueth. Vol. 9199. Springer International Publishing, 2015, pp. 83–96. DOI: 10.1007/978-3-319-21233-3_7.

[8] Bassam AlKindy et al. “Improved Core Genes Prediction for Constructing Well-Supported Phylogenetic Trees in Large Sets of Plant Species”. In: *Bioinformatics and Biomedical Engineering: Third International Work Conference, IWBBIO 2015, Granada, Spain, April 15-17, 2015, Proceedings, Part I, LNCS*. Ed. by Francisco Ortuño and Ignacio Rojas. Vol. 9043. Cham: Springer International Publishing, 2015, pp. 379–390. DOI: 10.1007/978-3-319-16483-0_38.

[9] Bassam AlKindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M. Bahi. “Gene similarity-based approaches for determining core-genes of chloroplasts”. In: *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Proceedings*. Nov. 2014, pp. 71–74. DOI: 10.1109/BIBM.2014.6999130.

[10] Bassam Alkindy et al. “Finding the core-genes of chloroplasts”. In: *Bioinformatics and Biomedical Science: Third International Conference, ICBBS 2014, Copenhagen, Denmark, June 2014, International Journal of Bioscience, Biochemistry and Bioinformatics*. Vol. 4. 2014, pp. 361–368. DOI: 10.7763/IJBBB.2014.V4.371.

[11] Ali Kadhum Idrees, Karine Deschinkel, Michel Salomon, and Raphaël Couturier. “Coverage and lifetime optimization in heterogeneous energy wireless sensor networks”. In: *13th IARIA International Conference on Networks, ICN 2014, Nice, 23-27 February 2014, Proceedings*. 2014, pp. 49–54.

[12] Jean-François Couchot, Karine Deschinkel, and Michel Salomon*. “Suitability of artificial neural network for MEMS-based flow control”. In: *2012 Second Workshop on Design, Control and Software Implementation for Distributed MEMS, Proceedings*. Besançon, Apr. 2012, pp. 1–6. DOI: 10.1109/dMEMS.2012.17.

[13] Julien Henriet et al. “Adapting numerical representations of lung contours using Case-Based Reasoning and artificial neural networks”. In: *20th International Conference on Case-Based Reasoning, Proceedings, LNCS*. Ed. by B. Díaz Agudo and I. Watson. Vol. 7466. Springer, Heidelberg, 2012, pp. 137–151. URL: http://hal.inria.fr/hal-00714584.

[14] Jacques M. Bahi, Christophe Guyeux, and Michel Salomon*. “Building a Chaotic Proved Neural Network”. In: vol. abs/1101.4351. 2011. URL: http://arxiv.org/abs/1101.4351.

[15] Marc Sauget, Julien Henriet, Michel Salomon, and Sylvain Contassot-Vivier. “Large datasets: A mixed method to adapt and improve their learning by neural networks used in regression contexts”. In: *EANN/AIAI (1)*. Ed. by Lazaros S. Iliadis and Chrisina Jayne. Vol. 363. IFIP Advances in Information and Communication Technology. Springer, 2011, pp. 182–191. DOI: 10.1007/978-3-642-23957-1_21. Rank C.

[16] Marc Sauget et al. “Efficient domain decomposition for a neural network learning algorithm, used for the dose evaluation in external radiotherapy”. In: *ICANN (1)*. Ed. by Konstantinos I. Diamantaras, Wlodek Duch, and Lazaros S. Iliadis. Vol. 6352. Lecture Notes in Computer Science. Springer, 2010, pp. 261–266. DOI: 10.1007/978-3-642-15819-3_34. Rank B.

[17] Jacques M. Bahi, Ahmed Mostefaoui, and Michel Salomon*. “A local-control algorithm to prolong the lifetime of wireless ad hoc networks”. In: *Mobile Ad-hoc and Sensor Networks, Second International Conference, MSN 2006, Hong Kong, China, December 13-15, 2006, Proceedings*. Ed. by Jiannong Cao, Ivan Stojmen-

ovic, Xiaohua Jia, and Sajal K. Das. Vol. 4325. Lecture Notes in Computer Science. Springer, 2006, pp. 555–566. DOI: 10.1007/11943952_47.

[18] Jacques M. Bahi, Ahmed Mostefaoui, and Michel Salomon*. “Increasing lifetime of wireless ad hoc networks using a decentralized algorithmic approach”. In: *14th IEEE International Conference on Networks, ICON 2006, Singapore, 13-15 September 2006*. IEEE, 2006, pp. 1–6. ISBN: 0-7803-9746-0. DOI: 10.1109/ICON.2006.302668. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4087640. Rank B.

[19] Amine Abbas, Jacques M. Bahi, Sylvain Contassot-Vivier, and Michel Salomon*. “Mixing synchronism/asynchronism in discrete-state discrete-time dynamic networks”. In: *DYNAMICS OF CONTINUOUS DISCRETE AND IMPULSIVE SYSTEMS-SERIES B-APPLICATIONS & ALGORITHMS* 2 (2005). 4th Internationl Conference on Engineering Applications and Computational Algorithms, Guelph, Canada, July 2005, pp. 524–529.

[20] Jacques M. Bahi, Raphaël Couturier, and Michel Salomon*. “Solving three-dimensional transport models with synchronous and asynchronous iterative algorithms in a grid computing environment”. In: *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA*. IEEE Computer Society, 2005. ISBN: 0-7695-2312-9. DOI: 10.1109/IPDPS.2005.405. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=9722.

SOFTWARE PATENTS

[1] Jean-François Bosset, Julien Henriet, Rémy Laurent, Libor Makovicka, and Michel Salomon. *NEMOSIS V1.0 du 21/09/11*. Produit logiciel. Numéro de dépôt APP : IDDN.FR.001.170023.000.S.P.2012.000.31230 (logiciel oeuvre de l’Université de Franche-Comté). 2012.

[2] Julien Henriet et al. *EquiVox, projet T2IRM, Techniques Informatique Innovantes en Radiophysique Médicale*. Produit logiciel. Numéro de dépôt SPV-CNRS : 4047-01 (logiciel oeuvre de l’Université de Franche-Comté). 2011.

OTHERS

[1] Michel Salomon. “Introduction à l’intelligence artificielle”. In: *Journée CMI R&D Day, 21 Mars 2018, Besançon, France*. 2018. Oral presentation.

[2] Jean-François Couchot, Raphaël Couturier, and Michel Salomon*. “Deux (stéganalyseurs) valent mieux qu’un (stéganalyseur)”. In: *CNRIUT 2017, Congrès National de la Recherche des IUT, 4-5 Mai 2017, Auxerre, France*. 2017. Oral presentation.

[3] Raphaël Couturier and Michel Salomon*. “Deep Learning pour la stéganalyse (mésocentre de Franche Comté)”. In: *Journées scientifiques Equi@Meso, 30-31 Janvier 2017, Grenoble, France*. 2017. Oral presentation.

[4] Jean-François Couchot, Raphaël Couturier, Christophe Guyeux, and Michel Salomon*. “Steganalysis via a Convolutional Neural Network using Large Convolution Filters for Embedding Process with Same Stego Key”. In: *ArXiv e-prints* (May 2016). arXiv: 1605.07946 `[cs.MM]`.

[5] Raphaël Couturier and Michel Salomon*. “Parallelization and optimization of the neuromorphic simulation code. Application on the MNIST problem”. In: *Workshop dynamical Systems and Brain-inspired Information Processing, 2-3 Novembre 2015, Besançon, France*. 2015. Oral presentation.

Contents

General Introduction

1/ Motivations

Many innovations are the results of ideas taken from natural processes. For example, in the context of combinatorial optimization, which we studied during our PhD thesis, several stochastic algorithms inspired from physical or biological phenomenons, like particle swarm optimization and simulated annealing, have been designed over the past decades. Another field of computer science in which similar ideas have been applied with a large success is machine learning, leading to new developments in artificial intelligence. The fascinating information processing capabilities of the brain, an extremely complex and dense network that interconnects billions of small processing units called neurons, has particularly motivated scientists to develop mathematical models in order to mimic its functioning. These models have given rise to the branch of machine learning called artificial neural networks due to their analogy with biological neural networks.

Since the introduction of the perceptron at the end of the 1950s, neural networks have passed through several boom-and-bust episodes. The backpropagation algorithm popularized by Rumelhart and al. has thus been the starting point of the multilayer networks era. Earlier versions of these networks were shallow, typically composed of one hidden layer. It is well-known that they are universal approximation tools for nonlinear systems. More precisely, it has been proved that the commonly used types of multilayer feedforward neural networks with a single hidden layer, also known as MultiLayer Perceptron networks, and Radial Basis Function networks can approximate any nonlinear function to any arbitrary precision. On the one hand MLPs construct global approximations, on the other hand RBFs have local approximation capabilities. Each family of networks has some strengths and weaknesses, for example MLP networks are usually smaller whereas RBF ones will be trained much faster.

After a period without any real progress, more biologically plausible and complex models have then been developed. Indeed, thanks to the increasing computing power available at a lower cost, spiking and deep networks have emerged. These latter have led to breakthrough results in many problems with such impressive performances that they have produced a revival of interest in neural networks, becoming the new hip branch of machine learning known as deep learning. As a result, today neural networks are used to solve complex real world problems in a wide range of areas such as speech recognition, prognostics and health management, medical image analysis, and so on. For that matter, neural networks, and more especially deep learning, have achieved the most stunning achievements in artificial intelligence over the past few years. An iconic example is AlphaGo, the first program that managed to beat one of the best professional Go player.

Motivated by the desire to take advantage from the capabilities of neural networks to model complex relationships between inputs and outputs, we have investigated during the past decade their applications to many different problem domains. The research works described in this manuscript reflect the network architecture evolution, starting with works where the multilayer perceptron is considered to ones that study deep networks, and highlights the suitability of neural networks for successfully solving tasks in different areas. When designing a neural network for a given problem several questions have to be addressed, among which the main one is the choice of the architecture. Even if the kind of problem to be solved and the data are guidelines, usually allowing to choose between a feedforward or a recurrent network, between a supervised or an unsupervised training, and to define the inputs and outputs, few indications are available for the definition of the inner hidden part.

2/ Contributions and outline

This manuscript presents different research works in which neural networks are used to solve prediction or classification problems. The first part, which consists of Chapters 1 to 3, deals with problems in which the multilayer perceptron architecture is used as a function approximator. Thus, we show that a multilayer perceptron neural network can simulate lung motion and is able to predict the trajectory of several anatomical feature points. It can also be somewhat a substitute of a computational fluid dynamics software for active airflow control, and allows to build a true chaotic neural network.

The second part explores deep learning architectures in Chapters 4 and 5, while in the final chapter the Reservoir Computing concept is studied. First of all, convolutional neural networks (a main category of deep learning) are investigated to detect whether an image embeds a hidden message or not. The obtained steganalysis approach was at the time of its publication the state-of-the-art one. Then we propose a fully convolutional neural network for image denoising that is an encoder-decoder. This one can deal with different kinds of noise, but currently it is trained specifically for a targeted noise. Finally, considering the reservoir computing paradigm studied by colleagues of our research institute, the application of an Echo State Network, a recurrent architecture, on a handwritten digits recognition problem (namely the MNIST benchmark problem) is studied.

The structure of the manuscript follows the contributions presented in the above paragraphs and is outlined below.

- Chapter 1 takes place in the context of lung cancer treatment with external-beam radiation therapy. It presents an innovative method for the management of respiratory motion, since breathing motion is a crucial problem during radiation treatment of many tumors. The proposed approach allows to simulate the position of anatomical feature points and thus to provide a customized simulation of lung motions. Different neural networks are designed and trained, first to preprocess the input data and then to produce the movements of normal tissues.

- Chapter 2 describes the design of a multilayer network to approximate the force induced by an airflow over a backward facing step. The idea is to have fast and accurate predictions that allow to perform active control in order to maximize the force applied on the back of the step. This work is a proof of concept in the perspective of performing active control of turbulent flows for drag reduction and thus to improve the aerodynamic performance of moving objects.

- Chapter 3 investigates the use of neural networks when dealing with chaos. More precisely, we define a method to build chaotic neural networks using a chaotic iterations framework that enables to prove a chaotic behaviour in a rigorous mathematical sense. The dual case of checking whether a neural network is chaotic or not is also investigated. Finally, the ability of traditional multilayer perceptron networks to predict a chaotic dynamics, namely the folding process in 2D model, is discussed.
- In Chapter 4, following the trend of using deep learning for classification problems in computer vision, we first try to design a Convolutional Neural Network (CNN) architecture to solve the problem of detecting the existence of a secret message in an image. Unfortunately, the network is only competitive when the embedding of the message is done with the same key whatever the image. Thanks to a careful study of the images for which CNN-steganalysis fails, considering the CNN architecture giving the best detection rates at the time of this work, we are then able to propose a criteria to choose between the CNN and the classical steganalysis approach for a given input image. This proposal offers state-of-the-art detection performances and can perform blind steganaysis.
- Chapter 5 is devoted to the description of our most recent works, namely the design of a fully CNN architecture for image denoising. We present the great lines of the network architecture that consists in an encoder-decoder with skip connections and the loss function chosen to train the network is examined. The first experiments show that it can achieve promising results and treat different types of noise, namely additive white Gaussian noise and speckle noise.
- Chapter 6 is inspired by the performance of a neuromorphic computer developed by colleagues of the OPTICS department from the FEMTO-ST Institute on a speech recognition task. Its ability to classify spoken digits at a throughput of one million words per second, with a very low word error rate, has convinced us to explore the underlying concept: the Reservoir Computing (RC) paradigm, on which their physical implementation is based. Therefore, in this chapter we study the Echo State Network (ESN) architecture, the machine learning representative of RC, and we apply it to the MNIST handwritten digit recognition problem for which, at a first glance, it seems not to be designed. For this image classification problem, the lowest error provided by an ESN is far away from the best ones given by state-of-the art CNNs, but slightly lower than those of the LeNet convolutional networks.

Finally, conclusions are drawn and future directions for research are given.

I

Solving Problems with Multilayer Perceptrons

1
Simulation of Lung Motion

Among the different cancers of the thorax and the abdomen, lung cancer is the most prevalent and relevant disease for external-beam radiation therapy. Moreover, lung cancer is responsible for nearly one in five cancer deaths worldwide, being the leading cause of cancer deaths in men in many countries (such as in France) and in the top three ones in women. In fact, according to the World Heath Organization, cancer was the cause of death for 8.8 millions persons in 2015 and lung cancer was the main type of fatal cancer (1.69 millions deaths), followed by liver cancer (788,000 deaths) and colorectal cancer (774,000 deaths). Cancer mortality statistics for France, provided by the Institut National du Cancer (INCA), also clearly show that while lung cancer stagnates in men, it progresses strongly in women with a mortality rate getting closer to that of the breast cancer. Therefore, a better understanding of the individual lung motion of each patient would allow to follow the evolution of the clinical target volume according to the breathing phase, and thus to optimize the radiation therapy dose delivery. There is also a greater need for precision in lung motion monitoring due to the development over the past decade of stereotactic radiation therapy [1]. Indeed, by comparison with conventional radiation therapy, the stereotactic one employs multiple beams and thus delivers higher doses of radiation (up to 20 Gy during a session *versus* 2 Gy). On the one hand there is clinical evidence of survival advantage for higher dose levels which can be delivered with a better tumor targeting and on the other hand avoiding as much as possible normal tissues is of increasing importance due to the growing use of concomitant chemotherapy. Compared to chemotherapy, which is a treatment that affects the whole body of a patient, radiation therapy is a local treatment which focuses on cancer cells.

A four-dimensional computed tomography (4D-CT) scanner can provide information on the tumor movement, but biases and errors on the localization can be introduced and limiting the use of 4D-CT is interesting for radiation protection. The purpose of this chapter, which is a summary of our collaboration with colleagues of the IRMA team from the Chrono-Environnement laboratory [2, 3, 4], taking place during Remy Laurent's PhD thesis co-supervising, is thus to present an innovative method to simulate the position of points in a person's lungs at each breathing phase. The proposed approach consists of two steps using both MultiLayer Perceptron (MLP) neural networks and is based on data provided by the 4D-CT scanner of several patients. The first step is a data augmentation technique allowing to build a training set, which is then used in the second step to train a neural network that learns the lung motion. After training, this latter network can produce a customized simulation for breathing and thus simulates the respiration of an unknown patient. The resulting NEural NEtwork MOtion SImulation System (NEMOSIS) software platform has been protected by means of a patent [5].

The chapter is organized as follows. First, in Section 1.1 the limitations of the 4D-CT scanner and alternative approaches to simulate lung motion are discussed. In the following section, the data set and our proposal, more particularly the neural networks used to improve the data set and learn the lung movement are described. The obtained results are then presented in Section 1.3 and discussed in Section 1.4.

1.1/ Introduction

Radiation therapy, also called radiotherapy, is one of the most common treatments for cancer. It uses high doses of radiation to kill cancer cells and shrink tumors. The more the radiation beams concentrate on the targeted tumor, the better they can break of the DNA inside the cancer cells, and thus the more likely they are to prevent the growth and the division of the cells, leading ultimately to damages that cause them to die. In the case of lung tumors, it is crucial to be able to track them precisely, otherwise the respiratory motion will be a major problem in the treatment, reducing its efficiency. 4D-CT is a medical imaging modality used in routine clinical practice that enables temporal synchronization of respiratory acquisition, but at the cost of increasing patients' exposure to radiation (around 35 times the annual natural exposure). Even if the proposed method is based on data provided by the 4D-CT, our objective is not to recover the locations given by the 4D-CT, but rather to give a smooth realistic and regular motion simulation.

Let us start by giving some details on the 4D-CT acquisition process. As can be seen in Figure 1.1, it consists in the acquisition of 3D-CT data throughout the respiratory cycle. For each table position, the respiratory cycle is represented by 10 uniformly distributed “phases”, where each phase corresponds to an instant in the cycle. The phase values range from 0% to 100%, with 0% and 50% which represent, respectively, the maximum exhalation and inspiration. Furthermore, the two extreme phase values, 0% and 100%, point to the same phase due the periodicity of the respiratory cycle. The final 4D-CT scan is thus composed of 10 3D-CT images representing the different phases. Therefore, the acquired 3D-CT data are combined based on a respiratory-correlated signal which is acquired simultaneously (RPM signal) so that images belonging to the same phase are concatenated to form a single representative 3D-CT image. The figure shows how the image for the 50% phase is built using images from several respiratory periods.

Patients' ability to hold their breath reproducibly greatly impacts on the temporal distribution of the phases. In the figure, although phases 51%, 59%, and 44% are close to the target phase of 50%, the corresponding RPM signals are very different, with a gap of more than 1.25 cm (the average respiratory amplitude is 1.5 cm). These differences explain the resulting kinetic artifacts which can be easily detected by the discontinuity in the anatomical structures being imaged. Another kind of artifacts, characterized by a poor definition of anatomical structures, is the blur associated with slow rotation time of the x-ray tube. It has been shown [6] that both artifacts lead to contouring error of the Gross Tumor Volume (GTV), with size differences up to 90%. Thus, even if 4D-CT imaging is commonly used, it leads to images with artifacts because of its inability to adapt to respiratory variations, in period and magnitude, and slow acquisition process unsuitable to capture tumor motion.

Different research works have investigated the simulation of breathing motion. However, they usually lack either in time efficiency or in precision. Non-linear registration methods

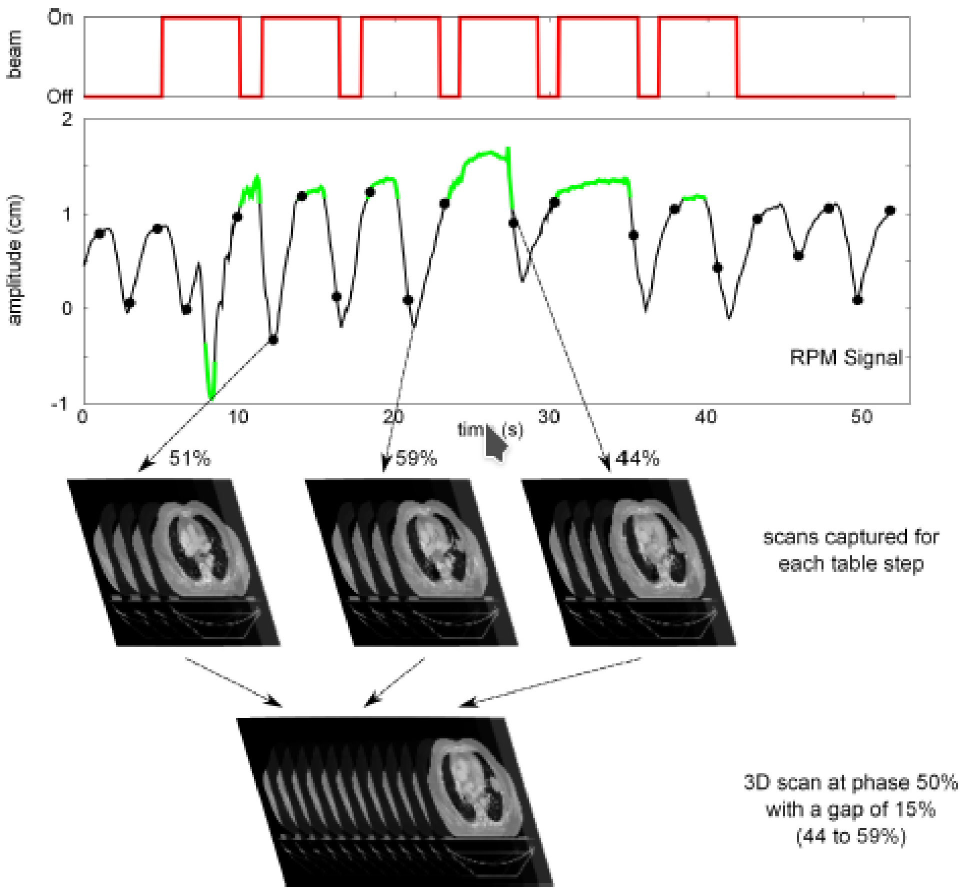

On
Off
beam
amplitude (cm)
RPM Signal
time (s)
51%
59%
44%
scans captured for each table step
3D scan at phase 50% with a gap of 15% (44 to 59%)

Figure 1.1: 4D-CT imaging: reconstruction process of a 3D-CT image at a discrete phase (50% in this example) for a patient according to its respiratory signal.

applied to images issued from different breathing phases allow to estimate respiratory motion fields [7, 8], but underlying physiological and anatomical processes of breathing are mostly ignored. To take into account these processes and thus have a more realistic motion, various biomechanical model based simulation methods have been proposed. Sarrut *et al* [9, 10] have designed an algorithm based on the continuum mechanics theory, considering the impact of the diaphragm and ribs on lung deformations. They obtained subvoxel accuracy, but their proposal cannot be run in real-time. The approach of Boldea *et al.* [11], which simulates a 4D-CT using two extreme 3D-CT scanner images and vector interpolation to build intermediate images, gives similar performance. Methods based on marks at different positions on the patient's skin and a scanner image [12] can face the real-time constraint, but they offer low-precision. An important point that should be taken into account when designing a respiratory model is the intrinsic hysteresis behavior of the breathing motion. In fact, a respiratory model able to reproduce the hysteretic behavior of lungs would enable the radiation therapy to be more accurate, since lung tumors are also subject to hysteresis [13].

To overcome the limitations and drawbacks of 4D computed tomography, we have first proposed a method using a multilayer perceptron neural network for a real-time simulation of lung motion [2]. Artificial Neural Networks (ANN) are widely used in medicine, for example to help diagnosis of breast [14] and lung [15] cancer, in dosimetric computations [16], to predict a tumor motion according to the position of external markers [17], and so on. Murphy *et al.* [18] have also trained a MLP with one hidden layer in order to simulate the breathing signal of a patient in real time. Then, in order to improve the quality and the accuracy of the motion information, we have investigated the use of additional neural networks to process the 4D-CT data used to learn the movement [4]. Such a preprocessing of data is relevant [19] in our case for two reasons. First, it allows to effectively increase the size of the data set and second it reduces the influence of artifacts and bias introduced by 4D-CT. More precisely, the initial small and noisy data set is used to train neural networks, from which a larger set is then built. In this way, we obtain smoothed data that allow to greatly improve the learning of lung motion by another neural network. Recently, Park *et al.* [20] have proposed an architecture that is a combination of fuzzy logic and a neural network with four hidden layers (explaining why they call their proposal fuzzy deep learning) to predict intra- and inter-fractional variations. Intra-fractional motion indicates changes observed when the patient receives a radiation therapy session, while inter-fractional motion covers variations arising between different sessions (such as tumor weight gain or loss). This predictor, which was trained on breathing data of 130 patients collected using the CyberKnife Synchrony treatment facility, produces an accurate estimate of the next breathing signal before the incoming signal.

1.2/ Materials and method

1.2.1/ Data set construction

The neural network designed to simulate the lung motion of a given patient is trained through a holdout validation, using the motion of anatomical feature points during the breathing cycle. Hence, the data set consists of trajectories of feature points across the 10 discrete and regular phases of the respiratory cycle in 4D-CT images from 10 patients. Among them, five patients (patients $0-4$) compose the training set, while the remaining ones (patients $5-9$) are used for testing. The different feature points were localized by a radiation therapist in non-pathogenic areas and thus our proposal only considers normal tissue motion. As an illustration, Figure 1.2 presents for patient 1 the plotting of two feature points for phases 10% and 70%.

Obviously, the manual plotting of the feature points introduces some biases and noise in the data. Indeed, on the one hand when a point is plotted the corresponding phase is not always displayed by the scanner and, on the other hand, the resolution of the 3D-CT image limits the accuracy of the plotted points localization. The limited accuracy comes from the use of different 4D-CT scanning devices and protocols, producing images with inhomogeneous spatial resolution (voxel size of $1 \times 1 \times 2$ mm^3 and $1 \times 1 \times 2.5$ mm^3). Table 1.1 shows how the 1,518 features points of the data set are distributed among the 10 patients. Let us notice that patients 4, $7-9$ belong to the PoPi-model [21] and that two phases are missing for patient 0.

Table 1.1: Distribution of the feature points among the different patients.

	Patients used for training the MLP used for data preprocessing							Patients used for testing		
Patient ID	0	1	2	3	4	5	6	7	8	9
Lung vol.	4.62	2.86	5.48	2.60	5.94	3.10	4.42	3.72	4.91	4.05
Point/phase	21	25	24	8	38	6	4	10	10	10
Phases	$[0; 20-70; 90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$	$[0-90]$
Total points	168	250	240	80	380	60	40	100	100	100

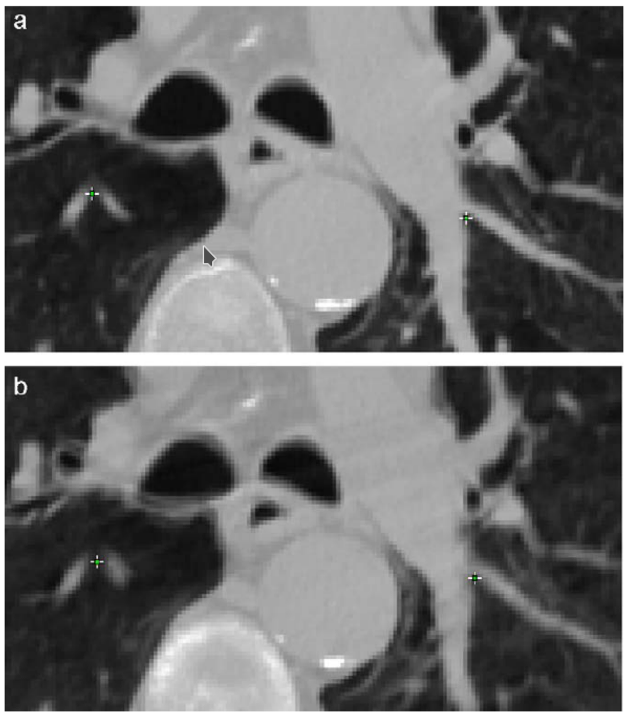


Figure 1.2: Definition of two points on CT images from patient 1 for (a) phase 10% and (b) phase 70%.

The carina, the point where the trachea splits into two bronchi, has been chosen as anatomical landmark because it was easy to identify regardless of the patient. It has been marked for all patients and phases, and its position at phase 0% defines thus the axis origin of the common coordinate system. The coordinates of all points have been normalized according to it. Figure 1.3 gives a view of the spreading of the feature points for patients $0-4$.

1.2.2/ Multilayer perceptron neural networks

In a first step, to reduce potential sources of error and enlarge the data set, the plotted points are fitted using a multilayer perceptron per feature point, a feedforward neural network architecture well-known for its universal approximation property [22]. All these networks have the same topology. They have four inputs: the three coordinates of the considered feature point at phase 0% and the phase number for which a neural network must predict its new position (the output vector is thus three-dimensional). They consist of a single hidden layer of sigmoidal neurons, whose number is determined through an incremental approach [23] according to the motion complexity, and an output layer of neurons with a linear activation function. Once trained, a neural network associated to a feature point can produce a regular and smooth description of its trajectory, which does not depend on the scanner resolution and is less noisy. Such a network can also be used to enlarge the data set with missing information (like phases 10% and 80% for patient 0).

Thanks to this data preprocessing, a corrected version of the initial data set, less noisy and more realistic, can be obtained. The corrected data set, which will be subdivided in learning, validation, and testing sets, is then expected to improve the performance

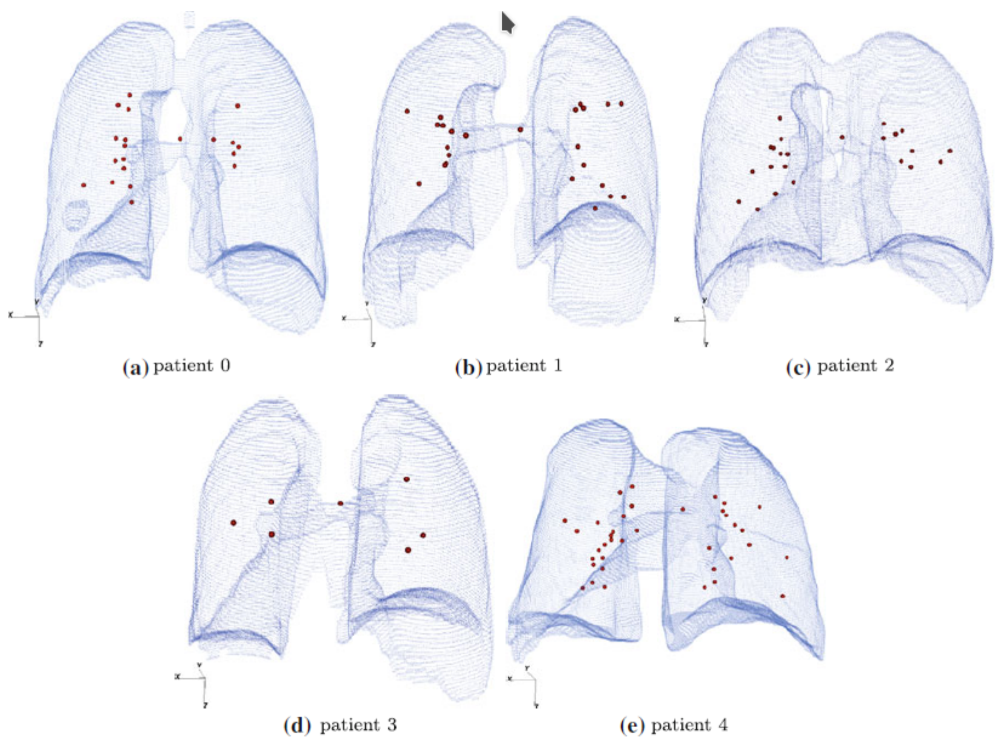


Figure 1.3: Distribution of the points on patients $0 - 4$ from the training set.

of the multilayer perceptron simulating the lung motion in the NEural NEtwork MOtion SImulation System (NEMOSIS) software platform. The topology of this MLP, trained in a second step, is nearly similar to the one of the networks doing the data preprocessing, it has simply twice more inputs for a patient's point:

- its coordinates at phase 0%;
- its coordinates at phase 50%;
- the lung volume to identify the patient;
- the phase number for which the point location must be predicted.

The neural networks are trained to minimize the Mean Squared Error (MSE) using the quasi-Newton Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS), in combination with the Wolfe linear search to determine the optimal step size [24]. This algorithm is appealing because it achieves fast convergence and has a small memory consumption. Five networks are trained to simulate the lung motion using each time the feature points of a different patient as validation set for overfitting control (holdout validation with 5 patients).

Finally, Figure 1.4 gives an overview of our proposal. As can be seen, in the learning process a data preprocessing allows to obtain a Corrected Learning Set (CLS) from the Initial (ILS) one, which is then used to learn the lung motion. Once the learning process is completed, the normal (or clinical routine) execution allows to predict the lung motion of an unknown patient given phases 0% and 50%.

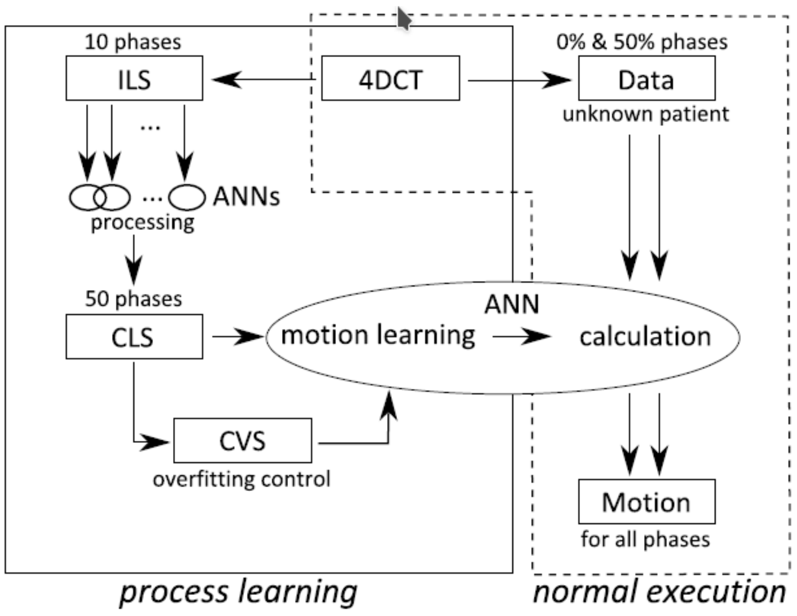


Figure 1.4: Overview of the proposed approach for lung motion simulation (ILS = Initial Learning Set; CLS = Corrected Learning Set; CVS = Corrected Validation Set).

1.3/ Results

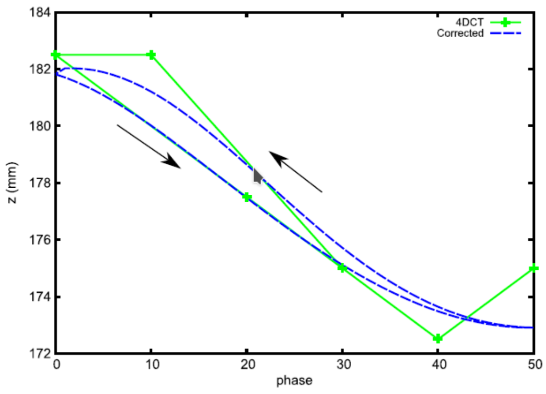
The neural networks issued from the learning process usually have a hidden layer of less than ten neurons. Therefore we will only present in the following part results corresponding to this network configuration. From a computational point of view, on a typical laptop (a dual-core CPU running at 2.1 GHz) the NEMOSIS platform takes about 205 ms to compute the points of patient 4 (the one with the most feature points) over 100 phases, which means that it provides an almost real-time simulation of lung motion.

To obtain a complete breathing cycle with a good continuity between adjacent simulated phases 99% and 0%, the phase domain is extended between -20% and 120%. Since the lung motion is periodic, a correspondence between the phases of this new domain is done as follows:

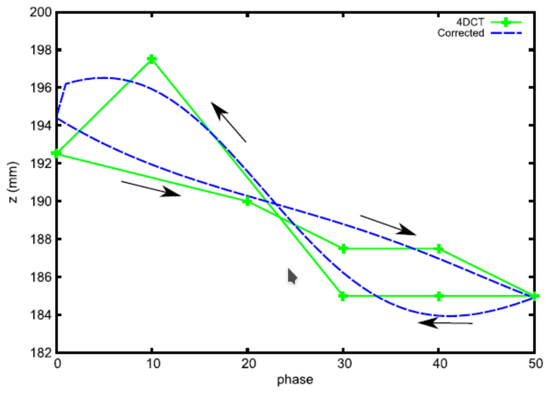
- on the upper part of the domain, phases 0%, 10%, and 20% correspond, respectively, to phases 100%, 110%, and 120%;
- while on the lower part, phases 80% and 90% are, respectively, equal to phases -20% and -10%.

1.3.1/ Data preprocessing

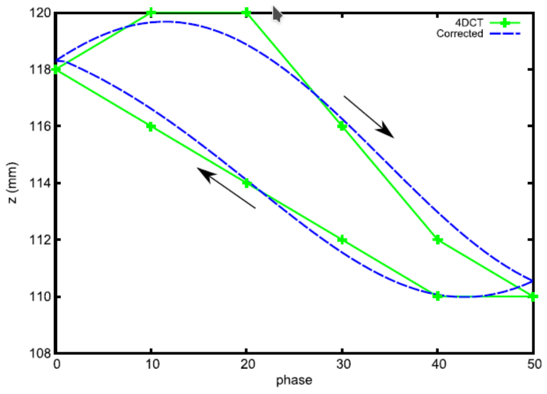
To highlight the interest of the data preprocessing step, Figure 1.5 shows the noisy trajectories resulting from the 4D-CT for points belonging to patient 0 (points 1 and 2) and patient 4 (point 3) and the smoothed ones resulting from the treatment. The trajectories are given considering the z cranio-caudal (from head to foot) axis. Note that in these figures phases 10%, 20%, 30%, and 40% give the point position for them and also, respectively, for phases 90%, 80%, 70%, and 60%.



(a) Trajectories of point 1 from patient 0.



(b) Trajectories of point 2 from patient 0.



(c) Trajectories of point 3 from patient 4.

Figure 1.5: Impact of the data preprocessing step on the motion of three points.

The bias introduced by the 4D-CT are clearly visible in the different figures. For example, in Figure 1.5a the 4D-CT describes a large displacement for point 1 at the end of the exhalation (towards phase 50%) covering 5 mm between two consecutive phases, whereas a small displacement is expected. In Figure 1.5b, point 2 of patient 0 seems motionless between phases $30-40$% and $60-70$%. The missing phases 10% and 80% for the points from patient 0 can also be noticed in these two figures. The expected hysteresis-like motion of the lung clearly appears in the three figures, for both 4D-CT and smoothed data.

1.3.2/ Predicted respiratory motion

The quality of the respiratory motion predicted by the NEMOSIS platform is studied using results given according to the patient considered as validation set to control overfitting. Among the five validation patients, the results obtained for patient 1 will not be presented because this patient is at the border of the definition domain, being the one whose points are the most scattered inside the lungs. However, this patient is not completely excluded from the data set, since it remains in the training set. The tables presented thereafter give the average error between the expected positions and the ones predicted by the neural network for the points of different patients. The average error is computed using only 10 phases, which is the number of phases in a 4D-CT image.

Table 1.2: Average errors (standard deviation) in mm according to the number of phases that defines the motion of the different points.

Validation	Phases			
Patient ID	10	20	50	100
0	0.88 (0.83)	0.86 (0.82)	0.86 (0.82)	0.87 (0.80)
2	1.01 (0.81)	1.05 (0.87)	1.00 (0.84)	1.02 (0.83)
3	1.23 (0.80)	1.08 (0.63)	0.92 (0.68)	1.00 (0.65)
4	1.35 (0.96)	1.36 (1.01)	1.27 (0.99)	1.27 (1.00)

Table 1.2 is devoted to the study of the impact of the number of phases describing the motion of the points in the training set. More precisely, it shows the evolution of the average error (in mm) for the points of the patient used as a validation set. Several observations can be made. First, the prediction error is the more stable for patient 2, whatever the number of phases, because the points of this patient are localized in a lung area with a simple motion. Second, conversely patient 4 has the largest errors due to the biggest lung volume. Third, 50 is the best number of phases to describe the motion and therefore this value is retained to enlarge the original data set. Obviously, the errors for patient 0 are biased by the two missing phases and more particularly the 80% one that usually exhibits large variations in motion.

Finally, we will present the motion prediction performance of the neural network obtained with patient 2 as validation set. In that case the $1,518$ points of the initial data set have led to an enlarged corrected data set of $6,200$ points used as follows:

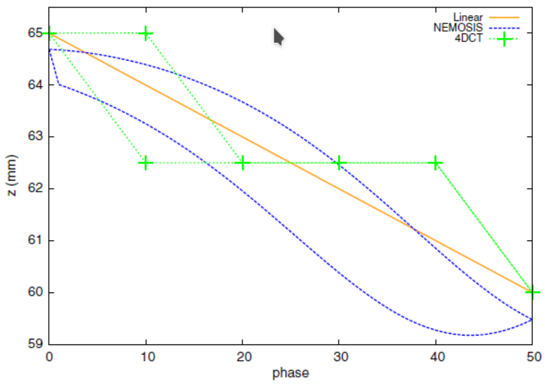
- the Corrected Learning (training) Set (CLS) is composed of $4,600$ points from patients 0, 1, 3, and 4 considering 50 phases;
- patient 2 $(1,200$ points) defines the Corrected Validation Set (CVS) ;
- the remaining 400 points compose the testing. They correspond to the feature points of patients $5-9$, which are defined over 10 phases (see Table 1.1).

Hence, the corrected data set is subdivided in such a way that the learning, validation, and testing sets respectively represent about 74%, 20%, and 6% of the whole data. Both learning and validation points have been preprocessed to improve the training of the network simulating lung motion. The testing points have not undergone this preprocessing, since in a normal execution the network will take the original 4D-CT data as input.

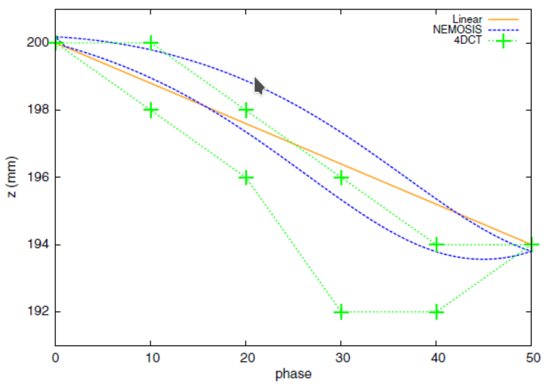
Table 1.3 presents the results obtained for the patients belonging to the testing set for two training / validation setups: one that uses the initial data set (ILS/IVS) and another its corrected version (CLS/CVS) provided by the proposed preprocessing. First, these results clearly show the relevance of the data preprocessing as the (CLS/CVS) setup always improves the prediction accuracy. The mean improvement of the accuracy is about 16% and reaches 30% for patient 9. Second, great error variations can be observed according to the patient. Indeed, our approach appears to be sensitive to the coverage of the possible patients, since the largest errors are obtained for patients 7 and 9 whose lung volumes are the most different from those of the patients used during training. A neural network can only give suitable predictions for unknown inputs similar to some inputs used during the training. For patients 5, 6, and 8, the errors are lower because their respective lung volume is close to one of the patients from the learning set.

Table 1.3: Average errors (standard deviation) in mm obtained for the different test patients (Initial / Corrected - Learning / Validation Sets).

Patient ID	ILS / IVS	CLS / CVS
5	1.15 (0.68)	1.02 (0.62)
6	1.45 (0.55)	1.25 (0.72)
7	2.30 (2.32)	2.00 (2.26)
8	1.37 (0.96)	1.16 (0.83)
9	2.18 (1.49)	1.53 (1.15)



(a) Motion simulated for patient 6.



(b) Motion simulated for patient 8.

Figure 1.6: Motion simulated for two points, each belonging to a different patient.

In order to further check if NEMOSIS is able to produce a realistic lung motion for normal tissues, we carried out a comparison of its prediction with a linear approximation. This approximation is obtained through a linear regression between the maximum exhalation and expiration phases (0% and 50%), without taking into account the hysteresis. It leads to a symmetrical motion between the phase ranges $0 - 50\%$ and $50 - 100\%$. Figure 1.6 shows the motion predicted for two distinct points from testing sets, together with the trajectories obtained by linear regression and from 4D-CT data. The first one (a) belongs to patient 6, while the second one (b) is from patient 8. Obviously, the linear approximation is not accurate enough to describe the motion, being ineffective to simulate a motion whose gradient depends on the phase, as highlighted by the 4D-CT data. Conversely, the motion predicted by NEMOSIS is more relevant: it exhibits an hysteric motion closer to the 4D-CT. Furthermore, when the 4D-CT produces a motion without hysteresis, as in Figure 1.6(a), the neural network simulates a motion that has one. The minimum position of both points is between the 50% and 60% phases, while we expected the 50% phase to be the minimum, because such behavior appears in the data used for the training.

This latter observation is reinforced by the average prediction errors, shown in Figure 1.7, for the points belonging to patient 8. According to the respective error histograms, computed without considering the 0% and 50% phases since the error would have be 0 for the linear regression, the neural network gives in average positions which are closer to the 4D-CT and have a lower standard deviation. We can also note that compared to the 4D-CT images the average prediction errors are less than 2 mm, except for point 2, and that is lower than the z-axis image resolution.

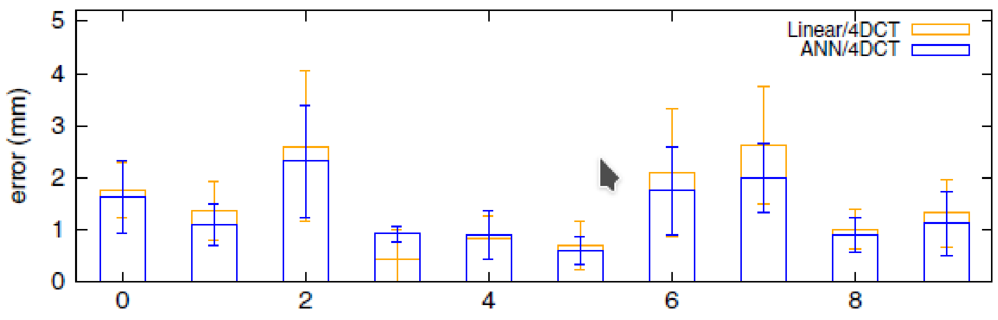


Figure 1.7: Comparison of NEMOSIS with linear regression: average prediction errors for all points of patient 8, considering almost all phases (except 0% and 50%).

1.4/ Discussion

Even if the proposal is based on 4D-CT data that contain artifacts, both data preprocessing and NEMOSIS have been validated, and thus we have a tool able to accurately simulate the lung motion. Unfortunately, in spite of the good performances shown previously, this version of NEMOSIS is not suitable for clinical use due to several limitations. First, NEMOSIS takes input data from two extreme breathing phases (0% and 50%) and even if the data for each phase could be acquired independently using a 3D-CT, in practice it is not possible. Indeed, it would suppose that the patient can control its breathing and block its airflow for a long time, which is usually difficult for patients suffering from lung cancer. Thus, to obtain the images of both phases, the 4D-CT scanner that has a much higher dosimetric impact than its 3D counterpart (7 times more dose delivered) must still be used. Second, as the objective is to follow the tumor, the radiation therapists must define the tumor contour in both phases. However, such contouring is again practically intractable.

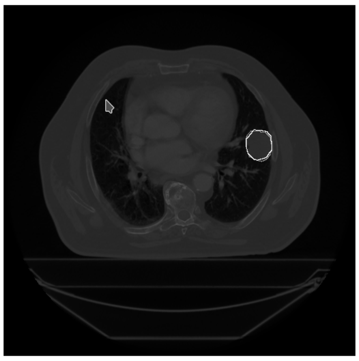
In order to improve the patient's condition effectively, an optimized version of NEMOSIS [3] using only 5 inputs (NEMOSIS(5)) instead of 8 (inputs) (NEMOSIS(8)) has been studied. In this new version, only one respiratory phase, namely the 0% phase, is used to define the coordinate of a point. Hence, a single 3D-CT image, in which the tumour is contoured, is sufficient to provide a localized and customized lung motion using a 3D model of a patient. Compared to NEMOSIS(8), the "optimized" NEMOSIS(5) version is less accurate: in the most favorable case, when patient 2 is used as a validation set, errors increase by 35%. That suggests that the deleted 50% respiratory phase brings a valuable information allowing to obtain more accurate predictions. In fact, both phases give a measure of the motion amplitude, an information complementary to the lung volume, which characterizes the patient. The study presented in [3], based on three new patients using images focused on the tumor, has shown that, in the worst case, an average accuracy of 3 mm for all points of a patient, at every phase, is obtained. Another point investigated in [3] to assess the potential clinical use of NEMOSIS(5), although the prediction are less accurate, is a comparison of the GTV (Gross Tumor Volume) contoured by a radiation therapist and the one computed by NEMOSIS(5). A Dice (or Sorensen) index, which evaluates the similarities between two sets through a value ranging from 0 and 1 (this latter value means a perfect matching), of 0.80 was computed between these two volumes. Figure 1.8 shows tumor contours over one slice from the three patients for the 30% phase.

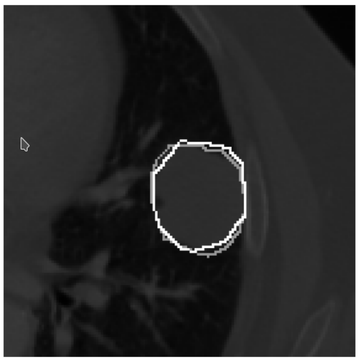
More recently, a customized 4D numerical phantom representing the organ contours based on a new adaptation of NEMOSIS has been explored [25]. Therefore, in order to apply NEMOSIS to anthropomorphic phantoms several adjustments have been made. First, an increase of the data set through the addition of both new patients and the overall lung contours for some of them. Thus, the data set consists of $37,160$ points for 15 patients over 10 phases, comprising of $4,680$ anatomical feature points and $32,480$ contour points. Second, a modification of the inputs of the neural network to have a better customization by taking into account two parameters giving an approximation of the real pulmonary volume by a cylinder. This new version of NEMOSIS has thus 7 inputs and is denoted thereafter by NEMOSIS(7).

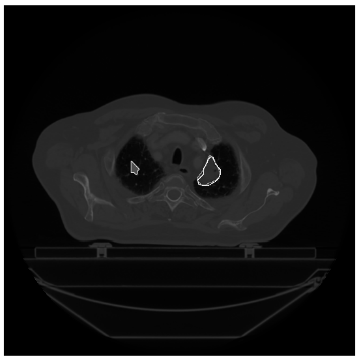
After training with data points belonging to 12 patients, NEMOSIS(7) gives for the remaining 3 patients a Dice index greater than 0.93 with 4D-CT during a respiratory cycle. From a neural network topology point of view, the incremental learning process has produced a hidden layer of 20 neurons. Figure 1.9 presents the breathing motion produced by NEMOSIS(7) for the 185 cm phantom (female phantom from IRSN). The figure displays the phantom at five respiratory phases: 10%, 30%, 50%, 70%, and 90%. A color code highlights the amplitude of a point's motion, from its current position to its position in the next phase, and the initial lung contour at phase 0% is plotted in gray line for each phase to render the global motion. Hence, we can see that the lungs mainly move in the Superior-Inferior (SI) axis in accordance with the reality. Overall, NEMOSIS(7) seems to predict the maximum exhalation slightly earlier than in reality and is still limited by the small data set which is not representative enough.

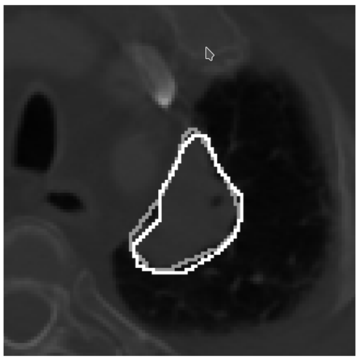
1.5/ Conclusion

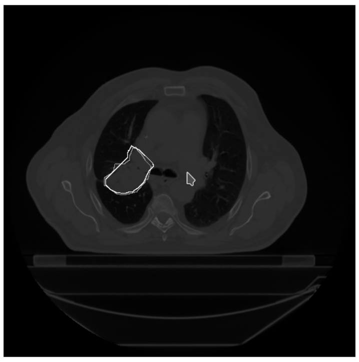
A challenging problem in lung cancer radiation therapy is the management of respiratory motion, since breathing motion is a crucial problem for radiation therapy treatment of many tumors. To address this problem, we have investigated an approach allowing a customized simulation of lung motions using multilayer perceptron neural networks. The idea is thus to learn the lung motion using some patient data in order to predict the breathing of unknown patients. To begin, before trying to simulate the lung motion, we have used neural networks to enlarge the initial limited and noisy data set obtained by plotting anatomical feature points in 4D-CT scans. Then, we have designed a first version of the NEural NEtwork MOtion SImulation System (NEMOSIS) to reproduce for normal tissues the movements included in the training domain as realistically as possible. The domain does not cover the whole lung due to the initial limited data set. A second version, better suited to a potential use in clinical routine, needing only the coordinates of anatomical feature points in the first respiratory phase has been proposed. This latter version, in which NEMOSIS has less inputs, requires thus only data provided by a single 3D-CT scan and no more by a 4D-CT one. Despite less precise predictions, this second version allows to simulate the evolution of tumors contours which, in comparison with the contours manually plotted by a radiation therapist, have a Dice index stating a minimum correspondence of 0.80. More recently, promising results have been obtained with a third version of NEMOSIS for a customized 4D numerical phantom representing the lung contours. The resulting lung model has a Dice index of 0.93 with the 4D-CT scan defining a respiratory cycle.


(a) Slice from patient 1.


(b) Zoom on the tumor from patient 1.


(c) Slice from patient 2.


(d) Zoom on the tumor from patient 2.


(e) Slice from patient 3.

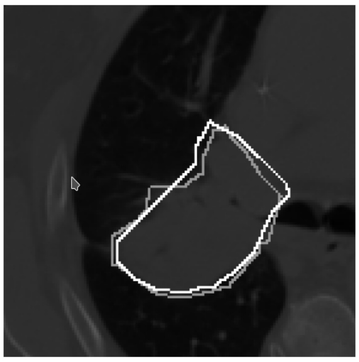

(f) Zoom on the tumor from patient 3.

Figure 1.8: Tumor contours on one slice from three patients. In gray the contours given by the radiation therapist and in white by NEMOSIS(5).

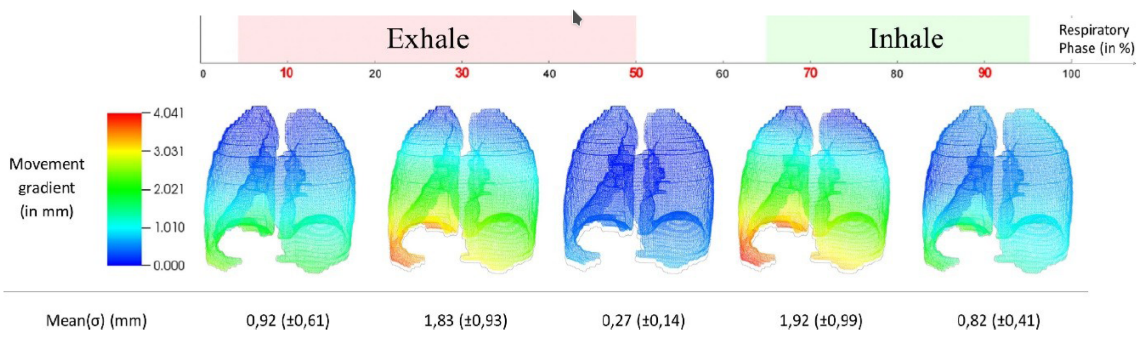


Figure 1.9: Simulation of the lung motion using NEMOSIS(7) on the 185 cm phantom.

2

Active MEMS-based Flow Control

Since 2012 the FEMTO-ST Institute consists of seven research departements, among which the DISC department is the last one to have been created. This latter is issued from the including of the former Computer Science Laboratory. To support this integration and give a scientific consistency to it, collaborative researches with other departments integrating transversal competencies have been encouraged. Therefore, with two colleagues of the AND team, Karine Deschinkel and Jean-François Couchot, we have started in 2010 a collaboration with members of the THERMIE team of the ENERGY department. A research topic of the THERMIE team is the study of complex flows such as hydrodynamic flows in presence of ultrasounds or supersonic jets. The purpose of the collaboration was thus to assess the ability of the multilayer perceptron neural network to give a relevant modelling of the airflow evolution and then to study the application of Micro-Electro-Mechanical Systems (MEMS) for aerodynamical active flow control. First investigations were supported by two Master of Science internships, one in each team.

Various research works have studied this problem, meanwhile controlling such MEMS-based systems remains a challenge. Let us notice that the choice of an active control using MEMS is further motivated by the fact that one of the scientific aims of another department of FEMTO-ST, the MN2S department, is precisely the design and construction of multiphysics micro and nanosystems. Several control approaches for time varying systems exist, many of them use a process model representing the dynamic behavior of the process to be controlled. Our research work, which was first presented in a workshop [26] and later published in a issue of selected papers from it in the Mechatronics international journal [27], is such a approach. Indeed, after having established the suitability of a multilayer perceptron to predict the force corresponding to the flow induced by the MEMS, we have optimized the flow w.r.t a numerical criterion. We have considered a dynamic flow over a backward facing step (a rough model of the back of a car) where MEMS actuators velocities are adjusted to maximize the pressure applied on the step surface.

The first step of the work was to set up a database thanks to computational fluid dynamics simulations using a model resulting from the Master of Science internship in physics supervised by the colleagues of the THERMIE team. Then, two approaches to find a multilayer perceptron network topology giving suitable predictions have been investigated: one that uses a trial and error process and another that dynamically builds the single hidden layer. Finally, a dynamic control of the flow by changing the MEMS configuration, leading to promising results, is explored.

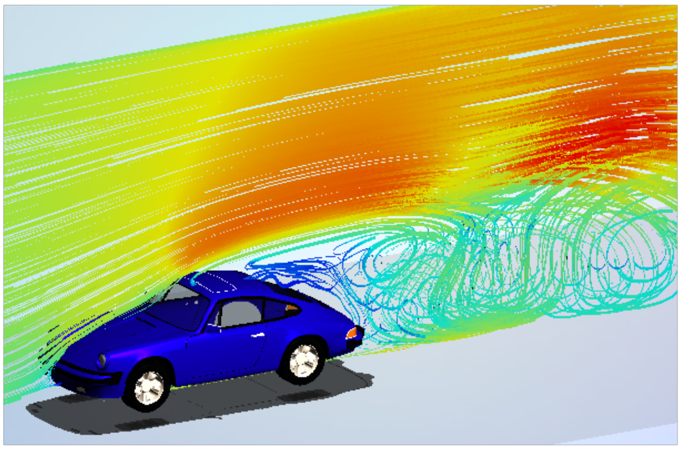


Figure 2.1: Image from an Autodesk tutorial displaying flow lines over a car.

The remainder of this chapter, which is a short version of the journal paper [27], is composed of the following sections. In Section 2.1 active MEMS-based airflow control is discussed, in particular existing works. Section 2.2 describes precisely the considered dynamic flow and the corresponding computational fluid dynamic model. The next section is devoted to the design of a multilayer perceptron for the prediction task, the dynamic building and training approach gives the better results. Finally, before concluding, we propose a control strategy that exploits the process model provided by the MLP.

2.1/ Introduction

The active control of turbulent flows for drag reduction in order to improve the aerodynamic performance of moving objects, such as an aircraft or a car, has received a lot of attention. This interest comes from several important potential benefits for economy and environment of such a control. Indeed, it would allow a better maneuverability, an increasing of the range or payload capability, and improved environmental compliance. For example, an integrated flow control system in a car would permit to control the massive recirculation zone which can be observed behind it, as shown in Figure 2.1, and thus reduce the noise and pollutants emission.

Among the various microsystems that can be used to do active fluid control, MEMS [28] are an interesting solution due to their properties: small size, low power consumption and cost, and real-time control. In fact, with this technology, a fluid flow is controlled by continuously adjusting MEMS actuators based on some information given by MEMS sensors. Moreover, one of the most likely futuristic vision of active control consists of large sets of interacting MEMS flow sensors and actuators built on aerodynamic surfaces. A practical implementation of a set of pulsed air-jet actuators, using a rough control approach by simply switching them on at high speeds, has been done on a Citroen C-SportLounge concept car in 2005. Therefore, this vehicle had an virtual airfoil on the tail using 40 MEMS actuators producing pulsed microjets, supposed to improve the vehicle's drag and stability at high speeds. Figure 2.2 describes in the case of an aircraft other possible approaches to perform active airflow control than through actuators producing modulated pulsed air-jet, namely vibrating surfaces and asperities on the leading edge of the wing.

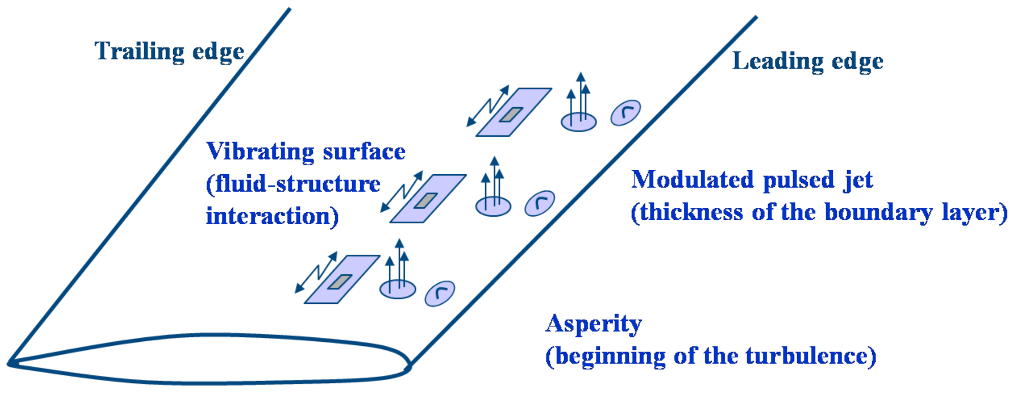


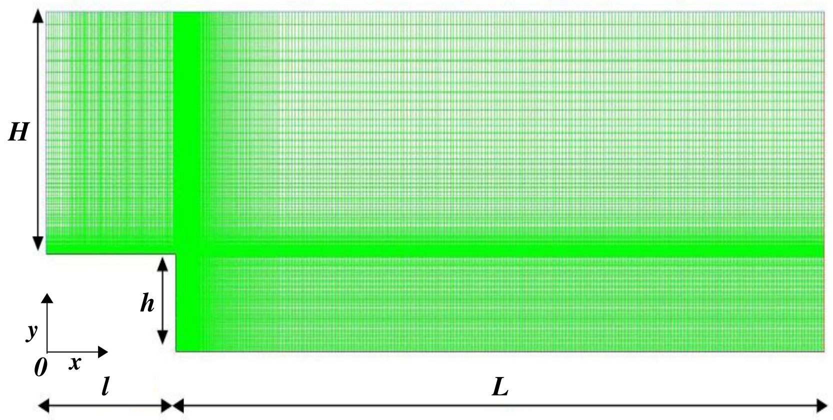
Figure 2.2: Different kinds of MEMS allowing to control the airflow on the upper part of an aircraft wing.

At the present time, the practical control of a turbulent flow with MEMS is a challenging scientific problem in fluid mechanics. Furthermore, when designing a control method one must not only focus on the performance, but must also keep in mind several requirements that have to be addressed: scalability, robustness, and reliability. To build a closed-loop flow control system two components have to be designed: a model that captures the actuation effects on the flow and a controller.

Fluid models are well described through Navier-Stokes equations, which are highly nonlinear partial differential equations, therefore a nonlinear model is needed. This kind of equations is usually solved thanks to computational fluid dynamics (CFD) tools like FLUENT [29]. Although state of the art CFD tools can predict what will happen, as they handle a huge number of data in an iterative process a prohibitive time is usually required to compute the future forced flow. Thus, they are irrelevant for real-time active control of turbulent flows. Worst, they cannot be applied in a context where inflows velocity is always changing. Reduced-order models can be broadly categorized into three classes [30]: physical, mathematical, and data-fitting methods. In the first model category some physical processes are neglected, in the second one the flow field representation is changed to a more convenient one, whereas in the latter category the model is constructed in order to fit data resulting from experiment or numerical simulation. For the considered control problem, the chosen model should offer the best balance between accuracy and efficiency.

An adaptive active closed-loop separation control on a rounded step has been presented by Pamart *et al.* in [31]. They showed that it is possible to automatically control the recirculation bubble using a synthetic jet frequency effect and measures provided by a unique wall pressure sensor. Their control approach combines a nonlinear polynomial NARX black-box model, which is obtained through unsteady RANS computations (a Reynolds-averaged Navier-Stokes model), and an improved extremum-seeking algorithm. We consider a similar context: the control of a turbulent wall bounded flow over a backward facing step. We first focus on the identification of a nonlinear model that can predict the forced flow. More precisely, our objective is to show the suitability of a multilayer perceptron neural network, a data-fitting approach, to play the role of process model. Later, we show that this neural network might be easily used to provide a convenient active fluid control. Let us remark that in comparison with [31], in our case study we take into account multiple MEMS actuators producing pulsed microjets [32].

(a)



(b)

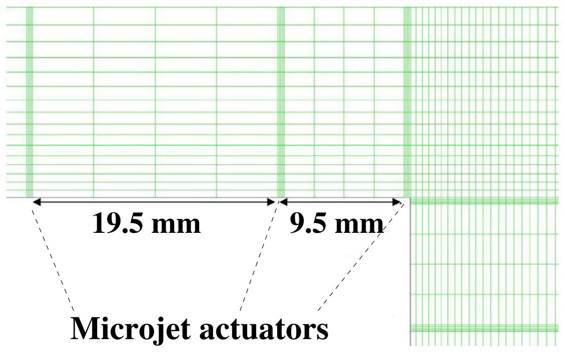


Figure 2.3: Backward facing step geometry and nonuniform 2D mesh (a), and MEMS microjet actuators location around the step corner (b).

2.2/ Problem specification and computational fluid dynamic model

We focus on the dynamic flow over a static step, considering an airflow in the left-right direction as a case in hand that may approximate a car moving in the right-left direction.

The geometry specification of the flow is displayed in Figure 2.3(a). O, x, and y represent, respectively, the origin of the coordinate system, the longitudinal, and vertical directions. This backward facing step is of height $h = 0.288$ m and $l = 0.5$ m long. The upstream section has a height H of 0.712 m and the longitudinal length L of the downstream section is equal to 2.5 m. Finally, the flow is initialized at atmospheric pressure with a temperature of 300 K, which corresponds to a classical ambient temperature.

Five MEMS microjet actuators are placed on the upper part of the step. Each actuator has a slot width of 500 μm and is supposed to produce, when activated, a microjet with a velocity (expressed in m.s^{-1}) that takes discrete values in $\{0, 33, 66, 99\}$. Figure 2.3(b), which is a zoom around the step corner, shows how the MEMS actuators are placed. The first one is placed on the step corner, the second is 10 mm away from the corner, while the last three actuators are respectively 30 mm, 50 mm, and 70 mm from the corner. Notice that each air microjet is always oriented in the bottom-up direction.

Figures 2.3(a) and 2.3(b) also highlight the computational fluid dynamics grid-mesh used to perform the numerical simulations with the CFD code FLUENT. We chose a nonuniform mesh in both longitudinal and vertical directions. The grid is coarser before the step, refined close to the step walls, and the finest near the actuators position. This grid-mesh, composed of more than $150,000$ cells, offers a good compromise between the computation time for one iteration and the accuracy of the simulation. The average computation time to complete a simulation carried out on a regular Linux workstation with an AMD Athlon(tm) 64 X2 dual core processor 4000 + running at 2.1 GHz is about 5 minutes. Note that we made the following assumptions for the boundary conditions. At the inlet, a uniform velocity profile with an inflow velocity (expressed in m.s^{-1}) that again takes discrete values in the set $\{30, 35, 40\}$. In addition, the outlet outflow boundary conditions are used and wall functions at the lower wall. At an actuator location, the boundary layer is similar to the inlet one when activated and to the lower wall layer otherwise.

Intuitively, reducing the vacuum that may appear along the down step may be seen as an application of this work. Then, among the results returned by FLUENT, we focus on pressure data for this step face. However, since pressure is the amount of force per unit area, dealing with pressure is memory consuming: for each configuration there are as many pressure data than there are grid points on the down step, *i.e.* 244 results. For each configuration, we rather aggregate results in a force obtained as the integral of pressure over the step surface. This force is numerically computed thanks to the classical trapezoidal rule. The neural network presented in the next section aims at approximating the calculus of this force.

2.3/ Predicting the flow evolution using a multilayer perceptron neural network

To predict the forced flow resulting from an actuation configuration a nonlinear process model is required. Hence, we search a nonlinear mapping between inputs defining the current flow and an actuation configuration, and outputs describing the flow evolution.

2.3.1/ A first network design

2.3.1.1/ Network topology and implementation

The first objective is to have a neural network able to model the resulting flow for various values of boundary velocity in $[30, 40]$ (m.s^{-1}) and for any configurations of MEMS actuators in $\{0, 33, 66, 99\}$[5]. Practically, that means that the network has for input a couple (C, v), where C describes how the actuators are set and v defines the boundary inflow velocity. In fact, C is a vector $(c_0, c_1, c_2, c_3, c_4)$ whose components represent the velocity of a microjet produced by an injector taking discrete values in $\{0, 33, 66, 99\}$ (m.s^{-1}), such that the larger a component index, the closer the corresponding actuator to the step angle. Thus, at a first glance, the neural network would have 6 inputs that produce a single output: the resulting force $f(C, v)$ expressed by the integral of pressure over the backward step surface. However, in order to improve the neural network performances, more precisely its ability to deal with nonlinear relationships that represent very sharp variations and to have a faster training, we decided to change the structure of the inputs for the Higher-order Processing Unit (HPU) one [33]. The principle of this latter structure is to artificially increase the number of inputs by adding polynomial combinations of the initial inputs up to a given degree, called the order of the network.

The MLP is composed of a single hidden layer of sigmoidal neurons and linear output ones, a classical case. It is trained through supervised learning by feeding it with data obtained from the FLUENT simulation, using as optimization algorithm the L-BFGS algorithm like in the work presented in the previous chapter. To address the overfitting problem, we retained the popular holdout validation. As we consider a case study of 5 actuators, each one producing a microjet having a velocity value chosen among 4 values, and 3 different inflow velocities $v \in \{30, 35, 40\}$ (m.s^{-1}), the data set is composed of $4^5 \times 3 = 3,072$ samples. These data samples are then randomly distributed over the training, validation, and testing sets. The random sampling is performed such that the cardinalities of these sets are, respectively, 65%, 10%, and 25% of the whole data set. To find a suitable number of hidden neurons, in a first step we used a trial and error process.

2.3.1.2/ Results

Different single hidden layer MLPs are considered. They differ from each other in number of hidden neurons: 15, 20, 25, and 30 neurons networks are studied. Some preliminaries experiments have also shown that the best results are gained with HPU neural networks of order 3. Thus, this structure, which replaces the 6 initial inputs with 83 ones, has been retained. Despite the large increase in the number of synaptic weights, this approach allows to improve the results. To control the training process, different maximum number of training epochs are used: 500, 750, and 1000 epochs.

The accuracy of the predictions is assessed using two statistical measures on the predictions obtained for the data belonging to the testing set. First, the Coefficient of Variation of the Root Mean Squared Error (CVRMSE) expressed by:

$$\%\text{CVRMSE} = \frac{100}{\overline{o_k}} \cdot \sqrt{\frac{\sum_{k=1}^{N} (o_k - p_k)^2}{N}}, \tag{2.1}$$

where o_k is the observed output of the k–th input-output test pair, p_k the corresponding prediction, $\overline{o_k}$ the mean value of the observed values, and N the testing set cardinality. Second, the coefficient of determination R^2, mostly defined as follows:

$$R^2 = 1 - \frac{\sum_{k=1}^{N} (o_k - p_k)^2}{\sum_{k=1}^{N} (o_k - \overline{o_k})^2}. \tag{2.2}$$

These statistics give a fair estimation of the ability of a MLP to reflect the variance of the data and how well it will predict the output value. The closer the CVRMSE to 0 and R^2 to 1, the better the neural network model. While the CVRMSE aims at describing if we have a good model fit by measuring the residuals relative to the predicted value, the coefficient of determination is indicative of the level of explained variability in the data set.

Table 2.1 presents for the different training setups the mean values of CVRMSE and R^2. These ones are computed using each time 20 trainings with random sets, synaptic weights, and biases initialization. As one can see, the different neural networks offer very similar performances. In fact, the closeness of the two statistical measures for the four network topologies, whatever the maximum number of epochs, can mainly by explained

Table 2.1: Performances of the different networks for a testing set of 769 data samples.

Topology	6 initial / 83 HPU inputs and 1 output		
Hidden neurons	Epochs	%CVRMSE	R^2
15	500	10.26	0.9475
	750	10.21	0.9480
	1000	10.23	0.9477
20	500	10.20	0.9481
	750	10.14	0.9487
	1000	10.15	0.9486
25	500	10.31	0.9469
	750	10.28	0.9473
	1000	10.26	0.9475
30	500	10.25	0.9476
	750	10.20	0.9480
	1000	10.20	0.9481

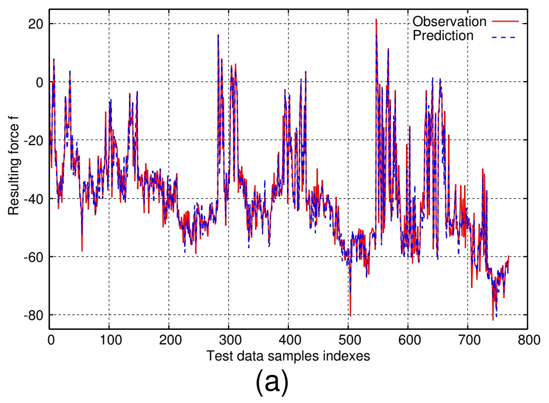
by the HPU structure. The better multilayer perceptron has 20 sigmoidal neurons on its hidden layer and was trained during 750 epochs. Let us notice that the lowest validation error was observed in that case after 579 training epochs. In comparison, a MLP for simple nonlinear regression (with a single hidden neuron and no HPU structure) has a CVRMSE of 23.99%, which is more than twice the coefficient value of any network, and means that a simple nonlinear regression model is not relevant.

Figures 2.4(a) and 2.4(b) further illustrate the quality of the predictions given by the chosen MLP. The first figure plots the observations and predicted values for the whole testing set, while the second one is a zoom on data samples having an index in the range 500 to 700. This positive point of view on the prediction performance of the neural network is further reinforced by the following figures: on the one hand, Figure 2.5(a) presents the absolute error obtained for each of the 769 test data samples; on the other hand Figure 2.5(b) is a view of the corresponding error distribution. It can be seen that the network gives relevant predictions and the error distribution histogram follows a normal distribution. More precisely, the errors are normally distributed with a mean of 0.236 and a standard deviation of 3.428, which means, according to the three-sigma rule, that the prediction error lies between -6.62 and 7.092 for 95% of the test data samples. Therefore, we can say that the multilayer perceptron neural network can model the flow evolution with a very good accuracy.

2.3.2/ An improved network design

A major question that must be addressed is the size of the hidden layer. A too large layer will need many epochs to be well-trained, which is not the case with a small network, but a too small network may be unable to learn the relationships, or with a long training time will get trapped into overfitting. Therefore, the optimal number of hidden neurons is the one which offers the better trade-off between the computation time needed to fulfill the training process and networks' generalization ability.

In the previous section, we have used a classical trial and error process to find a suitable number of hidden neurons. Although, thanks to this method, we have shown that a hid-



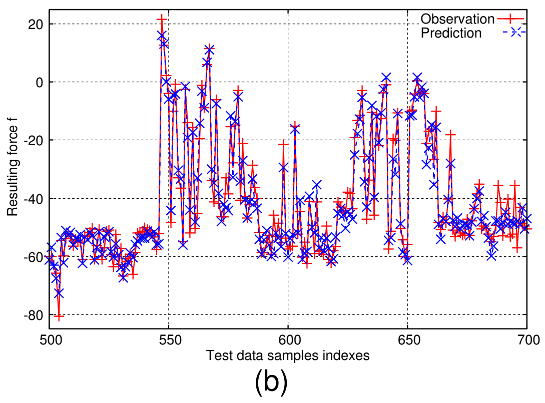
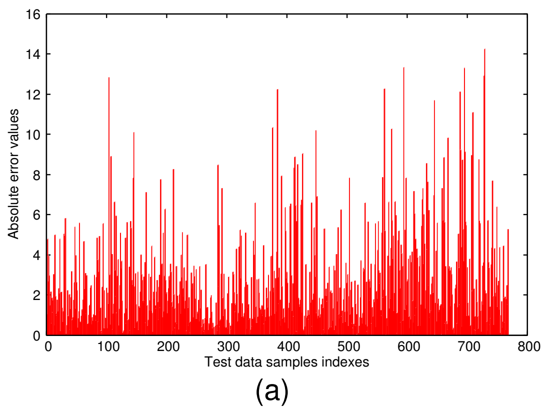


Figure 2.4: Observed and predicted values for the test data samples (a) and zoom on the samples with an index going from 500 up to 700 (b).



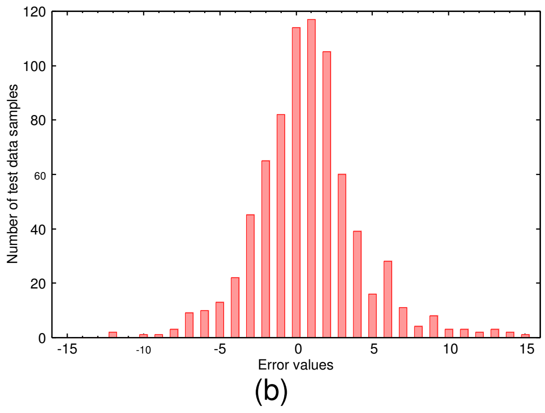


Figure 2.5: Absolute error values (a) and error distribution histogram (b).

den layer of 20 neurons allows to obtain relevant predictions it has a major drawback: the waste of time due to the training of many different MLPs. An interesting alternative consists in a dynamic building and training of the network. Thus, the purpose of this section is first to describe the principle of this kind of approaches, and then to study its relevance in our context. Let us notice that dynamic neural network construction approaches have been successfully applied in various domains, like the prediction of dose deposit during external radiotherapy treatments [23], or the simulation of respiratory lung motion [2, 3, 4] described in the previous chapter.

2.3.2.1/ Dynamic building and training

Many dynamic approaches have been proposed to simultaneously define an optimal network topology and train it. The background idea is to start with few hidden neurons and then to successively add new neurons, one after another, as long as the overfitting problem does not appear. The proposed algorithms differ from each others in the way the new neurons are added and the training is performed. In [34] when a new neuron is added only its synaptic weights and bias, which are initialized to zero, are updated by the training process, while in [23] it is the weights and biases of the whole network. Usually, dynamic methods construct networks with less hidden neurons than a trial and error process.

A dynamic building method similar to the incremental approach proposed in [23] is considered. Such an approach has been already used in the work on the simulation of lung motion, but the addition of a new hidden neuron was controlled in a different way. Let us give some details on the algorithm and particularly on the approach used to stop it.

The algorithm starts with a MLP that has few neurons on a single hidden layer, next it trains the network during a fixed number of iterations, then it adds one hidden neuron and performs the training again, and so on until the addition of a new neuron does not produce more accurate predictions, or when the hidden layer reaches a maximum size (typically 30 neurons). In the first case, the retained network will be the one obtained previously, with one less neuron in the hidden layer. We initialize the parameters of a new hidden neuron to zero. After addition of a neuron, the training of the new network is stopped after a fixed number of iterations, and also when a criterion defined in [35] is satisfied.

This stopping criterion uses the generalization loss at an epoch and the training progress measured during a sequence of consecutive epochs: stop as soon as the quotient of generalization loss and progress is a above a fixed value α. Obviously, high generalization loss indicates overfitting, *i.e.*, a practical reason to stop training. However, a fast training progress (training error decrease) may reduce the generalization loss. Indeed, as noticed by the author of [35], overfitting begins when the training error decreases slowly. The length of the sequence used to estimate the training progress (10 epochs) and the value of α (7.5) were chosen after some preliminary experiments, and thus may not be optimal.

2.3.2.2/ Results

The same data set and random sampling are used to build the training, validation, and testing sets. Again, we control the training process at beginning and after addition of a new hidden neuron using various numbers of training epochs: 50, 100, 150, 200, and 250 epochs. For example, if the number of training epochs is 50 and two hidden neurons are finally added, this means that the maximum number of training epochs will be $50+2\times 50+50 = 200$ epochs. Indeed, 50 epochs are used to train the starting network, followed by 50 epochs for each added neuron, and finally 50 epochs are used to train the network with a third new neuron that will not be retained. This maximum number of training epochs can be compared to the one used for the networks presented in Subsection 2.3.1.2.

Table 2.2 presents the performances obtained for different starting neural networks: the initial hidden layer size ranges from one to 15 neurons. For each training setup are given the mean values of the final number of hidden neurons, CVRMSE and R^2. Like in Table 2.1, these values are computed considering 20 trainings. The first observation which can be made is that for almost each starting hidden layer size, a final network giving predictions with better or similar accuracy than that of the previously retained network of 20 hidden neurons is obtained (shown in Table 2.1). In particular, networks starting with one and 10 hidden neurons provide the most accurate final networks: a network composed of 7 neurons, neurons which are trained successively during 100 epochs, and one of 14 neurons, each new added neuron being trained during 200 epochs. Overall, in average the better networks constructed by the dynamic approach consist of 7 neurons and have required $100+6\times 100+100 = 800$ epochs for the whole process. These networks are far more smaller than the best one found through trial and error, which has almost three times much more hidden neurons. Such a network will be used to predict the resulting force along the down face of the backward facing step when optimizing the flow.

Table 2.2: Performances of different dynamically build and trained networks for a testing set of 769 data samples.

Hidden	6 initial / 83 HPU inputs and 1 output			
Start neurons	Epochs	Final neurons	%CVRMSE	R^2
1	50	8	10.14	0.9486
	100	7	10.05	0.9495
	150	5	10.64	0.9086
	200	4	10.61	0.9192
	250	4	10.65	0.9429
5	50	11	10.42	0.9457
	100	10	10.14	0.9487
	150	9	10.17	0.9484
	200	8	10.19	0.9482
	250	8	10.14	0.9487
10	50	18	10.39	0.9462
	100	15	10.20	0.9482
	150	14	10.15	0.9487
	200	14	10.11	0.9489
	250	13	10.12	0.9489
15	50	22	10.60	0.9437
	100	20	10.27	0.9474
	150	19	10.19	0.9482
	200	19	10.17	0.9484
	250	19	10.19	0.9221

Another observation concerns the number of epochs needed to train a network after addition of a neuron. It can be seen that the most suitable number increases with the size of the starting hidden layer. Indeed, the larger the neural network, the larger the number of epochs needed to properly train it.

2.4/ Optimizing the flow with a model predictive control

2.4.1/ Model of control

The objective is to adjust in real time the microjet velocity of the different injectors in order to maximize the force $f(C, v)$, which is induced by the downstream recirculation area, that depends on the injectors configuration C and the boundary inflow velocity v. We suppose that the boundary velocity varies over time and we try to change from one configuration to another in minimum time to maintain a maximal force.

Ideally, the controller should make the MEMS actuators change instantaneously from the current configuration $C = (c_0, \dots, c_4)$ to the optimal one $C^*(v)$. However, this is not really practicable. Indeed, even if all the MEMS are simultaneously modified, this change takes time δt. Let then v' be the boundary velocity at time $t + \delta t$. The velocity v' may be distinct from v and thus $C^*(v')$ is not necessarily equal to $C^*(v)$. Therefore, we have decided to restrict the way the MEMS configuration evolve by considering rather that the change of configuration is done according to a neighborhood structure. The idea is to consider only changes in the shortest time and thus to enable only a transition from the current configuration to a configuration that is only a little different. Let $N(C)$ be the neighborhood defined as follows:

$$N(C) = \left\{C' = \left(c'_0, \ldots, c'_4\right) \mid \left|c'_j - c_j\right| \leq 33 \text{ and } 0 \leq j \leq 4\right\}. \tag{2.3}$$

According to this definition, $N(C)$ is composed of configurations C' such that the velocity difference $\left|c'_j - c_j\right|$ for each injector $j \in \{0, \ldots, 4\}$ will be at most 33 m.s^{-1}. Then, for any given configuration C, and any boundary inflow velocity v, we determine the configuration $C' \in N(C)$ that maximizes the force f behind the backward-facing step:

$$\tilde{C} = \text{argmax}_{C' \in N(C)} f(C', v) \tag{2.4}$$

However, it is unlikely that the inflow velocity remains unchanged and therefore Equation (2.4) must take into account the possible evolution of the velocity. Indeed, given a sequence of velocities v_i measured at time $t_i = i \times \delta t$, the force applied behind the backward-facing step when the boundary inflow velocity is v_i results from the MEMS configuration optimized for the velocity at time t_{i-1}: $f_i = f\left(\tilde{C}(v_{i-1}), v_i\right)$, and obviously the more the velocity changes, the more $\tilde{C}(v_{i-1})$ may be different from the optimal configuration $\tilde{C}(v_i)$. Therefore, we should compute $f(C', v')$ for all the C' in $N(C)$ and for all the velocities v' that are close to v_{i-1}.

For efficiency reasons, the set of considered velocities is obtained using the current velocity and the current acceleration. Formally, let a_{i-1} be an approximation of the acceleration at time t_{i-1}, *i.e.*, $a_{i-1} = v_{i-1} - v_{i-2}$. When the acceleration is positive ($a_{i-1} \geq 0$), a set of velocities $\{w_0, w_1, w_2, \ldots, w_k\}$ such that $w_0 < w_1 < w_2 < \ldots < w_k$ is computed as follows. The maximal value w_k is defined as the minimum between the maximal value of the velocity (*e.g.*, 40 in our case) and $v_{i-1} + a_{i-1}$. Let $\delta_{i-1} = (w_k - v_{i-1})/k$. Then let w_0 and w_j, $1 \leq j \leq k-1$, be respectively defined by $w_0 = v_{i-1} - \delta_{i-1}$ and $w_j = v_{i-1} + j \times \delta_{i-1}$. Note that $[w_0, w_k]$ contains v_{i-1} which is located on the left part of this interval. A similar reasoning is applied to have $w_0 > w_1 > w_2 > \ldots > w_k$ when the acceleration is negative.

We are then left to compute $\tilde{C}(v_{i-1})$ that takes into account these ranges of velocity. The configuration $\tilde{C}(v_{i-1})$ is then defined by:

$$\tilde{C}(v_{i-1}) = \text{argmax}_{C' \in N(C)} \frac{1}{k+1} \sum_{j=0}^{k} f(C', w_j) \tag{2.5}$$

In other words, for any configuration C' in the neighborhood of C, we compute the mean of $f(C', v)$ where v belongs in $\{w_0, \ldots, w_k\}$, using suitable approximations of $f(C', w_j)$ provided by the multilayer perceptron composed of seven hidden neurons.

2.4.2/ Validation

The proposed model of control has been evaluated on scenarios developed as follows. A scenario is composed of a random initial configuration C_0 and a sequence of n real numbers randomly picked in the range $[30, 40]$, each one representing a velocity v_i measured at time $t_i = i \times \delta t$. Due to physical constraints, if v_{i+1} is greater than v_i then we have $v_{i+1} - v_i < k_a$, while conversely, if v_{i+1} is lower than v_i then we have $v_i - v_{i+1} < k_d$, where k_a

(resp. k_d) stands for the acceleration (resp. deceleration) bound. Intuitively, it means that between two measures, the system cannot accelerate more than k_a m.s^{-1} and cannot decelerate more than k_d m.s^{-1}. k_a and k_d were respectively set to 2.5 and 5.0 m.s^{-1}.

The relevance of the proposal is estimated by comparing two sums:

$$\sigma = \sum_{i=1}^{n-1} f_i = \sum_{i=1}^{n-1} f\left(\tilde{C}(v_{i-1}), v_i\right), \tag{2.6}$$

$$\overline{\sigma} = \sum_{i=1}^{n-1} \overline{f_i} = \sum_{i=1}^{n-1} f(C_0, v_i). \tag{2.7}$$

The former is the sum of the forces produced by updating the MEMS configuration thanks to the model of control, while the latter is observed when the configuration remains unchanged and equal to C_0.

Based on 20 scenarios randomly built such that each sequence of velocities had a length of 10 values, we can say that the proposed control allows to produce forces which are always positive, whereas without it negative forces are systematically observed. A negative force means an air resistance (drag) that slows down the car. More precisely, the average value of $\overline{\sigma}$ is -328 N, while the one obtained for σ is 201 N, *i.e.*, an increase of 529 N. It clearly shows that the optimization process allows an increase of the force behind the backward-facing step. From computational time point of view, on a typical personal computer (processor Intel Core i5 running at 2.50 GHz and 6 GB RAM) about 7.8 seconds are required in average to process $\tilde{C}(v)$ defined in Equation (2.5) on a neighborhood $N(C)$ whose average cardinality is 98.

2.5/ Conclusion

Active air flow control using MEMS actuation remains a scientific challenge due to several difficulties to overcome, among which a key obstacle is the availability of a prediction of the flow evolution that is both fast and accurate. We have shown that a multilayer perceptron with higher-order input correlations is able to provide, after a proper training, convenient predictions of the force behind a backward-facing step with 5 MEMS microjet actuators. Using such a neural network to predict the evolution of the air flow, an approach allowing to continuously optimize the flow in order to maximize the force generated on the step has been proposed. It uses a neighborhood structure defined on the set of all possible configurations of actuators to control how the actuators evolve. Only slight changes are allowed during the transition from a configuration to another one. Thus, at every instant t we search for the best neighboring configuration. The experiments performed showed that with this approach the cumulative force obtained during a period of time is more appropriate.

Let us emphasize that this work is rather a proof of concept for active air flow control, since the considered problem is a rough approximation of a real moving vehicle. Moreover, as further investigations are needed to get closer to a potential real implementation producing relevant benefits, active air flow control is still a hot topic which is receiving an increasing attention. The scalability is particularly a key issue to be overcome, because in real applications, tens, or even hundreds of MEMS actuators would be used, and thus a more large number of actuators has to be controlled.

3

Chaotic Neural Networks and Predictability of Chaotic Dynamics

Obviously, the possibility of a deliberate control like the proposed approach in the previous chapter is conditional upon the possibility of prediction. For forced convection problems such as the considered airflow one, CFD tools can provide accurate predictions in terms of fluid velocity and pressures. Indeed, even if the airflow is turbulent, having a Reynolds number above 350 K, and thus characterized by chaotic changes in pressure and flow velocity, CFD allows to extract time-averaged and large-scale quantities useful for engineering purposes. In that case CFD can simulate accurately the real world, although the computational cost might be prohibitive according to the geometry and physics.

The inner chaotic behavior of the airflow evolution could be overcome, but this may not be always possible. Furthermore, various natural processes exhibit a chaotic behavior and deterministic chaos is sometimes demanded in some applications. Consequently, the study of nonlinear dynamical systems, and particularly chaotic ones that have a very sensitive dependence on initial conditions, has been the subject of many research works. The design of chaotic neural networks, which are such systems, has received a great deal of attention in various fields of application. However, the resulting neural networks are usually claimed to be chaotic without any rigorous mathematical proof.

To fill this lack, the chaotic iterations scheme devised by Christophe Guyeux during its PhD thesis [36], a mathematical theoretical framework satisfying the Devaney's [37] definition of chaos, has been applied to the modelling of chaotic neural networks. More precisely, we have cooperated with Christophe Guyeux, Jacques Bahi, and Jean-François Couchot on this topic, to establish an equivalence between chaotic iterations in the sense of Devaney and a class of recurrent multilayer perceptrons. In [38] we have proposed an approach for building a global recurrent neural network whose iterations are chaotic. This work has been deepened in [39] by adding a method to check whether some particular neural networks are chaotic or not and an investigation of the predictability of such dynamical systems using classical feedforward multilayer perceptrons.

In particular, in [40], one of the first work on the bioinformatics topic in the AND team, it has been proved that the protein folding in the 2D Hydrophobic-Hydrophilic (HP) square lattice model is chaotic according to Devaney by modelling the conformation process with chaotic iterations. This process has thus a huge sensitivity to initial conditions and it seems very difficult, in this 2D model, to predict with a good accuracy the structure of a protein by exploiting the knowledge of solved structures from similar proteins. Let us recall that the 2D HP square lattice model is a popular model with low resolution that focuses only on the hydrophobicity, by separating the amino acids into two sets: hydrophobic (H) and hydrophilic (or polar P) [41], and which has been used several times for protein folding prediction. The experiments presented in [40], using multilayer perceptrons to fulfill the prediction task, give results in accordance with the above statement. They show that predicting the structure of the protein molecule given only the amino acid sequence as input is difficult. Note that the chaotic behavior of the 2D model means that good predictions might be illusory, but not that the underlying biological process is itself a chaotic process.

This chapter sums up the results obtained in the different papers. After an introduction discussing some related works in Section 3.1 and a presentation of the basics of chaotic iterations and Devaney's chaos in the next one, Section 3.3 formally describes how to build a neural network that operates chaotically. Section 3.4 deals with the dual case of checking whether an existing neural network is chaotic or not, while the predictability of protein folding in the 2D HP square lattice model using multilayer perceptrons is discussed in Section 3.5. In this latter, the first subsection shows that the 2D model is a chaotic dynamical system as defined by Devaney, while the second one focuses on the structure prediction with MLP networks of very simple proteins.

3.1/ Introduction

Since a while neuroscientists discuss the existence of chaos in the brain. In the context of artificial neural networks, this interest has given raise to various works studying the modeling of chaos in neurons. The chaotic neuron model designed by Aihara *et al.* [42] is particularly used to build chaotic neural networks. For example, in [43] is proposed a feedback neural network architecture which consists of two layers (apart from the input layer) with one of them composed of chaotic neurons. In their experiments, the authors showed that without any input sequence the activation of each chaotic neuron results in a positive average Lyapunov exponent, which means a true chaotic behavior. When an input sequence is given iteratively to the network, the chaotic neurons reach stabilized periodic orbits with different periods, and thus potentially provide a recognition state. Similarly, the same authors have introduced another model of chaotic neuron: the nonlinear dynamic state (NDS) neuron, and used it to build a neural network which is able to recognize learned stabilized periodic orbits identifying patterns [44].

Another field of research in which chaotic neural networks have received a lot of attention is data security. In fact, chaotic cryptosystems are an appealing alternative to classical ones due to properties such as sensitivity to initial conditions or topological transitivity. Thus chaotic neural networks have been considered to build ciphering methods, hash functions, digital watermarking schemes, pseudorandom number generators, etc. In [45] such a cipher scheme based on the dynamics of Chua's circuit is proposed. More precisely, a feedforward MLP with two hidden layers is built to learn about 1,500 input-output vector pairs, where each pair is obtained from the three nonlinear ordinary differential equations modeling the circuit. Hence, the proposed chaotic neural network is a network trained to learn a true chaotic physical system. In the cipher scheme the neural network plays the role of chaos generator with which the plain-text will be merged. Untrained neural networks have also been considered to define block ciphering [46] or hash functions [47]. Sometimes, such as in [47], a neural network is claimed to be chaotic simply because it is built considering activation functions and initial conditions that are both chaotic.

Unlike all these previous works, in this chapter a theoretical framework based on Devaney's definition of chaos is introduced to show how to build a recurrent neural network that is equivalent to a chaotic map.

3.2/ Chaotic iterations in the sense of Devaney

In the sequel S^n denotes the n–th term of a sequence S and E_i denotes the i–th component of a vector E. $f^k = f \circ ... \circ f$ is for the k–th composition of a function f. Finally, the following notation is used: $[\![1;N]\!] = \{1, 2, \dots, N\}$.

3.2.1/ Asynchronous Iterations

Let us consider a *system* with a finite number $\mathsf{N} \in \mathbb{N}^*$ of elements (or *cells*), so that each cell has a boolean *state*. A sequence of length N of boolean states of the cells corresponds to a particular *state of the system*. A sequence whose elements belong to $[\![1;\mathsf{N}]\!]$ is called a *strategy*. The set of all strategies is denoted by $\mathbb{S}$.

Definition 1: Chaotic iterations

The set $\mathbb{B}$ denoting $\{0, 1\}$, let $f : \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$ be a function and $S \in \mathbb{S}$ be a strategy. The so-called *chaotic iterations* are defined by $x^0 \in \mathbb{B}^{\mathsf{N}}$ and

$$\forall n \in \mathbb{N}^*, \forall i \in [\![1;\mathsf{N}]\!], x_i^n = \begin{cases} x_i^{n-1} & \text{if } S^n \neq i \\ \left(f(x^{n-1})\right)_{S^n} & \text{if } S^n = i. \end{cases}$$

In other words, at the n–th iteration, only the S^n-th cell is "iterated ". Note that in a more general formulation, S^n can be a subset of components and $\left(f(x^{n-1})\right)_{S^n}$ can be replaced by $\left(f(x^k)\right)_{S^n}$, where $k < n - 1$, describing for example, delays transmission [48].

3.2.2/ Devaney's chaotic dynamical systems

Consider a topological space $(\mathcal{X}, \tau)$ and a continuous function f on $\mathcal{X}$.

Definition 2: Topological transitivity

f is said to be *topologically transitive* if, for any pair of open sets $U, V \subset \mathcal{X}$, there exists $k > 0$ such that $f^k(U) \cap V \neq \varnothing$.

Definition 3: Periodic element (point)

An element (a point) x is a *periodic element* (point) for f of period $n \in \mathbb{N}^*$, if $f^n(x) = x$.

Definition 4: Regularity

f is said to be *regular* on $(\mathcal{X}, \tau)$ if the set of periodic points for f is dense in $\mathcal{X}$: for any point x in $\mathcal{X}$, any neighborhood of x contains at least one periodic point.

Definition 5: Chaotic function

f is said to be *chaotic* on $(\mathcal{X}, \tau)$ if f is regular and topologically transitive.

The chaos property is strongly linked to the notion of "sensitivity", defined on a metric space $(\mathcal{X}, d)$ by:

Definition 6: Sensitivity

f has *sensitive dependence on initial conditions* if there exists $\delta > 0$ such that, for any $x \in \mathcal{X}$ and any neighborhood V of x, there exists $y \in V$ and $n \geqslant 0$ such that $d\left(f^n(x), f^n(y)\right) > \delta$.
δ is called the *constant of sensitivity* of f.

Indeed, Banks *et al.* have proved in [49] that when f is chaotic and $(\mathcal{X}, d)$ is a metric space, then f has the property of sensitive dependence on initial conditions (this property was formerly an element of the definition of chaos). To sum up, quoting Devaney in [37], a chaotic dynamical system "is unpredictable because of the sensitive dependence on initial conditions. It cannot be broken down or simplified into two subsystems which do not interact because of topological transitivity. And in the midst of this random behavior, we nevertheless have an element of regularity". Fundamentally different behaviors are consequently possible and occur in an unpredictable way.

3.2.3/ Linking asynchronous iterations and Devaney's chaos

In this section we give outline proofs of the properties on which the study of chaotic neural networks is based. The complete theoretical framework is detailed in [50].

Denote by Δ the *discrete boolean metric*, $\Delta(x, y) = 0 \Leftrightarrow x = y$. Given a function $f : \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$, define the function $F_f : [\![1;\mathsf{N}]\!] \times \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$ such that

$$F_f(k, E) = \left(E_j.\Delta(k, j) + f(E)_k.\overline{\Delta(k, j)}\right)_{j\in[\![1;\mathsf{N}]\!]}, \tag{3.1}$$

where + and . are the boolean addition and product operations, $\overline{x}$ is for the negation of x.

Consider the phase space $\mathcal{X} = [\![1;\mathsf{N}]\!]^{\mathbb{N}} \times \mathbb{B}^{\mathsf{N}}$ and the map

$$G_f(S, E) = \left(\sigma(S), F_f(i(S), E)\right) \tag{3.2}$$

where $\sigma : (S^n)_{n\in\mathbb{N}} \in \mathbb{S} \mapsto (S^{n+1})_{n\in\mathbb{N}} \in \mathbb{S}$ is the *shift* function, and the *initial function* i is the map which associates to a sequence its first term: $i : (S^n)_{n\in\mathbb{N}} \in \mathbb{S} \mapsto S^0 \in [\![1;\mathsf{N}]\!]$.

Thus chaotic iterations can be described by the following iterations [50]:

$$\left\{ \begin{array}{l} X^0 \in \mathcal{X} \\ X^{k+1} = G_f(X^k). \end{array} \right. \tag{3.3}$$

Let us define a new distance between two points $(S, E), (\check{S}, \check{E}) \in \mathcal{X}$ by:

$$d((S, E); (\check{S}, \check{E})) = d_e(E, \check{E}) + d_s(S, \check{S}), \tag{3.4}$$

where

- $d_e(E,\check{E}) = \sum_{k=1}^{\mathsf{N}} \Delta(E_k,\check{E}_k) \in [\![0;\mathsf{N}]\!]$,
- $d_s(S,\check{S}) = \frac{9}{\mathsf{N}} \sum_{k=1}^{\infty} \frac{|S^k - \check{S}^k|}{10^k} \in [0;1]$.

This new distance has been introduced in [50] to satisfy the following requirements. When the number of different cells between two systems is increasing, then their distance should increase too. In addition, if two systems present the same cells and their respective strategies start with the same terms, then the distance between these two points must be small because the evolution of the two systems will be the same for a while. The distance presented above follows these recommendations. Indeed, if the floor value $\lfloor d(X,Y) \rfloor$ is equal to n, then the systems $E, \check{E}$ differ in n cells. In addition, $d(X,Y) - \lfloor d(X,Y) \rfloor$ is a measure of the differences between strategies S and $\check{S}$. More precisely, this floating part is less than 10^{-k} if and only if the first k terms of the two strategies are equal. Moreover, if the k−th digit is nonzero, then the k−th terms of the two strategies are different.

It is proven in [50], by using the sequential continuity, that the vectorial negation $f_0(x_1,\ldots,x_\mathsf{N}) = (\overline{x_1},\ldots,\overline{x_\mathsf{N}})$ satisfies the following proposition:

Proposition 1:

G_{f_0} is a continuous function on $(\mathcal{X},d)$.

It is then checked, also in [50], that in the metric space $(\mathcal{X},d)$, the vectorial negation fulfills the three conditions for Devaney's chaos: regularity, transitivity, and sensitivity. This has led to the following result.

Proposition 2:

G_{f_0} is a chaotic map on $(\mathcal{X},d)$ in the sense of Devaney.

3.3/ Chaotic neural networks according to Devaney

3.3.1/ Neural network design

Let us consider the vectorial negation function denoted by $f_0 : \mathbb{B}^\mathsf{N} \to \mathbb{B}^\mathsf{N}$ and its associated map $F_{f_0} : [\![1;\mathsf{N}]\!] \times \mathbb{B}^\mathsf{N} \to \mathbb{B}^\mathsf{N}$. First, it is possible to define a MLP which recognizes F_{f_0}. That means, for all $(k,x) \in [\![1;\mathsf{N}]\!] \times \mathbb{B}^\mathsf{N}$, the response of the output layer to the input (k,x) is $F_{f_0}(k,x)$. Second, the output layer can be connected to the input layer as it is depicted in Figure 3.1, leading to a global recurrent neural network working as follows:

- At the initialization stage, the neural network receives a boolean vector $x^0 \in \mathbb{B}^\mathsf{N}$ as input state, and $S^0 \in [\![1;\mathsf{N}]\!]$ in its input integer channel $i()$. Thus, a new boolean vector $x^1 = F_{f_0}(S^0,x^0) \in \mathbb{B}^\mathsf{N}$, representing the new state of the dynamical system, is computed by the neural network.

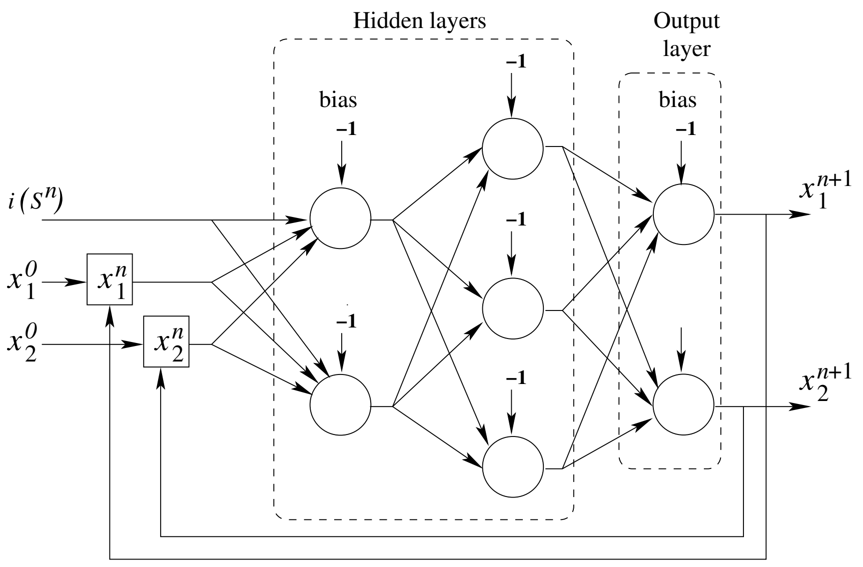


Figure 3.1: A globally recurrent neural network equivalent to chaotic iterations by modeling function F_{f_0} such that $x^{n+1} = \left(x_1^{n+1}, x_2^{n+1}\right) = F_{f_0}\left(i(S^n), \left(x_1^n, x_2^n\right)\right)$.

- The state x^1 is published as an output and sent back to the input layer, to act as boolean state in the next iteration.
- At iteration number $n+1$, the recurrent neural network receives the state $x^n \in \mathbb{B}^\mathsf{N}$ from its output layer and $i(S^n) \in [\![1;\mathsf{N}]\!]$ from its input integer channel $i()$. It can thus compute $x^{n+1} = F_{f_0}(i(S^n), x^n) \in \mathbb{B}^\mathsf{N}$, which will be the new output of the network.

If the initial state $x^0 \in \mathbb{B}^\mathsf{N}$ is sent to the network with a sequence $S \in [\![1;\mathsf{N}]\!]^\mathbb{N}$ applied in the input integer channel $i()$, then the sequence $(x^n)_{n\in\mathbb{N}^*}$ of the outputs is exactly the same than the sequence obtained from the following chaotic iterations: $x^0 \in \mathbb{B}^\mathsf{N}$ and

$$\forall n \in \mathbb{N}^*, \forall i \in [\![1;\mathsf{N}]\!], x_i^n = \begin{cases} x_i^{n-1} & \text{if } S^n \neq i \\ \left(f_0(x^{n-1})\right)_{S^n} & \text{if } S^n = i. \end{cases} \tag{3.5}$$

From a mathematical viewpoint, the MLP with global recurrent connections defined in this subsection and chaotic iterations recalled above have the same behavior. In particular, given the same input vector $\left((S^n)_{n\in\mathbb{N}}, x^0\right)$, they produce the same output vector $(x^n)_{n\in\mathbb{N}^*}$: they are two equivalent reformulations of the iterations of G_{f_0} in $\mathcal{X}$. As a consequence, the behavior of the proposed recurrent network faithfully reflects the behavior of G_{f_0} which is chaotic according to Devaney.

The approach proposed to build chaotic neural networks is not restricted to an ad hoc function $f_0 : \mathbb{B}^\mathsf{N} \to \mathbb{B}^\mathsf{N}$, it can be generalized. Indeed, the function F_{f_0} can be replaced by any function $F_f : [\![1;\mathsf{N}]\!] \times \mathbb{B}^\mathsf{N} \to \mathbb{B}^\mathsf{N}$ associated to a function f such that its chaotic map G_f is chaotic, as defined by Devaney. Alternative functions f for f_0 can be easily identified using the following theorem established by Christophe Guyeux during its PhD thesis:

Theorem 1:

Functions $f : \mathbb{B}^n \to \mathbb{B}^n$ such that G_f is chaotic according to Devaney, are functions such that the oriented graph of iterations $\Gamma(f)$, see Definition 7, is strongly connected.

Definition 7: Graph of iterations

In the oriented graph of iterations $\Gamma(f)$, vertices are boolean vectors of $\mathbb{B}^{\mathsf{N}}$, and there is an arc labeled i from x to x' if and only if $x' = F_f(i, x)$.

Since it is easy to check whether a graph is strongly connected, thanks to this theorem we can discover new functions $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ such that the neural network associated to G_f behaves chaotically, as defined by Devaney. In [38], as illustrative examples, we have given five functions among which three led to neural networks which behave chaotically. An experimental investigation of the ability to learn two functions whose graph of iterations are strongly connected and one for which it is not the case, considering neural networks built as explained previously, is presented thereafter.

Let us introduce the two following functions:

$$f_1(x_1, x_2, x_3) = (\overline{x_1}, \overline{x_2}, \overline{x_3}) \text{ and } f_2(x_1, x_2, x_3) = (\overline{x_1}, x_1, x_2).$$

It can easily be checked that these functions satisfy the hypothesis of Theorem 1, thus their iterations are chaotic according to Devaney. Then when the neural network architecture defined previously learns to recognize F_{f_1} or F_{f_2}, it tries to learn these chaotic iterations, that is, a chaotic behavior as defined by Devaney. On the contrary, the function

$$g(x_1, x_2, x_3) = (\overline{x_1}, x_2, x_3)$$

is such that $\Gamma(g)$ is not strongly connected. In this case, due to Theorem 1, the neural network does not learn a chaotic process. We will now recall the study presented in [38] of the training process of functions F_{f_1}, F_{f_2}, and F_g, that is to say, the ability to learn one iteration of chaotic iterations.

3.3.2/ Experimental results

For each recurrent neural network we have considered architectures with one and two hidden layers, with in the first case different numbers of hidden neurons. Thus we will have different networks modeling the same iteration function. Only the size and number of hidden layers may change, since the numbers of inputs and output neurons are fully specified by the function. Each network has sigmoidal hidden neurons and linear output neurons, and is trained like the MLPs considered in Chapters 1 and 2. The training is performed until the learning error is lower than a chosen threshold value (10^{-2}).

Table 3.1 gives for each considered neural network the mean number of epochs needed to learn one iteration, and a success rate that reflects a successful training in less than 1000 epochs. Both values are computed considering 25 trainings with random weights and biases initialization. These results highlight several points. First, the two hidden layer structure seems to be quite inadequate to learn chaotic behaviors. Second, training networks so that they behave chaotically seems to be difficult for these simplistic functions only iterated once, since they need in average more epochs to be correctly trained. However, the correctness of this point needs to be further investigated by using larger sets of iteration functions.

Table 3.1: Results of some iteration functions learning using different globally recurrent neural networks.

Topology	One hidden layer			
	8 neurons		10 neurons	
Function	Mean epoch	Success rate	Mean epoch	Success rate
f_1	82.21	100%	73.44	100%
f_2	76.88	100%	59.84	100%
g	36.24	100%	37.04	100%

Topology	Two hidden layers: 8 and 4 neurons	
	Mean epoch number	Success rate
f_1	203.68	76%
f_2	135.54	96%
g	72.56	100%

3.4/ How to determine whether a neural network is chaotic or not

Assume now that a neural network is already available and for which we want to determine whether it is chaotic or not. Being able to perform such checking might be of great interest to researchers when their work rely on neural networks said to be chaotic without any rigorous mathematical proof. In the following we describe an approach for a particular category of MLP, introduced in [39], but we think that this approach can be extended to a large variety of neural networks.

The considered neural networks are multilayer perceptrons having an architecture with two key characteristics. First, the input layer consists of l binary digits and one integer value, while the output layer is a binary vector of l components. Second, each binary output is connected to an input one such that both have the same index value. Moreover, these recurrent neural networks are supposed to evolve in discrete time.

- To start such network evolution, the input layer is seeded with l bits denoted $(x_1^0, \dots, x_l^0)$ and an integer value S^0 from $[\![1;\mathsf{N}]\!]$.
- At iteration n, the last output vector $(x_1^n, \dots, x_l^n)$ becomes the input vector used to compute the new output one $(x_1^{n+1}, \dots, x_l^{n+1})$, given a new input integer $S^n \in [\![1;\mathsf{N}]\!]$.

Neural networks having both architecture and evolution rule as defined above can then be proven to be chaotic based on the following considerations. First, denote by $F : [\![1;l]\!] \times \mathbb{B}^l \to \mathbb{B}^l$ the function which maps an input vector $(s,x) = (s,(x_1,\dots,x_l)) \in [\![1;l]\!] \times \mathbb{B}^l$ into a vector $y = (y_1,\dots,y_l)$, where y is produced by the output layer of a neural network receiving (s,x) as input vector. Second, let $f : \mathbb{B}^l \to \mathbb{B}^l$ be the function defined by:

$$f(x_1,\dots,x_l) = (F(1,(x_1,\dots,x_l)),\dots,F(l,(x_1,\dots,x_l))). \tag{3.6}$$

Thus, $\forall j, 1 \leq j \leq l$ we have $(f(x_1,\dots,x_l))_j = F(j,(x_1,\dots,x_l))$. By seeding the neural network with $(x_1^0,\dots,x_l^0)$ and S from $[\![1;l]\!]^{\mathbb{N}}$, it will produce exactly the same outputs than the chaotic iterations given by Equation (3.3) with initial configuration $X^0 = (S,x^0) \in \mathcal{X}$ where $\mathcal{X} = [\![1;\mathsf{l}]\!]^{\mathbb{N}} \times \mathbb{B}^l$. That means that the iterations defined by G_f model formally the evolution of a neural network which is a chaotic iterations-based recurrent multilayer perceptron. Hence, according to Theorem 1, to check whether such a neural network behaves chaotically consists simply to determine if the correspondig graph of iterations $\Gamma(f)$ is strongly connected or not. Topological properties of chaotic neural networks, like their convergence, have also been investigated in [39].

3.5/ Predictability of chaotic dynamics: an example with protein folding

We have studied the suitability of feedforward multilayer perceptrons for predicting chaotic and non-chaotic iterations in [39], motivated by the context of steganalysis techniques. Indeed, these techniques, whose purpose is to automatically detect the presence of hidden messages inside cover media, can rely on detectors like support vector machines, neural networks, and Markov chains. Therefore, as to the best of our knowledge none of the embedding schemes used to hide a message that have been steganalyzed are chaotic according to Devaney's definition of chaos, we wondered whether usual detectors and more particularly neural networks are still able to give good results when dealing with truly chaotic embedding schemes.

In this section, we will rather focus on protein folding which is one of the most interesting challenges in computational biology. To study this folding, tools like neural networks and genetic algorithms have received a lot of attention, mainly due to the NP completeness of the folding process. The background idea that has given rise to the use of these algorithms is obviously that the folding process is predictable. However, this important assumption is disputable as chaotic properties of such a process have been highlighted [51, 52]. For instance, in [51] the Lyapunov exponent of a folding process has been experimentally computed, resulting in a positive exponent. In [40], we have evaluated the topological behavior of a well-known dynamical system used for protein folding prediction and the subsections thereafter present the following points of this evaluation: the first subsection establishes mathematically that the 2D model is a chaotic dynamical system as defined by Devaney, while the second discusses structure prediction of very simple proteins with MLP networks.

3.5.1/ Folding process in 2D model is chaotic

3.5.1.1/ 2D hydrophobic-hydrophylic (HP) model

- HP model

  In the HP model, hydrophobic interactions are supposed to dominate protein folding. This model was formerly introduced by Dill, who considered in [41] that the protein core freeing up energy is formed by hydrophobic amino acids, whereas hydrophilic amino acids tend to move in the outer surface due to their affinity with the solvent (see Figure 3.2). In this model, a protein conformation is a "self-avoiding walk (SAW)" on a 2D or 3D lattice such that its energy E, depending on topological neighboring contacts between hydrophobic amino acids that are not contiguous in the primary structure, is minimal. In other words, for an amino-acid sequence P of

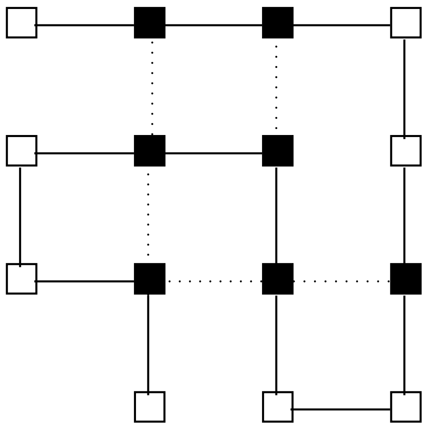


Figure 3.2: Hydrophilic-hydrophobic model (black squares are hydrophobic residues).

length N and for the set $\mathcal{C}(P)$ of all SAW conformations of P, the chosen conformation will be $C^* = min\{E(C)/C \in \mathcal{C}(P)\}$ [53]. In that context and for a conformation C, $E(C) = -q$ where q is equal to the number of topological hydrophobic neighbors. For example, $E(c) = -5$ in Figure 3.2.

- Protein encoding

  Additionally to the direct coordinate presentation, at least two other isomorphic encoding strategies for HP models are possible: relative encoding and absolute encoding. In relative encoding [54], the move direction is defined relative to the direction of the previous move. Alternatively, in absolute encoding [55], which is the encoding chosen in this work, the direct coordinate presentation is replaced by letters or numbers representing directions with respect to the lattice structure.

  For absolute encoding in the 2D square lattice, the permitted moves are: forward $\rightarrow$ (denoted by 0), down $\downarrow$ (1), backward $\leftarrow$ (2), and up $\uparrow$ (3). A 2D conformation C of $\mathsf{N}+1$ residues for a protein P is then an element C of $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$, with a first component equal to 0 (forward) [54]. For instance, in Figure 3.2, the 2D absolute encoding is 00011123322101 (starting from the upper left corner). In that situation, at most 4^{N} conformations are possible when considering $\mathsf{N}+1$ residues, even if some of them are invalid due to the SAW requirement.

3.5.1.2/ Protein folding as chaotic iterations

- Initial premises

  The primary structure of a given protein P with $\mathsf{N}+1$ residues is coded by $00\ldots0$ (N times) in absolute encoding. Its final 2D conformation has an absolute encoding equal to $0C_1^*\ldots C_{\mathsf{N}-1}^*$, where $\forall i, C_i^* \in \mathbb{Z}/4\mathbb{Z}$, is such that $E(C^*) = min\{E(C)/C \in \mathcal{C}(P)\}$. This final conformation depends on the repartition of hydrophilic and hydrophobic amino acids in the initial sequence.

  Moreover, we suppose that if the residue number $n+1$ is forward the residue number n in absolute encoding ($\rightarrow$) and if a fold occurs after n, then the forward move can only by changed into up ($\uparrow$) or down ($\downarrow$). That means, in our simplistic model, only rotations of $+\frac{\pi}{2}$ or $-\frac{\pi}{2}$ are possible.

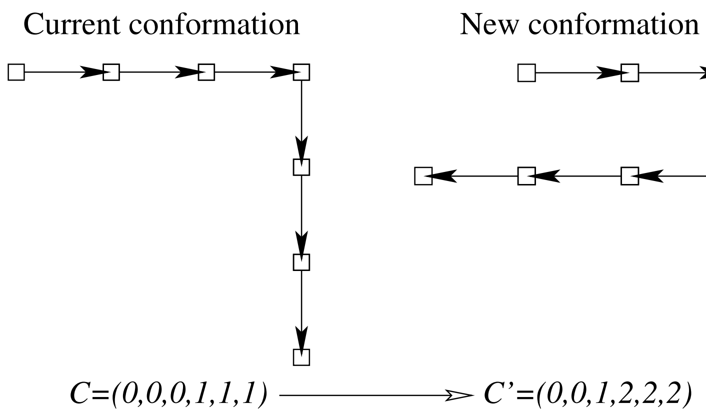


Figure 3.3: Example of folding process.

Consequently, for a given residue that is supposed to be updated, only one of the two possibilities below can appear for its absolute move during a fold:

- $0 \longmapsto 1, 1 \longmapsto 2, 2 \longmapsto 3$, or $3 \longmapsto 0$ for a fold in the clockwise direction, or
- $1 \longmapsto 0, 2 \longmapsto 1, 3 \longmapsto 2$, or $0 \longmapsto 3$ for an anticlockwise.

This fact leads to the following definition:

Definition 8:

The clockwise fold function is the function $f : \mathbb{Z}/4\mathbb{Z} \longrightarrow \mathbb{Z}/4\mathbb{Z}$ defined by $f(x) = x + 1 (\text{mod } 4)$.

Obviously the anticlockwise fold function is $f^{-1}(x) = x - 1 (\text{mod } 4)$.

Thus at the n–th folding time, a residue k is chosen and its absolute move is changed by using either f or f^{-1}. As a consequence, all of the absolute moves must be updated from the coordinate k until the last one N by using the same folding function. For example, if the current conformation is:

$$C = (C_1, \ldots, C_6) = (0, 0, 0, 1, 1, 1)$$

and the third residue is chosen to fold by a rotation of $-\frac{\pi}{2}$ (mapping f, also referred to as $+$), in which case the fold F is denoted by $+3$, the new conformation will be:

$$C' = \left(C'_1, \ldots, C'_6\right) = (C_1, C_2, f(C_3), f(C_4), f(C_5), f(C_6)) = (0, 0, 1, 2, 2, 2).$$

This folding process example is illustrated graphically by Figure 3.3.

- A formulation as chaotic iterations

  Modelling protein folding using chaotic iterations can be justified as follows. At each iteration, the same process is applied to the system (*i.e.* to the conformation), which is the folding operation. Additionally, it is not a necessity that all of the residues fold at each iteration: indeed it is possible that, at a given iteration, only some of these residues folds. Such iterations, where not all the cells of the considered system are to be updated, are exactly the iterations modeled by chaotic iterations in Section 3.2.

Indeed, the protein folding process with folding sequence $(F^n)_{n\in\mathbb{N}}$ consists in the following chaotic iterations: $C^0 = (0,0,\ldots,0)$ and,

$$C^{n+1}_{|i|} = \begin{cases} C^n_{|i|} & \text{if } i \notin S^n \\ f^{sign(i)}(C^n)_{|i|} & \text{else} \end{cases}, \tag{3.7}$$

where the chaotic strategy $S^n, \forall n \in \mathbb{N}$, is defined by $S^n = [\![-\mathsf{N};\mathsf{N}]\!] \setminus [\![-F^n;F^n]\!]$.

Thus, to prove that the protein folding process is chaotic as defined by Devaney, is equivalent to prove that the graph of iterations of the chaotic iterations defined above is strongly connected. This last fact is obvious, as it is always possible to find a folding process that maps any conformation $(C_1,\ldots,C_\mathsf{N}) \in \mathfrak{C}_\mathsf{N}$ to any other $(C'_1,\ldots,C'_\mathsf{N}) \in \mathfrak{C}_\mathsf{N}$ (this is Lemma 1 in [40]).

$\mathfrak{C}_\mathsf{N}$ is the subset of conformations that are obtained starting from the initial conformation $(0,0,\ldots,0)$ after N folding operations and such that all intermediate conformations satisfy the SAW requirement. This SAW folding process requirement is the most usual meaning of “SAW requirement” in the literature (it is used, for instance, in [52, 56, 57, 58]).

Let us finally remark that it is easy to study processes such that more than one fold occur per time unit by using chaotic iterations.

3.5.2/ Can a neural network predict a future protein conformation?

In this set of experiments, multilayer perceptrons are used to learn the conformation of very simple proteins (peptides). In fact, we consider proteins composed of five residues, of which only 4 can change since the first one is always 0, and folding dynamics of two or three folds. For example, if the current protein conformation is $(0)1222$, and folds $+4$ and -1 are successively applied, then the new conformation will be $(0)0112$. Obviously, these choices, that lead respectively to 20736 and 186624 potential conformations, do not correspond to realistic folding processes. However, they allow to evaluate the ability of neural networks to learn very simple conformations.

The networks consist of MLP with 3 or 4 inputs, the current conformation without the first residue and a sequence of 2 or 3 successive folds, that produce a single output: the resulting conformation. Moreover, like in the previous chapter (more precisely as in Subsection 2.3.1.1), we changed the classical MLP input layer structure for the HPU (Higher-order Processing Unit) one and we trained such network with a holdout validation through a random sampling strategy, thanks to the L-BFGS optimization algorithm. Similarly, to estimate the prediction accuracy, we considered the two same statistical measures than in Chapter 2, namely the CVRMSE and the coefficient of determination R^2 defined, respectively, by Equations (2.1) and (2.2). Let us notice that in [40] we replaced R^2 by an alternative equivalent measure called coefficient of efficiency (E) based on the Average Relative Variance (ARV) [59] ($R^2 \simeq \mathrm{E} = 1 - \mathrm{ARV}$).

The considered neural networks differ in the size of a single hidden layer and are trained until a maximum number of epochs is reached. As we set the order of the HPU structure to 3, instead of 3 and 4 initial inputs we have 19 and 34 inputs. We train neural networks of 15 and 25 hidden neurons, using as maximum number of epochs a value in $\{500, 1000, 2500\}$, whatever the number of initial inputs (see Table 3.2). The learning, validation, and test subsets are built in such a way that they respectively represent 65%, 10%,

Table 3.2: Results of the validation of networks with an HPU structure of order 3 for several numbers of hidden neurons.

Topology	3 initial / 19 HPU inputs and 1 output		
Hidden neurons	Epochs	%CVRMSE	R^2
	Data set of 5184 samples		
15 neurons	500	24.97	0.6176
	1000	20.67	0.7372
	2500	16.69	0.8269
25 neurons	500	23.33	0.6627
	1000	15.94	0.8435
	2500	10.75	0.9285
	Data set of 10368 samples		
15 neurons	500	26.27	0.5777
	1000	22.08	0.7000
	2500	18.81	0.7775
25 neurons	500	24.54	0.6315
	1000	16.11	0.8409
	2500	9.43	0.9440
	Data set of 15552 samples		
15 neurons	500	24.74	0.6249
	1000	19.92	0.7556
	2500	16.35	0.8341
25 neurons	500	22.90	0.6753
	1000	15.42	0.8533
	2500	8.89	0.9499
Topology	4 initial / 34 HPU inputs and 1 output		
	Data set of 46656 samples		
15 neurons	500	35.27	0.2394
	1000	33.50	0.3136
	2500	31.94	0.3741
25 neurons	500	35.05	0.2465
	1000	32.25	0.3615
	2500	28.61	0.4956

and 25% of the whole data set. In the case of 3 initial inputs data sets of $5,184$, $10,368$, and $15,552$ samples are used, they represent 25%, 50%, and 75% of the $20,736$ potential conformations. For the 4 initial outputs case we restrict our experiments to a data set of $46,656$ samples, that corresponds to 25% of the $186,624$ potential conformations.

In Table 3.2 we give, for the different learning setups and data set sizes, the mean values of CVMSE and R^2 for the output. To compute these values, 10 trainings with random subsets construction and network parameters initialization were performed. It can be seen that in all cases the best performances are obtained for the largest networks (25 hidden neurons) that are the most trained (2500 epochs). Furthermore, a larger data set allows only a slight increase of the prediction quality. We observe for a three time increase of the data set: from $5,184$ to $15,552$ samples, a very small improvement of 1.66% for R^2.

At a first glance, the prediction accuracy seems not too bad for the 3 initial inputs topology, with coefficients R^2 above 0.9. However, remember that we try to predict very simple conformations far away from the realistic ones. Furthermore, a look at the results obtained for the second topology, the one with 4 initial outputs, shows that predicting a conformation that undergoes only one more folding transition is intractable with the considered learning setups: R^2 is always below 0.5. Clearly, the different MLP have failed at predicting the protein folding dynamics. A larger neural network with a longer training process may be able to improve these results. But finding a learning setup well suited for the prediction of relevant proteins structures that are far more complex seems very hypothetical.

Finally, let us notice that the HPU structure has a major impact on the learning quality. Indeed, let us consider the coefficient R^2 obtained for the data set of $5,184$ samples and a network composed of 25 hidden neurons trained during 2500 epochs. As shown in Table 3.2 the coefficient R^2 is about 0.9285 if the neural network has an HPU structure of order 3, whereas experiments made without increasing the number of initial inputs give 0.6930 as mean value for R^2. Similar experiments in case of the second topology result in $R^2 = 0.2154$ for the classical structure that has no HPU structure. That represents a respective decrease of more than 25 and 50%, from which we can say that MLP networks with a classical structure would have given worse predictions.

At this point we can only claim that it is not completely obvious that computational intelligence tools like neural networks are able to predict, with a good accuracy, protein folding. To reinforce this belief, tools optimized for chaotic behaviors must be found – if such tools exist. Let us recall that we did not study the biological folding process, but the protein folding as it is described in the 2D HP square lattice model. Consequently, we only claim that the 2D folding process is chaotic and therefore unsuitable to give relevant predictions.

3.6/ Conclusion

In this chapter we have summed up different works [38, 39, 40] in which we have studied the ability of multilayer perceptron neural networks to deal with chaotic processes. All these works are based on a rigorous theoretical framework that allows to model the considered problems as dynamical systems shown to be chaotic as defined by Devaney. First, we have built a globally recurrent multilayer perceptron architecture that can be trained to learn chaotic iterations and thus constructed artificial neural networks proven to be chaotic. Second, this work has been extended by providing a method to check whether an existing neural network is chaotic or not through the verification of a property on a graph describing the neural network evolution. As this previous point is the dual of the first one, we have thus an equivalence between chaotic iterations and a particular class of recurrent neural networks. Finally, we have investigated the ability of classical feedforward multilayer perceptrons to predict chaotic dynamics considering a real unsolved computational biology problem, namely the protein folding. Indeed, to solve this problem, tools like neural networks have received a lot of attention assuming that the folding process is predictable. However, as we have established that protein folding in the 2D hydrophobic-hydrophylic (HP) square lattice model is chaotic according to Devaney, this assumption is questionable. The learning with feedforward multilayer perceptrons of this natural dynamics tends to show that such chaotic behaviors are more difficult to learn than nonchaotic ones.

II

Solving Problems with Deep Learning and Reservoir Computing

4

Image Steganalysis using a Convolutional Neural Network

The research works presented in the previous part have investigated the use of shallow neural networks, typically limited to one or two hidden layers. In such neural networks the inputs are hand-designed and the network learns only to map given input features to output data. Obviously, the choice of the input features is of great importance and a key reason for the success or the failure of the neural network. To solve this problem of input data representation, over the past years various machine learning methods that learns to automatically extract from input data the suitable features to fulfill the prediction or classification target task have been proposed. Since this family of methods consists of networks with deep architectures having many hidden layers to learn high-level feature representation, it is known as deep learning [60, 61, 62].

Today, for many challenging tasks deep learning has led to breakthrough improvements, becoming the state-of-the-art method in natural language processing, image captioning [63], or benchmark problems like the MNIST problem [64] that consists in the recognition of handwritten digits. Deep learning has taken off because it seems to have the promising potential to solve in a competitive way a wide-range of real-world applications. In fact, in a variety of domains the expected impact of deep learning is such that it has given rise to a sort of gold rush of technology innovation. Numerous deep learning startups have thus emerged, usually to face a specific kind of problem, for example medical image analysis, and all american information technology giant, namely Google, Apple, Facebook, Amazon, Microsoft (the GAFA or GAFAM), and even Tesla or Uber, have devoted large resources to this field of research, trying to attract people which are expert in it.

Various deep architectures have been proposed over the years, leading to a continuous enrichment, encompassing feedforward and recurrent networks. We can note Deep Belief Networks [65], Convolutional Neural Networks (CNN) [66, 67, 68], and even to some extent the so-called Reservoir Computing [69, 70] studied in the final chapter. In the latter case, this vision is supported by several research works which have proposed deep Reservoir Computing networks [71] obtained by stacking multiple reservoirs. The current amazing success of deep learning comes not only from the exponential advantage of depth compared to shallow networks, but also from the increasing availability of huge amounts of training data and powerful computing platform, and more particularly from the parallel computing ability of General-Purpose computing on Graphics Processing Units (GPGPU).

This chapter is devoted to research works on the use of Convolutional Neural Networks to perform image steganalysis in spatial domain. It is focused on our main result [72] which has been presented during the *32nd International Conference on ICT Systems Security and Privacy Protection* - IFIP SEC 2017 in May 2017. This result is a criterion to choose the appropriate steganalyzer between a CNN-based one and a conventional steganalysis method, considering the state-of-the-art approach in each family. To the best of our knowledge, at the time we presented this approach, we had obtained the best detection performance. We start with an introduction to image steganography and steganalysis in Section 4.1, followed by a brief description of the steganographic algorithms we consider and a review on steganalysis methods, respectively in Sections 4.2 and 4.3. In Section 4.3 we also introduce our first proposal, a shallow CNN using large convolution filters only suitable for the case where the embedding process is done with the same stego key. We continue with a study of the most competitive CNN-based steganalyzer proposed by Xu *et al.* [73] in Section 4.4, on which is based our main result described in Section 4.5.

4.1/ Introduction

Steganography is the art and science of covert communications, whose goal is to embed a secret message inside an innocent cover media transmitted through a public communication channel so that the message can be extracted at destination. In this work we focus on digital image as cover media since in our digital world images are easily produced and diffused. The security of such a steganographic scheme, which is used to embed and extract the secret message, relies on its ability to perform changes in the cover as less detectable as possible. Therefore, the security of a steganographic tool is assessed through its resistance against an adversary that tries to detect the presence of an embedded message. Steganalysis, as adversary of steganography, face this challenge and thus refers to approaches whose purpose is to distinguish malicious stego images from innocent cover ones.

Like for crypto systems, Figure 4.1 shows how the famous prisoners' problem formulated by Simmons [74] can be solved with help of steganography. We can see how two prisoners, Alice and Bob, use this kind of information hidding scheme to communicate together. Alice sends a hidden message to Bob by building a stego image thanks to a chosen steganographic tool, controlling the way the embedding is done with a stego key. After transmission through the unsecured channel, Bob can extract the message, assuming that he knows the steganographic tool and the stego key which were chosen. To decide whether it allows or not the transmission, the warden Eve uses a steganalysis method, also called a steganographic detector or steganalyzer.

There are two categories of embedding schemes in digital image steganography: spatial domain embedding and transform, or frequency, domain embedding. In the first category the embedding is performed by changing some pixel values, whereas in the second category the secret message is hidden in transform coefficients, typically Discrete Cosine Transform (DCT) coefficients or Discrete Wavelet Transform (DWT) ones. Most of these schemes consist in minimizing an additive distortion function defined as the sum of embedding costs computed individually for each pixel. Obviously, the cost value associated to a pixel reflects its detectability after modification: the smaller the cost, the less detectable the embedding will be. In this work, the image domain is the only information supposed to be known about the embedding process.

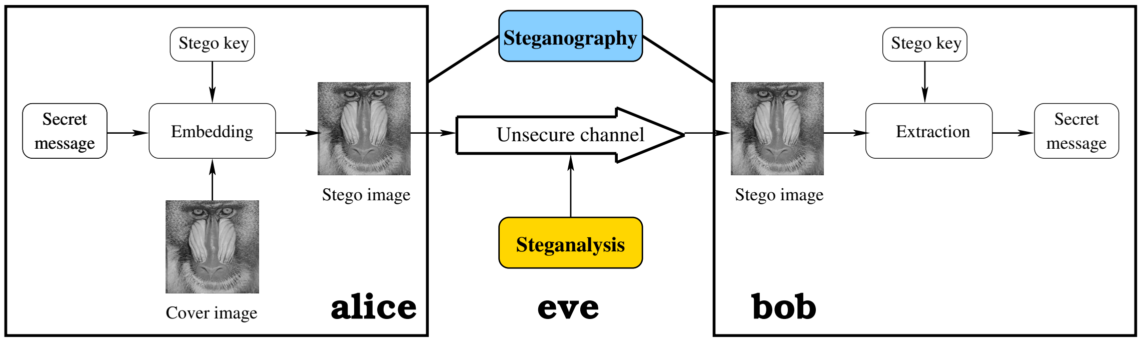


Figure 4.1: Illustration of steganography and its counterpart steganalysis with Simmons' prisoners' problem [74].

Several spatial domain steganography methods have been designed over the past years, going from simple LSB (Least Significant Bit) replacement to complex content-adaptive embedding algorithms, like the older HUGO [75] and WOW [76] schemes, or more recent ones as HILL [77], STABYLO [78] (designed by colleagues of the AND team), and MiPOD [79]. Examples of possible alternatives in frequency domain of JPEG images are nsF5 [80], the Uniform Embedding Distortion (UED) algorithm [81], and its variant UERD [82]. An interesting scheme proposed by Holub *et al.* [83], named UNIWARD, can be applied in both domain, leading respectively to S(patial)-UNIWARD and J(PEG)-UNIWARD.

In the following section we give some details on S-UNIWARD, MiPOD, and HILL, the three most secure spatial domain embedding schemes targeted by our work.

4.2/ Image Steganography in Spatial Domain

What makes the three aforementioned steganographic algorithms different from one another is the distorsion function ρ used to compute the modification cost for each pixel of the cover image. Indeed, once the per-pixel distortions are computed, they all choose the pixels that will embed the message by solving the minimization problem evoked above with the near-optimal Syndrome-Trellis Coding (STC). The selected pixel locations depend also from the stego key since it seeds the PRNG (Pseudo Random Number Generator) used in STC. Let X denote a cover image and $\rho(X)$ be the matrix whose elements represent the modification costs of the pixel values by -1 or +1, then, by ordering the pixels according to their value in $\rho(X)$, a set of pixels minimizing the embedding cost of the message can be computed. The distortion function is of great importance, because it must be defined in order to reflect as precisely as possible the underlying cover image model. Therefore it must return large values in a easy-defined or smooth area, like a blue sky, whereas in textured or "chaotic" areas low values are expected.

In the case of S-UNIWARD [83] and as one can understand from the complete name: UNIversal WAvelet Relative Distortion, the distortion function ρ_U estimates how the change of a pixel value affects a 2D block of wavelet coefficients whose size depends from the 2D wavelet support. Formally ρ_U is defined by:

$$\rho_U(X) = \sum_{k=1}^{3} \frac{1}{|X \star K^k| + \sigma} \star |K^k|^{\curvearrowleft}, \tag{4.1}$$

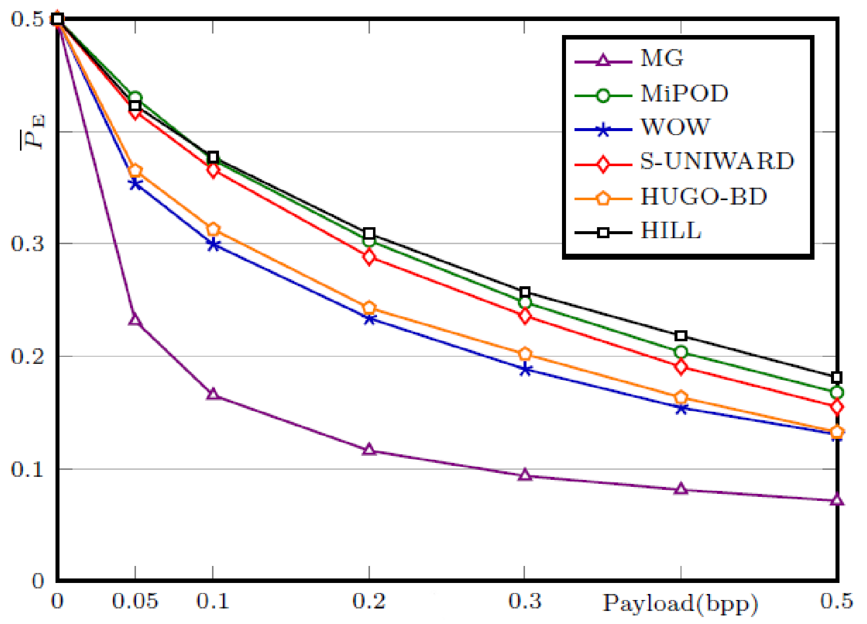


Figure 4.2: Detection error for different embedding schemes when steganalyzing with maxSRMd2.

where $\star$ is a convolution mirror-padded product, $Y^{\curvearrowleft}$ is the result of a 180 degrees rotation of Y, K^k with $1 \leq k \leq 3$ are Daubechies-8 wavelet kernels in the three directions, and σ is a stabilizing constant. It should be noticed that the multiplicative inverse is element-wise applied. An element of $\rho_U(X)$ is small if and only if there are large variations of large cover wavelet coefficients in the three directions, where texture and edges appear. Similar combinations of convolution products with a high-pass filter and two low-pass ones can be found in the definition of the distortion function ρ_H of the HILL steganographic scheme.

The steganographic algorithm MiPOD [79], which means Minimizing the Power of Optimal Detector, is quite different. It is a probabilistic approach based on a locally-estimated multivariate Gaussian cover image model from which a stego image model is derived. More precisely, given the matrix β of change rates β_n, where β_n is the probability that stego pixel y_n is obtained by changing the cover one x_n $(P(y_n = x_n + 1) = P(y_n = x_n - 1) = \beta_n)$, MiPOD tries to find probabilities minimizing a deflection coefficient, $\Sigma_n \sigma_n^{-4} \beta_n^2$, where σ_n is the variance of x_n. Notice that the product is element-wise applied and the sum concerns all the elements of the matrix. Thanks to a Wiener filter and a Lagrangian method, the change rates β_n can be computed. Considering such pixel probabilities, the distortion cost ρ_M is defined by:

$$\rho_M(X) = \ln\left(\frac{1}{\beta} - 2\right). \tag{4.2}$$

Again, the multiplicative inverse is applied element-wise.

Figure 4.2 from [79] shows clearly that, as noticed previously, the three steganographic algorithms we consider are the most secure, since they offer the better resistance against steganalysis with the state-of-the-art conventional steganalyzer maxSRMd2 [84]. It can be also seen that the detection errors increase as the embedding payload value, expressed in bits per pixel, decreases towards zero. Indeed, the lower the payload value the less pixels are modified and thus the more difficult it is to distinguish a stego from a cover. The experiments have been all carried out on BOSS [85] image database.

4.3/ Image steganalysis

Simultaneously with steganographic algorithms, steganalysis tools have evolved from early targeted techniques to approaches that are not specific for a given embedding scheme. In fact, the ultimate goal is to design an universal (or blind) steganalyzer able to deal effectively with a wide range of steganographic methods. Naturally, an universal steganalyzer is more complex to develop and can be less effective than a specific steganalytic algorithm for a given embedding scheme.

Most of the actual steganalyzers are feature-based approaches consisting of two steps. The purpose of the first stage is to extract useful information on image content by computing a set of features, while the second one will use this set to train a machine learning tool to decide whether the input image is a cover or a stego. Depending on how the features are defined, the current steganalyzers can be divided in two families. On the one hand, conventional image steganalysis approaches use Rich Models (RM) that include tens of thousands of handmade features and, on the other hand, as the deep learning approach called convolutional neural networks learns to automatically extract the most relevant high-level features of an input image to fulfill the classification task, the CNN-based steganalyzer family has recently emerged. In the following subsections we will present results in both steganalyzers families.

4.3.1/ Conventional steganalysis

Many rich models have been proposed over the past years for the spatial domain (SRM) [86, 87, 88] and the JPEG one [89]. Given a set of features provided by such a model, the second step, which aims at distinguishing stego images from cover ones, is then usually done in conventional state-of-the-art steganalyzers with Ensemble Classifier (EC) [90]. This classifier implements a decision function based on a simple majority voting between Fisher Linear Discriminant (FLD) components. A support vector machine [91] or a multilayer perceptron [92] are possible alternatives to EC.

The combination SRM+EC allows to detect images embedding a message with the steganographic algorithm HUGO [88] for payloads of 0.4 and 0.1 bpp, with errors of 13% and 37% respectively, on the BOSS steganalysis benchmarking database. These errors were slightly reduced (12% and 36%) in [87] thanks to an improved rich model, and a similar model denoted J(PEG)PSRM has been applied in the JPEG domain for the embedding scheme J-UNIWARD. Indeed, the rich image model used in [88] is based on features built as co-occurrence matrices from noise residuals and the size of these matrices increases exponentially with the neighborhood size of a pixel. The Projection Spatial Rich Model (PSRM) proposed in [87] solves this problem, and thus allows to reduce the detection errors, by projecting neighboring residual samples onto a set of random vectors and taking the histogram of the projections as the feature.

Today, Selection-Channel-Aware (SCA) variants of the popular SRM [84, 93, 94] provide the most accurate results, because they take into account the probabilities with which cover pixels are modified during the embedding process (content-adaptivity). For WOW, detection errors of 15% and 30% are obtained for payloads of 0.4 and 0.1 bpp, respectively, with maxSRMd2 [84], while for S-UNIWARD and HILL we have obtained the following corresponding errors: 19% and 40%, 23% and 43%. In [95] the authors have proposed a SCA variant for the JPEG domain, considering DCTR [89] and PHARM [96] as rich models. They also noticed that, as highlighted by the above detection errors with maxSRMd2, the security of S-UNIWARD and HILL is marginally decreased with SCA, whereas WOW suffers the most.

4.3.2/ CNN-BASED STEGANALYSIS

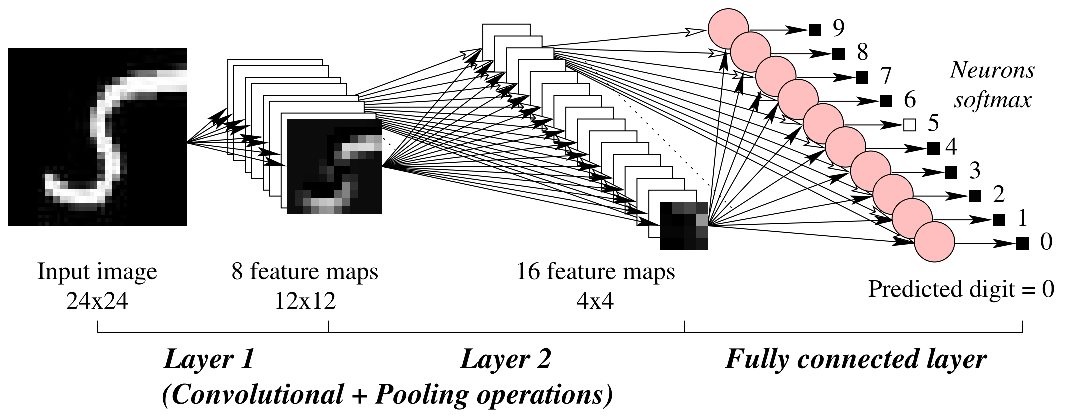
Deep learning architectures are particularly fruitful for solving various challenging tasks in computer vision, becoming the state-of-the-art for many of them. Among the different network architectures belonging to this family of machine learning methods, convolutional neural networks [66] are in particular well-suited for classification tasks based on the detection of variations in 2D shapes. Such deep learning networks achieved the best results for many image classification benchmarking problems like MNIST [64], CIFAR-10 [97], or CIFAR-100 [98]. CNNs are not only impressive with toy data sets like MNIST, they are also very successful when dealing with real life data such as images provided by medical imaging technologies. In this latter field of application, deep learning appears to be a breakthrough technology for fast and automated early detection of illnesses [99]. As image steganalysis is a similar classification problem and since both features extraction and classification steps of conventional image steganalysis approaches are perfectly embodied in the CNN architecture, different studies have investigated the design of a CNN-based steganalyzer these last years.

4.3.2.1/ NETWORK ARCHITECTURE

From an architecture point of view, a CNN is a feedforward neural network composed of two parts matching exactly the two steps used in the original conventional steganalysis scheme. The first part, called the convolutional part, consists of one or several layers trained to successively extract feature maps becoming smaller with the layer depth. The second part is composed of some fully-connected layers trained simultaneously to perform the classification task. This latter is somewhat similar to a multilayer perceptron that has more or less hidden layers of variable size. In fact, the most discriminant the features are, the less complex the second part. Hence, a CNN does not only learn how to classify, but also how to automatically find a set of features giving a better representation of the input image thanks to 2D convolution kernels.

A feature map is usually produced by one of the two following operations (see [67] for more details). First, a combination of filtered maps of the previous layer (or the input image for the first layer) followed by a nonlinear processing using an activation function. Second a size reduction (subsampling) through pooling, typically average or max pooling, over contiguous regions. While the first operation is done by a neuron of a convolutional layer, the second one is performed by a unit of a pooling layer. In the convolutional part, which consists of stacked convolutional and pooling layers in arbitrary order, the training aims thus at optimizing the kernel values and the bias of each convolutional neuron. Compared to fully-connected layers, convolutional layers have a lower training cost, since the sharing of all kernel values by all the pixels of the input image, or the feature maps in the case of inner layers, not only reduces the set of weights, but also improves the generalization performance. Sometimes for sake of simplicity, like in the following, when pooling is systematically applied after a convolutional operation, a layer will designate a combination of both operations.

(a)



(b)

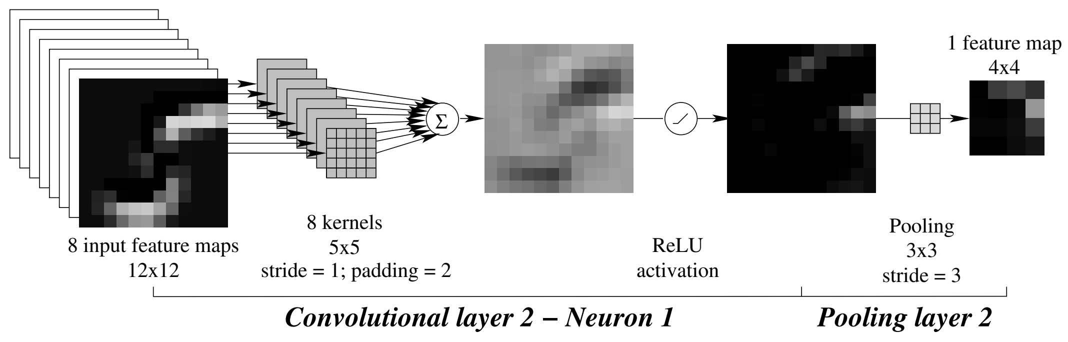


Figure 4.3: A Convolutional Neural Network for the MNIST problem: (a) global architecture and (b) detailed view of the second layer.

An illustrative example of a typical CNN architecture for the MNIST classification problem is shown in Figure 4.3(a). The MNIST is a large database of 28×28 pixels gray-level images obtained by normalizing and centering handwritten digits scans. It can be seen that the CNN consists of two layers, where each combines convolutional and pooling layers, producing 8 and 16 features maps, with size $12 \ \times 12$ and 4×4 pixels respectively, and a fully-connected part reduced to the output layer with one output neuron per possible digit value. The input image is slightly smaller (24×24 pixels) than the original one, because it results from a random cropping with a 24×24 window. Indeed, the final prediction for an original digit image is obtained by averaging the network prediction of four random crops, a data augmentation technique improving generalization.

Figure 4.3(b) shows a detailed view of the chaining of the different steps applied by the second layer in order to produce a feature map from the 8 features maps issued from the first layer. It can be seen that 8 convolutions of size 5×5, applied with stride and padding values equal, respectively, to 1 and 2, are combined together to form a feature map having a similar size to that of the input image. This map is then processed by a ReLU activation function to break the linearity introduced by the combination, followed by a max pooling with non-overlapping regions of 3×3 values (stride with value 3), finally leading to a feature map whose size, in comparison with the input image, is sixth times smaller in each dimension. A formal description of the different steps can be found in our paper [100], in which we have designed a CNN-based steganalyzer for embedding process with the same stego key. This later work is further discussed thereafter in Subsection 4.3.2.3.

4.3.2.2/ Training

As in other neural networks, the training (or learning) process consists in minimizing a training error (or loss function) using an optimization algorithm that updates the network parameters (weights and biases). A mini-batch gradient-based optimization algorithm is the most common choice and the Stochastic Gradient Descent (SGD) is very popular, particularly in the deep learning community, but more advanced techniques, such as Adadelta or Adagrad, might also be considered. Obviously, in that case the well-known backpropagation algorithm, which allows to compute the gradient of the training error, is the workhorse of the training process. A mini-batch gradient descent is interesting because it usually provides a more stable convergence. The network parameters are usually optimized until the number of epochs exceeds some specified maximum values, or when the training error reaches a given limit.

4.3.2.3/ A review of CNN-based steganalyzers

Since 2014, several research works have investigated the design of CNN-based steganalyzers bearing in mind the objective of outperforming state-of-the-art conventional steganalysis approaches. We will present works that have considered various steganographic algorithms, like HUGO [75], WOW [76], and S-UNIWARD [83] in [101], or only the last one in [102]. For the experiments, databases of 256×256 or 512×512 pixels gray scale images are classically considered, mainly from the well-known BOSS database [85] (*Break Our Steganographic System*) in the case of spatial steganalysis.

The first attempt at designing a CNN-based steganalyzer for image steganalysis is due to Tan *et al.* [103]. Their proposal, a stacking of convolutional auto-encoders, yielded for HUGO a detection error more than twice as bad as the one given by SRM+EC: 31% compared to 14% for a payload of 0.4 bpp.

The following study by Qian *et al.* [101] has dealt with 256×256 input images, and has proposed a CNN consisting of a convolutional part of 5 layers producing at the end 256 features, which are then processed by a fully-connected part of two hidden layers and a final output one of two softmax neurons giving, respectively, the probability of being a cover and a stego image. The preliminary high-pass filtering is done using a 5×5 kernel, called F_0, similar to the 5×5 kernel predictor defined in [86]. As noticed by Fridrich and Kodovský in [86], this kernel is inspired by a specific embedding algorithm, namely HUGO, but it worked well for the other steganographic algorithms they tested. The detection performance of this CNN was still slightly lower than the state-of-the-art SRM+EC conventional steganalyzer, but Pibre *et al.* [102] improved it thanks to a CNN with a different shape. By reducing the number of layers, but using larger ones resulting thus in more feature maps, they were able to reduce the detection error by more than 16% for S-UNIWARD at 0.4 bpp. They also emphasized that the high-pass filtering with F_0 was mandatory, since they were not able to train a CNN without it.

In comparison with the work of Pibre *et al.*, the CNN we designed in [104] being shallow as can be seen in Figure 4.4, was quite different and calling into question some assumptions previously made. On the one hand, we proposed a convolutional part of two layers: a first layer reduced to a single 3×3 kernel trained to replace F_0, followed by a layer using large kernels (almost as large as the image size). On the other hand, the resulting set of 256 features from second convolutional layer (for an input image of 512×512 pix-

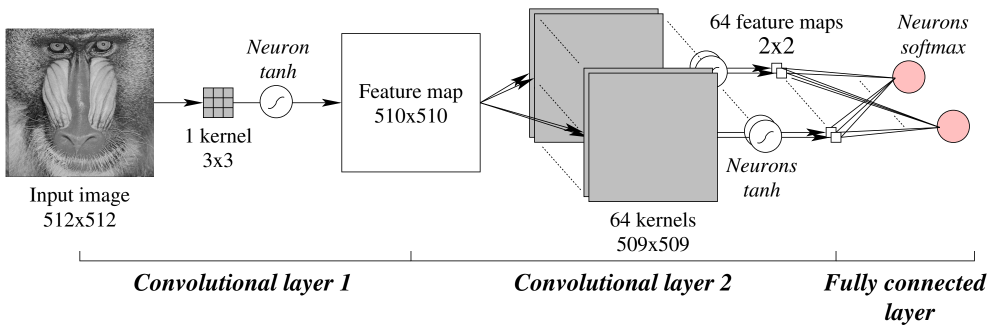


Figure 4.4: CNN for image steganalysis when embedding is done with same stego key.

els) was so discriminating that the fully-connected network doing the classification task could be shortened to the two final softmax neurons. As highlighted by Table 4.1, which reports detection accuracies for WOW, HUGO, and J-UNIWARD embedding schemes, with payloads of 0.1 and 0.4 bpp, this CNN is able to detect different steganographic algorithms, working in spatial and frequency domains, almost without any error for a payload of 0.4 bpp. The testing database consists of $8,000$ pairs of cover and stego images taken from Raise database and the CNNs are the best ones obtained after training on a part of the BOSS one with a scenario that can be seen as a version of the clairvoyant scenario considered in [102]. In this scenario, cover and stego images come from the same database and the steganographic algorithm, as well as the payload value, are supposed to be known. From the image-content point of view, the Raise database is rather different from BOSS database, therefore the testing scenario is equivalent to a cover-source mismatch scenario.

Unfortunately, our work, as well as the one of Pibre *et al.*, suffers from a crippling drawback: stego images were always obtained by using the same embedding key. In fact, this not recommended “same embedding key" scenario, since embedding several messages with the same key weakens security, comes from a mistake: the use of C++ embedding simulators from Binghamton DDE Lab website[1]. Consequently, when processing stego images produced with different embedding keys, as expected, the detection performance drops dramatically, far below the one of the state-of-the-art SRM+EC approach. Let us notice that the work by Qian *et al.*, described in the previous paragraph, certainly suffers from the same drawback.

More recently, the works [73] and [105] by Xu *et al.* have shown that CNN-based steganalysis remains competitive with conventional steganalysis. In [73] they first proposed a structural design of CNNs for steganalysis that is neither large, nor deep, and learns from noise residuals, since they considered as input image the one issued from high-pass filtering using the kernel F_0. The architecture of such convolutional networks, which is the basis of our work presented thereafter, will be described in detail in the next section. The experiments they completed have considered two spatial content-adaptive steganographic algorithms: S-UNIWARD and HILL. They have shown that the performance gained by an ensemble of five CNNs is comparable to the one of SRM+EC. In fact, they trained 5 CNNs because the data set was split into 5 test subsets, and thus an input image class was predicted by averaging the output probabilities of the CNNs. As an illustration, for S-UNIWARD they observed detection accuracies of 57.33% and 80.24% for payloads of 0.1 and 0.4 bpp, whereas the ones of SRM+EC were, respectively, of

[1]http://dde.binghamton.edu/download/stego_algorithms/

Table 4.1: Steganalysis results obtained from the best CNNs produced by the training process under clairvoyant scenario.

Steganographic algorithm and payload		Epoch	Detection accuracy		
Training	Testing	number	Cover	Stego	Total
WOW 0.4	WOW 0.4	34	94.96%	99.59%	97.28%
WOW 0.1	WOW 0.1	52	74.46%	81.03%	77.74%
WOW 0.1	WOW 0.4	52	74.46%	99.01%	86.73%
HUGO 0.4	HUGO 0.4	41	94.66%	99.53%	97.09%
HUGO 0.1	HUGO 0.1	54	79.90%	73.30%	76.60%
HUGO 0.1	HUGO 0.4	54	79.90%	99.28%	89.59%
J-UNIWARD 0.4	J-UNIWARD 0.4	9	85.60%	98.46%	92.03%
J-UNIWARD 0.1	J-UNIWARD 0.1	75	62.91%	83.91%	73.41%
J-UNIWARD 0.1	J-UNIWARD 0.4	75	62.91%	98.52%	80.72%

59.25% and 79.53%. However, the authors have not applied their approach to JPEG domain steganographic algorithms and they pointed out that in the spatial domain more advanced conventional methods, such as [94] or [106], still outperformed their approach.

In the following work [105], Xu *et al.* decided to study the merging of CNNs and ensemble classifier. The background idea was to train a second level classifier using information provided by CNNs. Furthermore, they also slightly modified the architecture of the original CNN designed in [73], which is denoted by ‘SIZE 128’. This new ‘SIZE 256’ CNN architecture, the number of final features given by the convolutional part, has one more layer and changed pooling sizes in the previous ones. In addition to the ensemble method described in the above paragraph, called PROB, where EC will use the output of 16 CNNs instead of five, they defined two further ensemble methods. The first one, PROB_POOL, is supposed to lower the loss of information induced by the pooling operation. Indeed, when the stride value is larger than one, some sampling operations are dropped. For a stride value $s > 1$, applying the pooling on a block of $p \times p$ pixels gives a single value, whereas for a stride of 1 the same block would have been replaced by $p \times p$ values. The idea is thus to also consider independently each remaining $p \times p - 1$ possible sampling, which means as they used pooling operations with stride of 2 that there are 4 possible samplings in a layer. Consequently, a CNN with a single layer would be applied 4 times on the same input image, giving a prediction for each possible sampling. For two layers, the number of sampling combinations is $4^2 = 16$, and so on. In the case of the ‘SIZE 256’ CNN architecture, since 16 trainings are done for each, the final prediction for an image is obtained by averaging $4^5 \times 16 = 1,024 \times 16 = 16,384$ predictions. The second new ensemble method, called FEA, is simpler: it uses an architecture merging the convolutional part and the ensemble classifier. Let us notice that the larger number of both CNNs and features results in lower bias and variance. From the experiments done with these 6 ensemble scenarios (two size and three methods), Xu *et al.* concluded that it might be interesting to replace the fully-connected part of the CNN by EC for image steganalysis. The FEA method was the best one for ‘SIZE 256’ with a detection error of 18.44%, but, compared to the 18.97% of PROB without EC, the improvement seems minor.

Another newly published contribution [107] by Qian *et al.* showed that a CNN trained to detect a spatial steganographic algorithm with a high payload embedding allows to improve the detection performance for lower payloads. Like in all the above works, the input of the CNN is again not the original image, but the one obtained after filtering with kernel F_0. For WOW, the pre-training with stegos obtained using a payload of 0.4 bpp led to lower detection errors in comparison with SRM+EC for 0.1, 0.2, and 0.3 bpp payloads.

Finally, we can notice the latest work [108] by Zeng *et al.* dealing with JPEG domain steganalysis. A domain which has received considerably less attention when designing CNN-steganalyzers. In this paper, the authors stated that compared to rich models, a CNN cannot efficiently learn to extract noise residuals, being unable to find similar or better kernels than the ones used in rich models. Therefore they proposed to start by manually applying to the input image the first two phases of DCTR [89], namely a convolution followed by *Q*uantization & *T*runcation (*Q&T*). They used 25 residual images, where each is obtained by using a 5×5 DCT basis pattern, and three *Q&T* combinations. Then, for each group of residual maps for a given *Q&T* combination, a subnetwork corresponding to a simplified version of the convolutional part proposed by Xu *et al.* in [73] is trained to produce a feature vector of 512 components. To obtain the final prediction, the three vectors are concatenated and given as input to a three-layer fully connected network, which is trained together with the three subnets. Based on the experiments performed on more or less large databases of images issued from ImageNet, the authors claimed that their proposal outperforms all other existing steganalysis approaches (no matter whether they are deep learning or conventional ones).

4.4/ Image steganalysis in spatial domain using a convolutional neural network

This section begins with the description of the CNN architecture proposed in [73], which is then experimentally studied in order to analyze what causes it to fail or succeed to detect whether an image embeds a hidden message.

4.4.1/ Shape of the CNN-based steganalyzer proposed by Xu et al.

As can be seen in Figure 4.5(a), Xu *et al.* proposed an architecture that takes as input the High-Pass Filtered (HPF) version of the input image issued from filtering with F_0 [86, 101, 102]. This filtering is obviously of great importance, since it provides the input information to the CNN, and thus must be suited to the classification task. The relevance of this kernel comes from its design for rich models. Overall, the CNN consists of 5 convolutional layers and a fully-connected part reduced to two softmax neurons, each of them giving the probability for an image to belong to one of the two classes (cover or stego). A classification part reduced to output neurons means that a linear classification is able to distinguish covers from stegos using the features produced by the final convolutional layer. We came to a similar classification part in our previous work [104] evoked above, even if in our case the problem was simpler due to the use of a single embedding key. Let us emphasize that this CNN was the first one not affected by the latter problem. In [101, 102] they had several hidden layers, more or less large (from 128 to $1,000$ neurons).

Starting with a HPF image of 512×512 pixels, the convolutional part results in a vector of 128 features, as shown by the detailed view of Figure 4.5(b). Each of the four first layers successively halves the image size by generating feature maps using an average pooling, while the fifth one replaces each feature map (32×32 pixels) by a single value

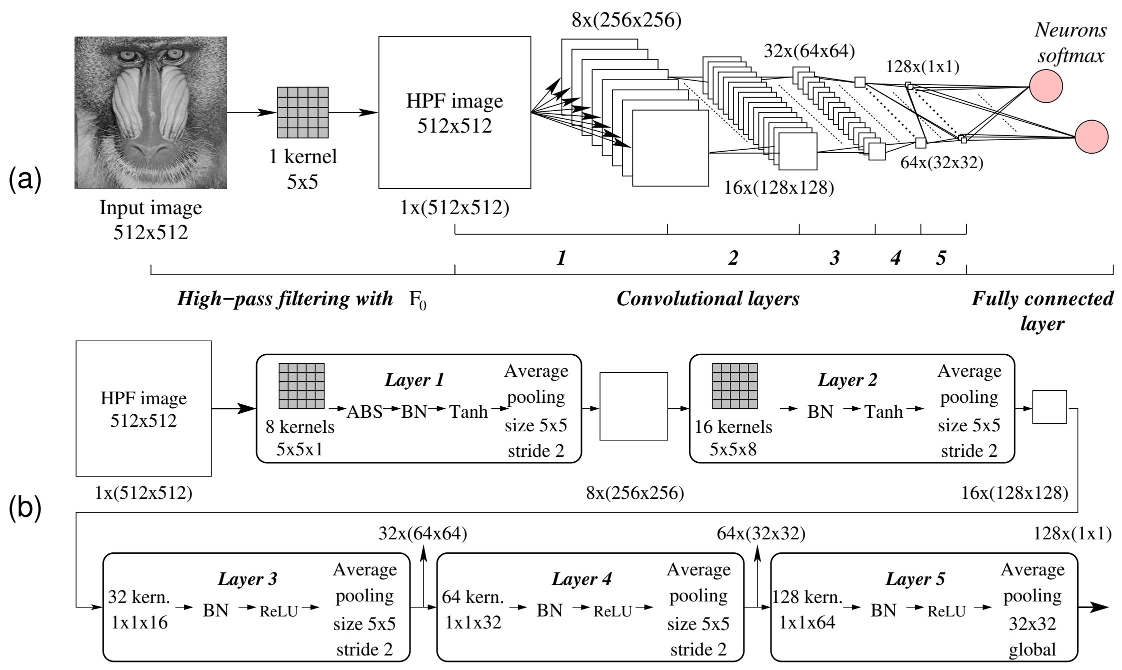


Figure 4.5: CNN proposed by Xu et al. in [73]: (a) overall architecture and (b) detailed view of the convolutional part.

obtained through a global average pooling. From the convolution kernel point of view, layers 1 and 2 learn 5×5 kernels, and the remaining layers 1×1 ones, the idea being to avoid an overfitting of the CNN to image content and/or stego noise. Layer 1 has also a specific function applied onto the outcome of the convolution, namely the absolute function (ABS), supposed to ensure that the model takes care of the symmetry in noise residuals like in rich models [86]. Batch Normalization (BN) [109] is performed in every layer, because it improves the training (faster and lower prediction error) by allowing a stable distribution of nonlinearity inputs. A mixing of *Tanh* and *ReLU* nonlinear activation functions offered the best performance.

4.4.2/ Detection performance evaluation of the CNN

To study and assess the performance of Xu *et al.* proposal, which was originally evaluated using a modified version of Caffe toolbox, the corresponding CNN has been implemented with the open source software library TensorFlow [110]. The implementation is available on download from `GitHub`[2]. We had previously used in [104] the computing platform Torch [111], but TensorFlow seems to make better use of the GPUs. All the experiments are performed on a NVIDIA Tesla Titan X GPU, using as cover images the $10,000$ images from BOSS database [85]. Six stego images are associated to each cover image. They are obtained by embedding a message with S-UNIWARD, MiPOD, and HILL schemes, thanks to the Matlab scripts downloadable from Binghamton DDE Lab website considering two different payload values: 0.1 and 0.4 bpp. During a training execution a CNN is trained on a set of $5,000$ cover-stego pairs and its detection performance assessed on the remaining $5,000$ pairs. Both training and testing sets are built by randomly picking pairs of images, using a different seed for each execution.

[2]https://github.com/rcouturier/steganalysis_with_CNN_and_SRM.git

Even if we implemented exactly a CNN according to the proposal, there is a major difference in comparison with the original work in the way the final prediction is obtained. In [73] Xu *et al.* generated from a training set five different non-overlapping $4,000/1,000$ splits and each of them is used to train separately a CNN occurrence. The final prediction for a given test image is then obtained by averaging the five output probabilities.

In our work, the final prediction is computed from a set of T trained CNNs which are denoted as $\mathrm{CNN}_1, \mathrm{CNN}_2, \ldots, \mathrm{CNN}_T$. First of all, each CNN_i, $1 \leq i \leq T$, memorizes its L last versions provided by the L last training epochs obtained all along the program execution. These internal CNNs are denoted as $\mathrm{CNN}_i^1, \mathrm{CNN}_i^2, \ldots \mathrm{CNN}_i^L$. Each of these internal CNNs gives an answer, which is 0 if the tested image I is declared as cover and 1 otherwise. Finally, the average of all the values is computed, and a discrete answer is returned by each CNN depending on whether this average is greater or equal to 0.5 or not. This is formalized for each i, $1 \leq i \leq T$, by:

$$\text{is\_stego}(I, \mathrm{CNN}_i) = \left\lfloor \frac{1}{L} \sum_{j=1}^{L} \text{is\_stego}(I, \mathrm{CNN}_i^j) + 0.5 \right\rfloor . \tag{4.3}$$

The aggregation of these results must take into consideration the fact that an image I we want to classify is used in training step in some CNN_i or not. Let us consider the set $\mathcal{T}_I = \{i | 1 \leq i \leq T \text{ and } I \text{ is used in testing step of } \mathrm{CNN}_i\}$ and T_I be its cardinality. The number T_I counts the number of times I is used as a testing image by some CNNs. The final answer is then the discrete answer of the average of all the CNNs that have used I as testing image. This is formalized by:

$$\text{is\_stego\_CNN}(I) = \left\lfloor \frac{1}{T_I} \sum_{i \in \mathcal{T}_I} \text{is\_stego}(I, \mathrm{CNN}_i) + 0.5 \right\rfloor . \tag{4.4}$$

Indeed, as both training and testing sets are built by randomly picking images, the number of times an image I is in a test set varies.

Due to the huge computation cost we have only trained CNNs using MiPOD data set and tested them directly on the S-UNIWARD and HILL data sets. Hence, we can also assess whether a CNN trained with an embedding scheme is still competitive for the other ones. We have chosen MiPOD because it is supposed to have the best security and thus should be the most difficult to detect, even if MiPOD and HILL are very close according to [79].

The key training parameters for reproducible experiments are discussed thereafter. First, a CNN is trained for a maximum number of training epochs $E_{\max}$ set to $1,000$ and 300, respectively, for embedding payloads of 0.1 and 0.4 bpp, without any overfitting control with a validation set. In fact, overfitting did not appear as a problem due to a careful choice of the values for $E_{\max}$ and the evaluation of the validation set was also time consuming, so we decided to drop this control. To compute the prediction given by a network CNN_i, $L = 20$ occurrences are used. Second, the network parameters are optimized by applying a mini-batch stochastic gradient descent, using a mini-batch size of 64 samples, but without ensuring that both cover and stego of a given pair are in the same batch like Xu *et al.*. The gradient descent parameters are: a learning rate initialized to 0.001, but with no weight decay, and a momentum set to 0.9.

The obtained average detection errors are reported in Table 4.2. The first line labelled with “Caffe [73]” recalls values given in [73]. It should be noticed that "X" means that HILL

Table 4.2: Average detection error as a function of classifier (original Caffe by Xu *et al.*, our TensorFlow trained only with MiPOD at 0.4 bpp, and SRM+EC) and of payload.

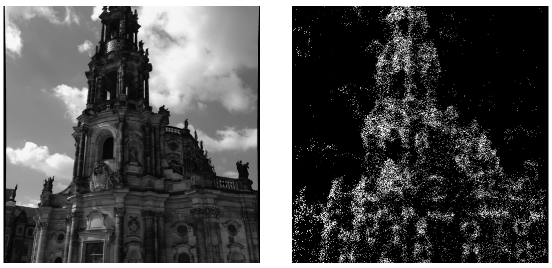
	S-UNIWARD		MiPOD		HILL	
	0.1	0.4	0.1	0.4	0.1	0.4
Caffe [73]	42.67	19.76	X	X	41.56	20.76
TensorFlow	47.38	20.52	43.72	19.36	46.79	20.25
SRM + EC	39.84	18.06	41.18	21.42	42.96	23.31
SRM + EC (blind)	40.57	20.85	41.18	21.42	43.35	23.99

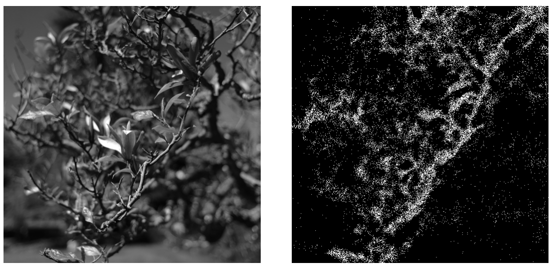
has not been evaluated with the Caffe framework. The second line gives the average error rates for embedding payloads of 0.1 and 0.4 bits per pixel, using respectively $T = 9$ and $T = 12$ independent training runs of the CNN TensorFlow implementation detailed in the previous paragraphs. The third gives the average error rates for 200 runs with classical SRM+EC. In this latter context maxSRMd2 [84] has been used as a feature set. Finally, the last line gives the results obtained when the training stage is executed with images modified by MiPOD, whereas the testing stage is executed with images modified with another embedding scheme (SRM+EC and maxSRMd2 are still used). Remember that an objective is to develop a completely blind steganalysis approach. It is not difficult to understand that the detection error is larger in this context, since the steganographic scheme used to learn is not necessarily the one used in the testing phase.

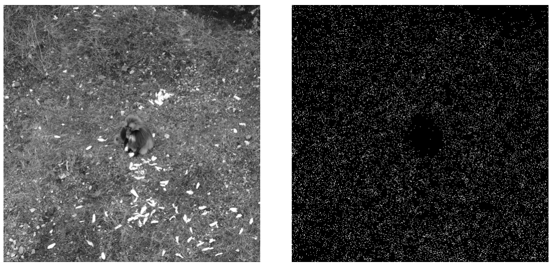
From the values given in this table we can draw several conclusions. First, despite the differences highlighted previously, for a payload of 0.4 bpp the TensorFlow implementation produces nearly the same performance for S-UNIWARD and HILL than the original Caffe one, while for 0.1 bpp the performance is worse due to the blind context. Second, we observe that SRM+EC results in the best performances for S-UNIWARD in case of non-blind steganalysis. Third, for MiPOD the CNN approach is still competitive with SRM+EC. Fourth, the CNNs trained by only making use of the MiPOD data set can provide a similar detection accuracy for S-UNIWARD and HILL, even if for the payload of 0.1 bpp a larger degradation of the accuracy can be noticed. Obviously, the lowest detection error is gained for the embedding scheme which has provided the training data. Fifth, CNNs outperform SRM+EC in blind steganalysis context with a payload of 0.4 bpp, which means that CNNs allow a better generalization to different steganographic algorithms.

4.4.3/ Characterizing the images misclassified by the CNN

Let us start with some illustrative examples of images describing the typical behavior of the CNN in the case of MiPOD with payload 0.4 bpp. Figure 4.6 presents four case examples where for each we have the cover image and the corresponding differences between it and the stego one. From the images showing the differences we can distinguish two groups of images according to the pixels modified by the embedding process. It clearly appears that for both images shown on the upper line, 1388.pgm and 8873.pgm, MiPOD mainly modifies pixels corresponding to edges. For 1911.pgm and 3394.pgm, changes are scattered without obviously highlighting any underlying image edge. Consequently, since a CNN mainly learns to detect underlying edges, one can easily guess that the CNN-steganalyzer is able to detect both cover and stego for 1388.pgm and 8873.pgm, whereas it fails for the two other images. We are then left to provide a metric on images


(a) Cover 1388.pgm and differences with stego.


(b) Cover 8873.pgm and differences with stego.


(c) Cover 1911.pgm and differences with stego.

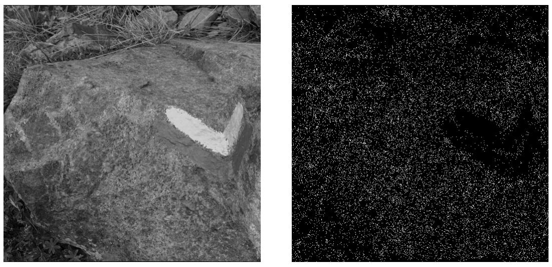

(d) Cover 3394.pgm and differences with stego.

Figure 4.6: Examples of differences images between cover and corresponding stego when embedding is performed using MiPOD with a payload of 0.4 bpp.

which reflects the difficulty to perform the CNN classification task.

The first function we have investigated as possible metric is the image Shannon entropy $H(I) = -\sum_i p_i \log_2(p_i)$, a function that returns the number of bits which are necessary to recompute the image, where p_i is the normalized histogram of pixel values of image I. On the BOSS database, the entropy ranges in $[0.7, 8]$ and its distribution is represented in Figure 4.7.

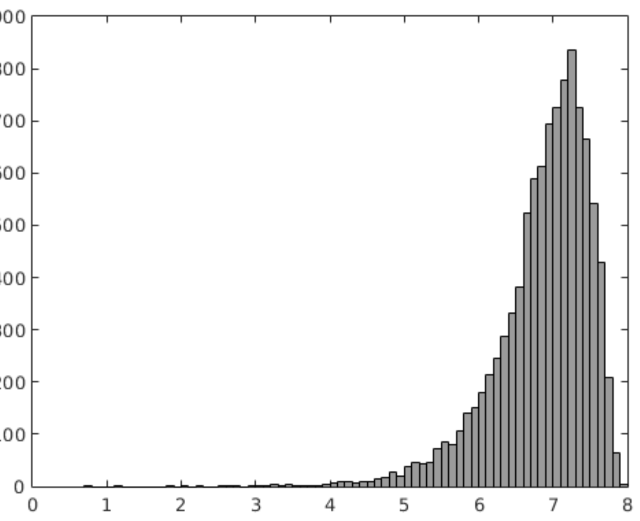


Figure 4.7: Distribution of the BOSS database images with respect to entropy.

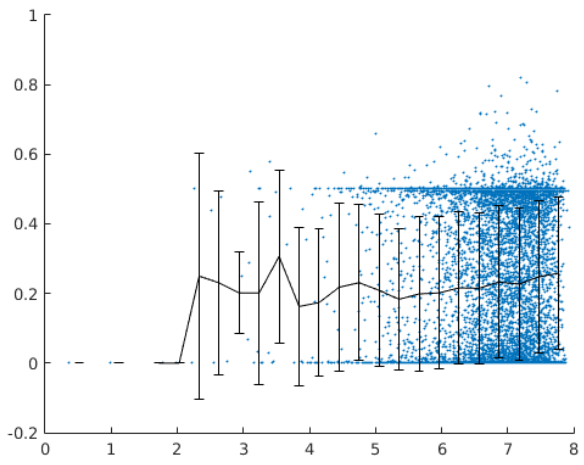
To check whether image entropy is a suitable measure of the difficulty to classify an image, we have then plot for each steganalyzer the average detection error for all images I of the database according to their respective entropy value $H(I)$. In the case of SRM+EC, 200 classification were performed for each image thanks to maxSRMd2 for the embedding algorithm MiPOD at payload 0.4 bpp in order to compute $\overline{e_{SRM+EC}}(I)$, the average testing error for image I when it is used in the testing set. We have also partitioned the entropy interval $[0.7, 8]$ into 25 equidistant classes. Figure 4.8(a) presents the resulting scatter plot of $(H(I), \overline{e_{SRM+EC}}(I))$ pairs and the curve linking the mean error of each class, whereas the bar displays its corresponding standard deviation. Similarly, Figure 4.8(b) shows the scatter plot, curve, and error bars, for the CNN. In that case, $\overline{e_{CNN}}(I)$ is the average testing error obtained after training 12 independent networks. This low number explains why, in comparison with the SRM+EC steganalysis context, the points are less vertically spread. A first conclusion of these experiments is that the detection error of SRM+EC seems to be quite independent of the entropy, whereas the one issued from CNN seems to increase as the entropy does. However most of the images have an entropy in $[6, 8]$ (as can be seen in Figure 4.7) and it is hard to extract from Figures 4.8(a) and 4.8(b) a specific behavior inside this interval.

As noticed previously, a key function in an embedding scheme is the distortion function ρ that reflects the underlying structure of the image. A pixel in an easy-modeled area has a high distortion value, while in a area showing large variations such as a textured area a low value is obtained. Therefore, we have then studied whether a metric can be deduced from the distortion function. Considering the same experimental context than for the entropy function, on the one hand, Figures 4.9a, 4.9c, and 4.9e display the scatter plots, the curve, and error bars for $(\overline{\rho_U}(I), \overline{e_{SRM+EC}}(I))$, $(\overline{\rho_H}(I), \overline{e_{SRM+EC}}(I))$, $(\overline{\rho_M}(I), \overline{e_{SRM+EC}}(I))$ respectively. On the other hand, Figures 4.9b, 4.9d, and 4.9f display the scatter plots, the curve, and error bars for $(\overline{\rho_U}(I), \overline{e_{CNN}}(I))$, $(\overline{\rho_H}(I), \overline{e_{CNN}}(I))$, $(\overline{\rho_M}(I), \overline{e_{CNN}}(I))$ respectively.

The scalar $\overline{\rho_U}(I)$ is the mean of all the matrices $\rho_U(X)$ presented in equation (4.1), where U means S-UNIWARD. The scalar $\overline{\rho_M}(I)$ has a similar definition for MiPOD. Finally $\overline{\rho_H}(I)$ is not directly the mean of all the matrices $\rho_H(X)$ of HILL. Due to its definition, some extremely large values may indeed result from an extremely small denominator. This would lead to a meaningless mean value. To avoid this behavior, extremely large values are excluded from the mean computation.

By focusing on Figures 4.9a, 4.9c, and 4.9e, it can be first deduced that the detection error of SRM+EC is quite independent of the value of $\overline{\rho}$. Secondly, considering Figures 4.9b and 4.9d, we can deduce that the CNN testing error continuously decreases with respect to $\overline{\rho_U}(I)$ and with $\overline{\rho_H}(I)$. This behavior is not observed in Figure 4.9f. The good correlation between the prediction accuracy of the CNN for a given image I and the value of $\overline{\rho}(I)$ can be observed in the two former cases but not in the last one. The functions $\overline{\rho_U}$ and $\overline{\rho_H}$ are thus an indicator of the CNN accuracy. For instance, in Figure 4.6, for the misclassified images we obtain $\overline{\rho_U}(1911) = 2.1$ and $\overline{\rho_U}(3394) = 3.06$, while for the well detected images we get $\overline{\rho_U}(1388) = 7.05$ and $\overline{\rho_U}(8874) = 7.39$. Thus $\overline{\rho_U}$ and $\overline{\rho_H}$ enable to cluster the images in two groups which are in accordance with those noticed at the beginning of the subsection.

(a)



(b)

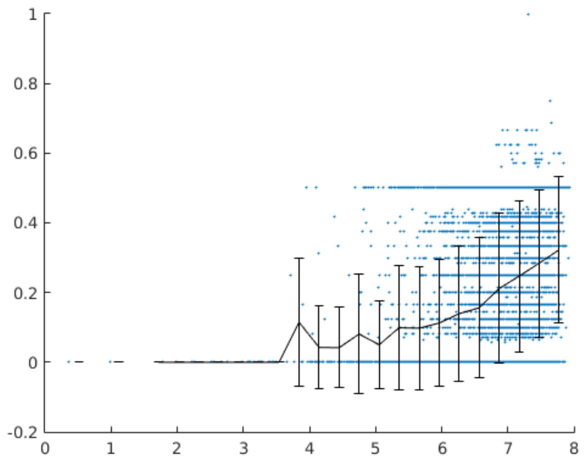


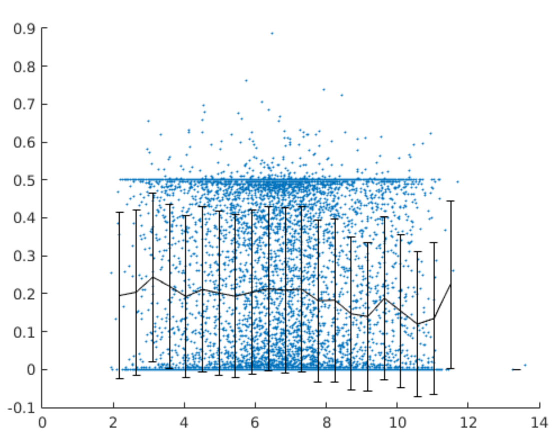
Figure 4.8: Relation between testing errors and image entropy for embedding with MiPOD at a payload 0.4 bpp with: (a) SRM+EC steganalyzer and (b) CNN-based steganalyzer.

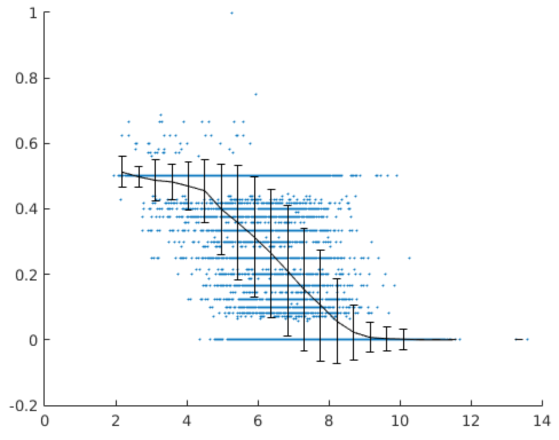
4.5/ Improving blind steganalysis in spatial domain

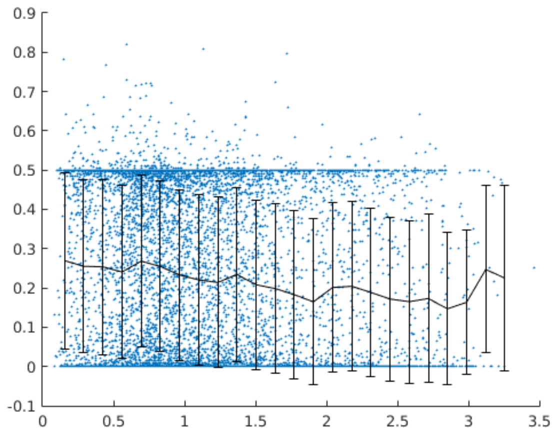
4.5.1/ Taking the best from CNN and SRM+EC predictions

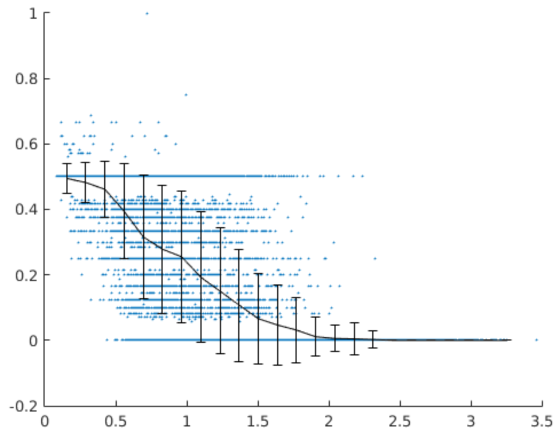
We have shown that the lower the distortion function mean $\overline{\rho_U}$ of an input image is, the more difficult it will be for the CNN to correctly detect whether the image is a cover or a stego. Conversely, SRM+EC gives rather regular detection errors, without showing too much sensitivity to $\overline{\rho_U}$, being robust against the image structure. A look at Figures 4.9a and 4.9b shows that we can take advantage from these different behaviors to improve the detection performance on the BOSS base.

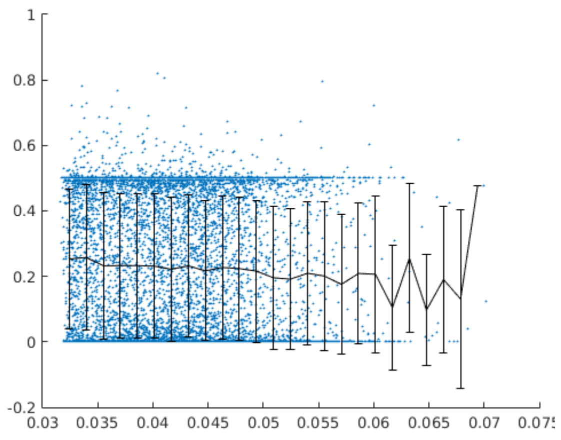
In fact, SRM+EC and the CNN can be combined due to complementary purposes. As can be seen in Figure 4.10, in which both curves obtained for $\overline{\rho_U}$ are superimposed, from the largest $\overline{\rho_U}$ value up to the point where both curves intersect the CNN is the most competitive, whereas after, towards the lowest $\overline{\rho_U}$ value, it is SRM+EC which is clearly


(a) Detection error w.r.t image $\overline{\rho_U}$ value for SRM+EC.


(b) Detection error w.r.t image $\overline{\rho_U}$ value for the CNN.


(c) Detection error w.r.t image $\overline{\rho_H}$ value for SRM+EC.


(d) Detection error w.r.t image $\overline{\rho_H}$ value for the CNN.


(e) Detection error w.r.t image $\overline{\rho_M}$ value for SRM+EC.

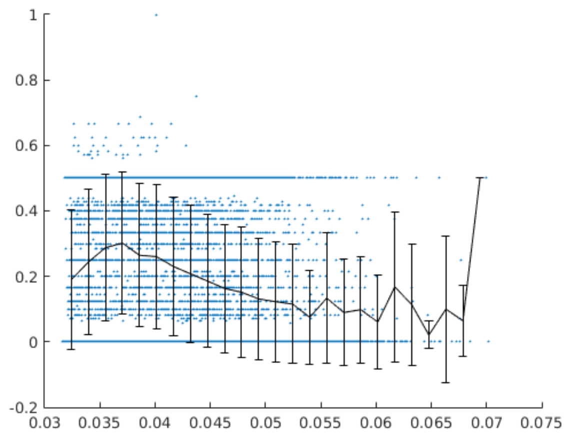

(f) Detection error w.r.t image $\overline{\rho_M}$ value for the CNN.

Figure 4.9: Relation between testing errors and distortion function mean.

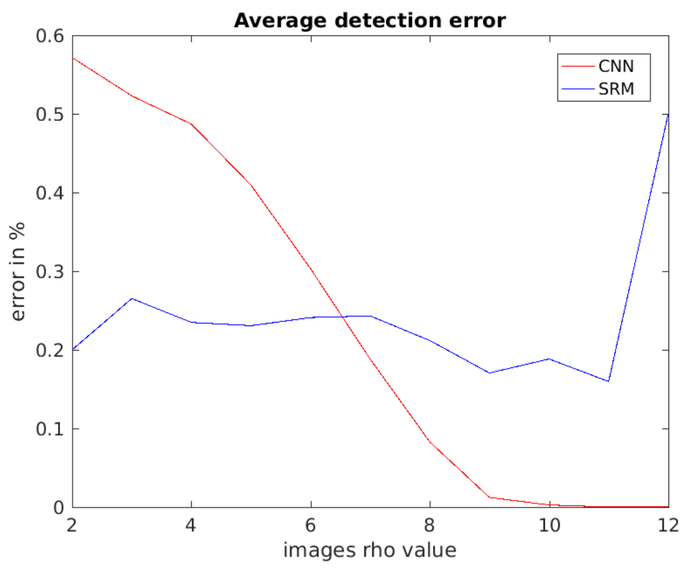


Figure 4.10: Average detection error of CNN-based and SRM+EC steganalyzers for MiPOD 0.4 bpp w.r.t $\overline{\rho_U}$.

the most accurate. Formally, this can be expressed as follows for an input image I, once $\overline{\rho_U}(I)$ is computed:

$$\begin{cases} \text{if } \overline{\rho_U}(I) < \overline{\rho_U^{\cap}} \text{ use SRM+EC prediction,} \\ \text{otherwise use CNN prediction} \end{cases} \tag{4.5}$$

where $\overline{\rho_U^{\cap}}$ corresponds to the intersection abscissa. For Figure 4.10, we have obtained $\overline{\rho_U^{\cap}} = 6.6$. Let us emphasize that the same approach can be applied to S-UNIWARD and HILL steganographic algorithms, leading to different values for $\overline{\rho_U^{\cap}}$.

Overall, the feature set generated by a spatial rich model is so large and diverse that it is able to give predictions yielding almost the same level of accuracy, regardless of the pixels modified by the embedding process. Moreover, the computing of the features is precisely defined. Conversely, the CNN learns to extract a set of features to fulfill its classification task according to the data given during the training step. Therefore, it will be well-suited to process images having the same kind of embedding than the main trend in the training data set. In other words, images having low $\overline{\rho_U}$ values are so underrepresented in the BOSS database that they have a limited influence during the training process.

4.5.2/ Detection performance evaluation of the proposal

Table 4.3 presents in its last column the average detection error obtained for the three steganographic algorithms. The first column gives the performance of SRM+EC computed on images I such that $\overline{\rho_U} < \overline{\rho_U^{\cap}}$, this last value is shown in the second column, while the third column shows the results gained from CNN for the remaining images. We can observe that the proposal improves the detection performance for each embedding algorithm. For a payload of 0.4 bpp, S-UNIWARD has the lowest error rate with 14.82%, whereas for MiPOD and HILL we have values slightly below 17%. The lines labelled as non blind correspond to situations where SRM+EC was trained with the same algorithm than the one used to perform the embedding. Conversely, the lines denoted as blind

Table 4.3: Average detection error according to $\overline{\rho_U^\cap}$ for different embedding payloads.

	SRM+EC		$\overline{\rho_U^\cap}$		CNN		CNN + SRM+EC	
Payload (bpp)	0.1	0.4	0.1	0.4	0.1	0.4	0.1	0.4
S-UNIWARD non blind	40.08	20.01	9.2	7.1	23.36	8.25	38.06	14.82
S-UNIWARD blind	41	22.05	9.2	6.9	23.36	9.5	38.88	15.87
MiPOD non blind	42.13	23.89	8	6.6	25.84	9.26	37.82	15.65
HILL non blind	43.48	24.51	8.9	6.6	21.88	9.78	40.24	16.22
HILL blind	44.30	25.41	8.3	6.6	27.72	9.78	40.64	16.61

mean that SRM+EC was trained with MiPOD and then used to detect S-UNIWARD or HILL. This also explains why for both blind and non blind situations the CNN gives the same error when both cases use the same value for $\overline{\rho_U^\cap}$. For the lower payload of 0.1 bpp, using $T = 9$ independent runs, the improvements provided by our method are also clearly visible. These results are also somewhat surprising, since they are obtained by training only CNNs using images embedding hidden messages with MiPOD. This means that even if each steganographic algorithm has its own distortion function, there is certainly a high redundancy among the modifications they made on the same cover image.

A closer look on the performances of each steganalyzer on the subset of images it has to classify according to $\overline{\rho_U^\cap}$ explains why our proposal is relevant. Indeed, in comparison with the performances shown in Table 4.2 we can remark that the SRM+EC error rate is slightly worse than on the whole data set. Thus we take advantage from the low error rate of the CNN at a price of a slightly worse misclassification by SRM+EC. Another point to notice is the evolution in opposite directions of $\overline{\rho_U^\cap}$ and payload values, which means that, as expected, the scatterness of the modified pixels increases and thus is more difficult to detect with the current CNN architecture. Nevertheless, our approach allows us to build a competitive blind steganalyzer, which gives lower detection errors than CNN based only or SRM+EC based only approaches.

4.6/ Conclusion

In this chapter, rather than designing a further new CNN-based steganalyzer for image steganalysis in spatial domain, we have studied the CNN architecture proposed by Xu *et al.*, which, to the best of our knowledge, was the most competitive one compared to the state-of-the-art conventional steganalyzer SRM+EC at that time. More precisely, we have investigated when this CNN fails in order to propose a method allowing to improve the detection performance on the BOSS database for the three most secure spatial steganography algorithms, namely S-UNIWARD, MiPOD, and HILL. Thanks to a TensorFlow implementation, a careful experimental study allowed us to identify a metric strongly correlated with the CNN classification performance. This metric consists in the mean of all the elements in the cost matrix provided by the distortion function of the considered steganographic algorithm for the input image. As the CNN and SRM+EC detection errors evolve in different ways according to this metric function, we were able to define a reliable criterion allowing to choose, for a given input image, which steganalyzer among the CNN and SRM+EC will provide the most accurate prediction. The relevance of the criterion has been assessed experimentally for the considered embedding schemes, regardless of the payload value. Finally, the designed steganalyzer is not targeted at a particular known steganographic algorithm, and consequently it can perform blind steganography detection.

5

Image Denoising using a Deep Convolutional Encoder-Decoder Network with Skip Connections

This chapter briefly describes our most recent work on the design of a deep neural network for an image restoration task. Image restoration is the set of all the image processing algorithms that are applied in order to improve the visual quality from a human observer point of view. Since an image can be degraded by many factors such as blur or noise, image restoration encompasses different tasks among which deblurring, denoising, super-resolution, and so on.

More specifically, we have focused our attention on image denoising whose objective is to recover a clean image from its noisy version while preserving its original underlying structure. Several reasons explain our interest in this problem. First, in many areas images can be corrupted by various types of noise and therefore image denoising is a prerequisite. For example, medical images like the 4D-CT considered in Chapter 1, or ultrasound ones, are prone to noise and artifacts that can affect diagnostic confidence. Remote sensing is another field for which image preprocessing is mandatory to improve the quality of source images. Synthetic Aperture Radar (SAR) images are typically corrupted by speckle noise generated by constructive and destructive electromagnetic wave interference during acquisition. Second, we have in mind that traditional existing denoising methods are usually specific for a kind of noise with a given level. Thus, as was the case for image steganalysis, our final objective is to develop a neural network able to perform blind denoising, which means without any prior knowledge on the noise. Moreover, denoising is a problem that has already been considered in our research team, since Perrot *et al.* [112] have worked on the design of a specific GPU-based high-speed denoising filter for additive white Gaussian noise. Let us finally precise that the work we present thereafter has not been published yet, is still in progress, and that only the great lines of the neural network architecture will be given.

The proposal is presented throughout four sections. We start in Section 5.1 with a discussion on related existing denoising methods. An overview of the proposed deep neural network design with its main characteristics is done in the following section. The possible choices for the loss function used to train the neural network, together with the one we have defined, are also analyzed. Section 5.3 is dedicated to the first experiments, showing the validity of the approach. Finally, some concluding remarks are given in Section 5.4.

5.1/ Introduction

In today's digital world, an increasing amount of digital images is produced every day. Nevertheless, the visual quality of an image is not guaranteed, since different sources of noise can influence the pixel values. A main source is the acquisition process and particularly the presence of defaults in the capturing device: noise can be produced by the sensor, misaligned lenses, and so on, but noise can also be added during its edition, storage or transmission. As a result different types of noise can appear in a digital image, such as Gaussian noise, Salt-and-pepper noise, uniform noise, etc., and at different levels. For an observer, the impact of noise can range from isolated speckles up to images that seem to show nothing but noise.

To recover as precisely as possible a clean image y from a noisy version x that is the outcome of an arbitrary stochastic corruption process n: $x = n(y)$, an efficient image denoising method is needed. Formally, the goal of image denoising is thus to find a function f that approximates as well as possible the inverse function of n:

$$f = \underset{f}{\operatorname{argmin}} \ \mathsf{E}_y \parallel f(x) - y \parallel_2^2 . \tag{5.1}$$

It should be noticed that additive white Gaussian noise is often targeted, in which case the corruption process can be rewritten as $x = y + \mathcal{N}(0, \sigma)$ where σ is the standard deviation. This inverse problem is known to be ill-posed and therefore, from a Bayesian perspective, it is important to exploit prior knowledge to constraint the solution space.

To solve this problem, there are two main categories of methods: model-based optimization methods and discriminative learning methods. The objective of the former methods is to directly solve the optimization problem, but as this problem is usually complex, they are time consuming. On the other hand, discriminative learning methods try to learn a set Θ of parameters defining a nonlinear function $\hat{f}$ that approximates n^{-1} by minimizing a loss function according to a data set that consists of clean-noisy images pairs. In that case, the previous problem can be expressed as follows:

$$\Theta = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \parallel \hat{f}(x_i) - y_i \parallel_2^2 \tag{5.2}$$

where x_i is the noisy version of y_i and N is the size of the data set. Compared to model-based methods, discriminative ones are less flexible since they are usually trained to deal for a specific underlying model of corruption.

Typical examples of model-based methods are BM3D [113] and WNNM [114], while neural networks are representatives of the discriminative family. Obviously, even if the MLP has been investigated [115], with the current rise of deep learning, deep neural networks are now the most actively studied discriminative learning methods. One of the first deep network proposal was made by Xie *et al.* [116] in 2012, it consisted in a stacking of auto-encoders where each auto-encoder was trained one after the other (the hidden layer of each auto-encoder is composed of sigmoidal neurons). In 2014, Long *et al.* [117] introduced the Fully Convolutional Networks (FCN) for semantic segmentation, an architecture that allows to produce segmentation maps whatever the image size and faster than with patch classification approaches. Furthermore, this work has led to the widespread use of deep networks in which the fully connected part is dropped, and thus reduced to the convolutional part with a single feature map / image as output, for dense predictions.

Image denoising is such a dense prediction task, whose objective is to recover for each pixel its original gray level value. Consequently, among the various deep networks that have recently been investigated to tackle the image denoising problem, almost all of them have adopted the FCN paradigm. However, even if these networks belong to the same family, differences among them can been observed. First, a FCN can be trained to recover directly the clean image, like other discriminative denoising models such as a MLP, or to predict a residual image that is subtracted from the noisy input one [118]. Second, a central problem when using a CNN for image denoising (or segmentation) is due to pooling layers. Indeed, a pooling layer usually performs a spatial downsampling and as the input and output images must have the same size, it means that an upsampling process is needed. On the one hand the pooling permits to enlarge the field of view, but on the other hand the aggregation throws away useful spatial information. To address this issue, several different architectures have emerged: architectures without pooling layers and the encoder-decoder architecture.

A first example of architecture for image denoising that only has convolutional layers is the deep network called DnCNN (Deep network CNN) proposed by Zhang *et al.* [118] considering a residual learning formulation. The CNN is composed of layers with three different convolutional blocks using a unique convolution kernel size of 3×3: Convolution+ReLU for the first layer, Convolution+BatchNorm+ReLU in the intermediate layers, and only Convolution in the final layer. It has a receptive field whose size depends on the network depth ($(2d + 1) \times (2d + 1)$ where d is the depth) and which is correlated with the effective patch size of other denoising methods. In fact, most of the denoising methods such as BM3D, WNNM, MLP, and so on operate on patches. The authors have thus chosen to increase the receptive field through a large depth, a choice that is more popular than the use of less layers with increasing kernel sizes due to the computational overhead resulting from a larger number of parameters to optimize. The networks of 17 and 20 layers that they trained for additive Gaussian denoising, respectively for a specific noise level and for blind denoising (unknown noise level), outperformed slightly both BM3D and WNNM on the BSD68 data set of grayscale images. The choice of these depth values is issued from an analysis of the effective patch size of other denoising methods (BM3D, WNNM, etc.).

A recently proposed alternative to an increased depth or to increasing filter sizes is the use of dilated kernels, also known as atrous convolution. Indeed, convolutional layers that only use 3×3 kernels but with multiple atrous rates perform an analysis of the image at multiple scales without needing a large depth. Figure 5.1, which can be found in [119, 120], shows the principle of atrous convolution with three different rates. It can be clearly seen that the model's field-of-view becomes larger as the rate value increases. This approach has also been studied by Zhang *et al.* in another work [121], leading to denoising performances similar to that of the first-mentioned work.

Zhang *et al.* have finally introduced another architecture [122] with the objectives of being able to handle a wide range of noise levels and spatially variant noise through the definition of a noise level map, and also to provide a faster execution than BM3D. This architecture, called FFDNet for fast and flexible denoising convolutional neural network, consists of a CNN similar to the one of DnCNN, but that does not predict the noise. The CNN receives as input four sub-images obtained from the initial input image using a reversible downsampling operator (factor is set to 2) and a tunable noise level map. As output it produces four denoised sub-images which are then upsampled to recover the final output image. For Additive White Gaussian Noise (AWGN) removal, the experiments

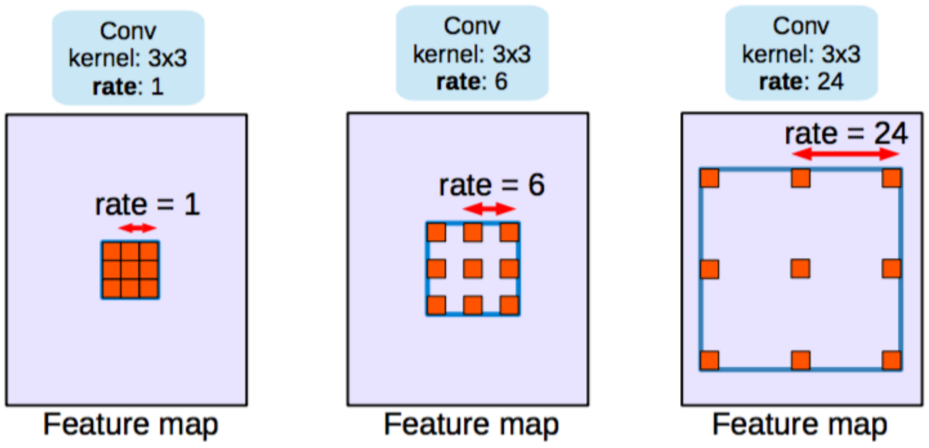


Figure 5.1: Atrous convolution with kernel size 3×3 and different rates.

show that DnCNN is better for low noise levels ($\sigma \leq 25$), whereas for larger values FFDNet becomes gradually slightly better with the increase of noise level. This result is all the more interesting as it is the version of DnCNN trained for a specific noise level that is considered, whereas FFDNet is trained in a blind context with noise level $\sigma \in [0; 75]$.

An encoder-decoder is quite different. First of all, the encoder consists of convolutional layers that successively downsample the input image into small abstraction maps from which the noise is removed as the process goes deeper. The decoder is then fed by the final abstraction map in order to reconstruct a clean image thanks to deconvolutional layers. The reconstruction by the decoder is clearly the most difficult part since image details might be lost during the features extraction process performed by the encoder. To mitigate this problem, a common approach is to adopt the skip connection method. In this work we have considered an encoder-decoder architecture with such connections and its broad outlines are given in the following section.

5.2/ The proposed deep network for image denoising

5.2.1/ Network architecture

Our network is similar to a generator architecture introduced by Isola *et al.* [123] in their investigation of conditional adversarial networks to solve image-to-image translation problems, an architecture which is itself an adaptation of one issued from [124]. The generator we consider is the U-Net [125] version corresponding to an encoder-decoder having skip connections between mirrored layers in both encoder and decoder stacks, as shown in Figure 5.2. The encoder extracts salient features preserving the detailed underlying structure of the image, while simultaneously removing the noise, whereas the decoder produces a clean version of the input image by recovering successive image details as it progresses through its layers in a bottom-up way from the bottleneck layer of the encoder. As can be seen, each skip connection allows to directly shuttle the information from an encoder layer to its corresponding decoder one, and this is appealing since the input noisy image and output clean version share large parts of the low-level information like the location of prominent edges. In fact, skip connections allow to remember different levels of details that are useful to reconstruct the final output image.

Symmetric skip connections are very interesting because they facilitate the training and improve image recovery. On the one hand, skip connections allow to solve the vanishing gradient problem by backpropagating the signal directly and, on the other hand as both input and output images have the same content, the recovery of the clean version can benefit from the details appearing in the corrupted version. As a consequence, better results are usually obtained with skip connections.

From an architecture point of view, following the specification given in the appendix of [123], the encoder is almost exclusively composed of Convolution-BatchNorm-ReLU layers, a typical choice in CNN, except the first layer that do not undergo the batch normalization. Batch normalization [109] is performed because it improves the training (faster and better denoised images) by allowing a stable distribution of nonlinearity inputs. Let us notice that, as illustrated in the previous section, this kind of convolutional block is also the one used in many FCN that do not have an encoder-decoder architecture. For its part, the decoder consists of a mixing of this kind of layer and a variant of it integrating a dropout rate of 50% before the ReLU activation. Neither pooling nor unpooling operations are used, because aggregation induces some losses of details which, in the context of image denoising, can be awkward. Since convolutions and deconvolutions use 4×4 kernels with stride 2, each encoder and decoder layer will produce feature maps which are downsampled and upsampled, respectively, by a factor of 2. The last top decoder layer is mapped back to the output clean image with a deconvolution followed by a $\tanh$ activation.

5.2.2/ Loss function

A factor that has a major impact on the obtained neural network is obviously the loss function used to drive the training process. However, despite its importance, the choice of this function is hardly discussed in most of the research works studying the design of neural networks for solving image processing problems. Indeed, usually the choice simply consists in deciding whether to use the $L1$−norm or the $L2$−norm, the latter being the most popular option, and besides, we already used it in the introduction section. However, even if properties of the $L2$−norm (or Euclidean norm) explain why it is the default choice, in

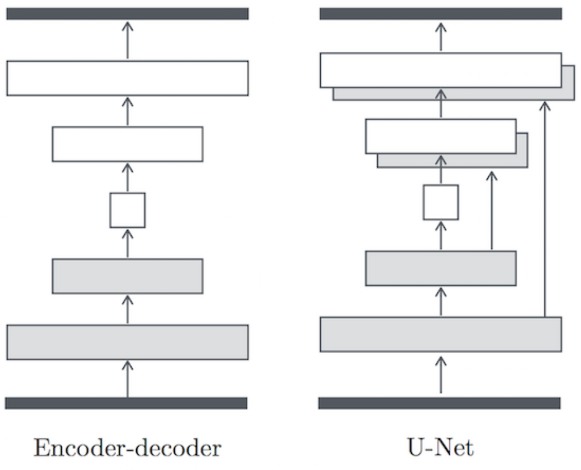


Figure 5.2: The U-Net encoder-decoder architecture.

the case of image restoration tasks and particularly for image denoising it is disputable. First, as noticed at the beginning of this chapter, the key objective of image restoration is to improve the visual quality from a human observer's point of view and the $L2-$norm is clearly not correlated with this desirable objective. Second, it is known that the Euclidean metric is optimal when white Gaussian noise is encountered, but for other noise schemes alternative metrics should be considered [126]. Therefore, a loss function that is based on a metric reflecting the visual quality should be investigated.

Such an investigation can be found in [127], a paper in which the authors compared several losses considering two state-of-the-art metrics for image quality: the Structural SIMilarity (SSIM) index [128] and the MultiScale Structural SIMilarity (MS-SSIM) index [129]. They compared both norms, SSIM, MS-SSIM, and their own loss function that is a combination of MS-SSIM and $L1-$norm on different image restoration tasks, among which joint denoising and demosaicking of color image patches (31×31 pixels) using a FCN of three layers with PReLU activation in the first two ones [130]. We independently came up with the same idea to investigate a loss function that combines the losses $\mathcal{L}^{L1}$ and $\mathcal{L}^{SSIM}$, denoted by $\mathcal{L}^{L1+SSIM}$ in the following. Note that the work presented in [127] focuses on the analysis of loss functions and not on the design of a FCN for image denoising.

Formally, $\mathcal{L}^{L1}$ and $\mathcal{L}^{SSIM}$ losses are, respectively, defined by:

$$\mathcal{L}^{L1}(x,y) = \frac{1}{|x|}\sum_{p\in x}|x-y| \tag{5.3}$$

and

$$\mathcal{L}^{SSIM}(x,y) = 1 - \frac{1}{|x|}\sum_{p\in x}\frac{2\mu_x\mu_y + C_1}{\mu_x^2+\mu_y^2+C_1}\cdot\frac{2\sigma_{xy}+C_2}{\sigma_x^2+\sigma_y^2+C_2}, \tag{5.4}$$

where x is the noisy version of the clean image y and μ, σ, are means and standard deviations that depend on pixel p. Both are computed using a Gaussian filter with standard deviation σ_G. $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$ are two constants, where L is the dynamic range of the pixel values (1 in our case due to normalization in $[0;1]$), $K_1 \ll 1$ and $K_2 \ll 1$. In fact, the SSIM index is a similarity measure that combines three comparison functions measuring different kinds of changes between images: luminance, contrast, and structure, but thanks to a simplification it can be expressed as a product of two terms.

5.3/ Experimental results

5.3.1/ Data set and network training

For image denoising, images from the Berkeley Segmentation Data Set (BSD or BSDS)[1] [131, 132] are widely used for training and testing. For example, in [133] they used the 300 images from BSDS300 to generate patches for training and 200 images for testing (the 200 fresh images from BSDS500). Similarly, Liu *et al.* [134] evaluated the denoising performance of different methods, in comparison with their proposal called Wide Inference Networks (WIN), on the same test set called BSD200. In [118], Zhang *et al.* followed [135] and hence trained their image denoising model DnCNN using 400 BSD images considering three different noise levels. More precisely, for each noise level they cropped

[1] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

$128 \times 1,600$ patches of size 40×40 from this set of 180×180 pixels images. In [122], they used a similar approach to train FFDNet for AWGN denoising in grayscale images, but with a larger patch size (70×70). For performance evaluation they considered images from two sets: 68 natural images from BSD data set and a collection of 12 images, respectively named BSD68 and Set12. Apart from Set12, smaller and larger collections of 5 and 14 images are also sometimes chosen. PSNR results of BSD68 data set are usually given and thus allow to compare different approaches between each other.

However, it is admitted that the training of deep networks can benefit from a large data set and therefore the question of extending the routinely used small BSD training set arises. Hence, in their most recent works [121, 122], Zhang *et al.* not only considered 400 images from BSD, but also selected 400 images from the validation set of ImageNet database and $4,744$ images of Waterloo Exploration Database [136]. According to their experimental study in [121], the training with an enlarged data set of almost $5,500$ images does not improve the PSNR results on BSD68, despite a slight improvement observed for other testing images. In a first evaluation we do not consider the BSD data set, neither for training, nor for testing. We rather use the same images than for the image steganalysis problem described in the previous chapter. Thus, we have used as data set a subset of the $10,000$ gray images of 512×512 pixels provided by the BOSS database [85] and we will only evaluate the proposed approach for additive white Gaussian and speckle denoising. Actually, the first $3,000$ images of the database are used, with $2,800$ images for training and the 200 remaining ones for testing. Let us emphasize that the results we present are preliminaries whose objective is to show in a first step the validity of our proposal.

The proposed network is trained during 50 epochs for a specific type of noise, using the loss function $\mathcal{L}^{L1+SSIM}$ and the Adaptive moment estimation (Adam) optimizer [137] that computes adaptive learning rates for each parameter. We have replaced the traditional stochastic gradient descent (SGD) by Adam due to the observation of Mao *et al.* [133] that it provides a faster training convergence of their encoder-decoder networks. Contrary to the CNN trained in the previous chapter for image steganalysis, up to now the optimization process is not iterated over mini-batches because the tuning of the mini-batch size has not been completed. The computations are still completed on a NVIDIA Tesla Titan X GPU and a typical training for a given noise level takes about 5 hours.

5.3.2/ Denoising performance

A measure used to assess the denoising performance of an approach is the Peak Signal-to-Noise Ratio (PSNR):

$$PSNR(y, x) = 10 \cdot \log_{10}\left(\frac{255^2}{mse^2}\right) \tag{5.5}$$

where x is the noisy observation of the clean image y and mse^2 is the mean squarred error. Let us remark that PSNR (dB) is a standard performance indicator even if it is known to be a poor quality metric when the purpose is to compare the images as perceived by the human visual system. Indeed, as demonstrated in [128], a high PSNR value and good visual quality do not necessarily go together. PSNR variants reflecting visual perceptual functions exist to overcome this drawback, but it is rather the simultaneous taking into account of PSNR and mean SSIM index which is a good indicator of the visual quality: when both metrics have high values, the quality is regarded as high. Thereafter are presented some experimental results obtained for AWGN removal with different noise levels and speckle noise removal.

Table 5.1: Average PSNR (dB) / SSIM obtained for AWGN and speckle according to the chosen denoiser.

AWGN			$\mathcal{L}^{L1+SSIM}$	
σ	Noisy input images	BM3D	Encoder decoder	DnCNN
10	28.37 / 0.5798	36.97 / 0.9282	36.07 / 0.9273	37.23 / 0.9304
30	19.17 / 0.2002	31.02 / 0.8284	32.06 / 0.8626	31.11 / 0.8334
50	15.10 / 0.1052	27.56 / 0.7591	29.97 / 0.8181	27.45 / 0.7613
70	12.64 / 0.0664	24.97 / 0.7091	28.48 / 0.7865	24.55 / 0.7041
80	11.69 / 0.0545	23.76 / 0.6882	27.99 / 0.7743	
Speckle $L = 1$	10.24 / 0.1441		27.86 / 0.7852	

Quantitative results

Table 5.1 shows the quantitative results gained for AWGN, including noise levels $\sigma \in \{10, 30, 50, 70, 80\}$, and speckle reduction on the test set of 200 images. The speckle noise is modeled as a multiplicative noise that follows a Gamma distribution $\Gamma(L, 1)$ of unit mean and variance $\frac{1}{L}$, where $L = 1$. In each case are given the average PSNR and SSIM values of the noisy input images, the corresponding outcomes produced by BM3D, and those issued by the proposed encoder-decoder and DnCNN. The results of DnCNN have been obtained by using directly network models provided by the authors in the `GitHub`[2] of their Matlab implementation. Fourteen models are available, each trained for a precise noise level (the values of σ are regularly distributed in $]10; 75]$). For the speckle case, the BM3D values have not been computed since the corresponding SAR version of BM3D should have been used, while DnCNN is dropped due its focus on Gaussian denoising. As can be seen, the encoder-decoder can achieve satisfactory denoising results and outperforms almost systematically BM3D and DnCNN. Indeed, except for AWGN with $\sigma = 10$, in which case the encoder-decoder gives high values but slightly lower than those of BM3D and DnCNN, better PSNR and SSIM results are obtained thanks to the proposed deep neural network. Moreover, the more noisy the images, the more advantageous it is to use the encoder-decoder to recover clean images. Finally, we can also notice that the neural network is able to deal with AWGN and speckle noise, once trained on the targeted noise, an important feature which is looked for in the perspective of blind denoising. Overall, the residual learning strategy adopted by DnCNN seems interesting for low noise levels, but as the noise increases the reconstruction of a clean image as performed by the proposed network is clearly more appropriate.

To further highlight the suitability of the encoder-decoder, it might be interesting to have an idea of the denoising performance given by other denoising methods. Therefore, in Table 5.2 are shown the behaviors observed by Zhang *et al.* on BSD68 set [122] in the case of AWGN for BM3D [113], WNNM [114], MLP [115], DnCNN, and FFDNet. It can be seen that both deep networks, DnCNN and FFDNet, outperform the other methods. Furthermore, interestingly, as noticed in Section 5.1, DnCNN is only better than FFDNet for low noise levels ($\sigma \leq 25$), a behavior similar to the one shown when compared to the proposed encoder-decoder. Even if these results are obtained on a different data set, considering the performances of BM3D and DnCNN in both tables as reference,

[2]https://github.com/cszn/DnCNN

Table 5.2: Average PSNR (dB) obtained by Zhang *et al.* [122] for AWGN on BSD68 according to the chosen denoiser.

AWGN σ	BM3D	WNNM	MLP	DnCNN	FFDNet
15	31.07	31.37	-	31.72	31.63
25	28.57	28.83	28.96	29.23	29.19
35	27.08	27.30	27.50	27.69	27.73
50	25.62	25.87	26.03	26.23	26.29
75	24.21	24.40	24.59	24.64	24.79

the proposed encoder-decoder appears at a first glance as a valuable competitor for state-of-the-art denoising approaches. Obviously, further experiments and particularly an evaluation on BSD68 are required to confirm this feeling.

In addition to the previous experiments, we have also completed a preliminary evaluation of the encoder-decoder blind denoising ability for AWGN with unknown noise level. To train the network, the first thousands image from data set are used, where for each image its corresponding noisy versions with $\sigma = 10, 30, 50$, and 70 are computed. Thus, the training data set consists of images corrupted with four different noise levels and has a size of $4,000$ images. As the training set size is increased by 43%, a similar increase is obtained in the computation time taken to complete the training that get from 5 hours to 7.2 hours. This AWGN blind denoiser yields the following results for noise level $\sigma = 80$: a PSNR of 27.50 dB and 0.7564 for SSIM value. Obviously, these values are less than the ones obtained with the network trained specifically for $\sigma = 80$ shown in Table 5.1. But they are slightly better than those given by the network trained only for $\sigma = 70$: a PSNR of 27.33 dB and 0.7542 for SSIM value. These results are encouraging but a deeper investigation is needed to confirm that the proposed network can be suitably trained to deal simultaneously with different unknown noise levels.

Visual results

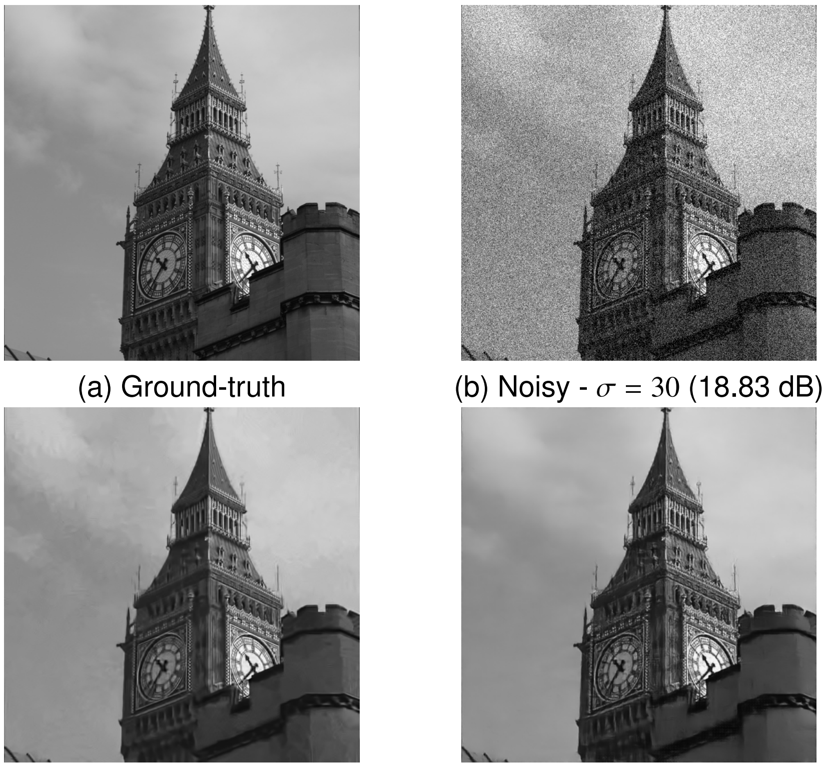
Figures 5.3-5.4 illustrate the visual results of BM3D and the proposed deep network, considering a same image, for AWGN with noise levels $\sigma = 30$ and 70, respectively. It can be seen that the encoder-decoder preserves sharp edges and finer details as the noise level increases. This point is clearly highlighted through the comparison of Figures 5.4(c) and 5.4(d), since the clock on the left shaded part of Big Ben appears far more blurry with BM3D. Furthermore, even if for noise level $\sigma = 30$ the PSNR result is better for BM3D, the neural network yields a image with a better visual quality: a look on the cloudy upper left part in the images (Figures 5.3(c) and 5.3(d)) is convincing in our opinion. This observation is further supported by the SSIM value which is equal to 0.9021 for the clean image recovered by the encoder-decoder, whereas the one for BM3D is equal to 0.8929.

Figure 5.5 shows the denoising results on two different images with noise level 70 given by the encoder-decoder and DnCNN. In both cases the proposed network recovers a clean image with far more details and a better visual quality. This is again confirmed by the higher values of PSNR and SSIM: for Big Ben the values obtained from the image recovered by the encoder-decoder are, respectively, 25.01 dB and 0.8204 *versus* 24.11 dB and 0.7875 for DnCNN, while for the image with the bird they are 21.86 dB and 0.7162

versus 21.24 dB and 0.6547. In the case of the speckle noise presented in Figure 5.6, despite the huge corruption interesting details are bring out, especially in the shaded part of the building.

5.4/ Conclusion

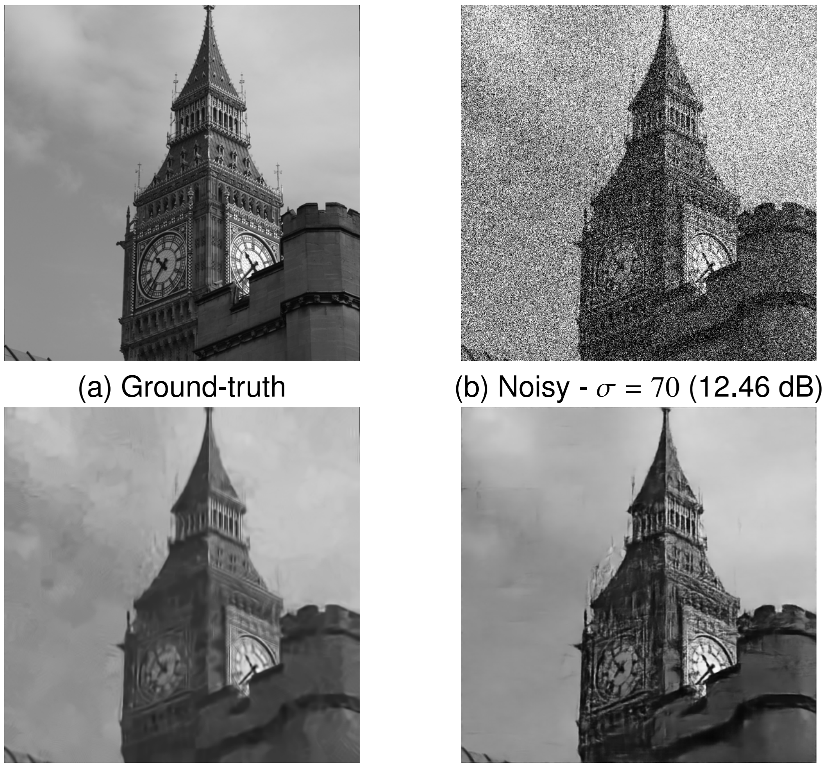
In this chapter, a fully convolutional network that consists in an encoder-decoder with skip connections has been proposed for image denoising. The great lines of the network have been presented and the choice of the loss function used to carry out the training discussed. The ultimate goal is to have an effective deep blind denoiser, able to handle different kinds of noise and without any prior knowledge on the noise level. Even if this work is still in progress, the first preliminary results showed that the network can remove additive white Gaussian noise and speckle noise, provided that it is suitably trained for the targeted noise. While we have demonstrated promising results, further experiments are needed to go towards fully blind denoising and to compare it with other approaches.



(a) Ground-truth (b) Noisy - $\sigma = 30$ (18.83 dB)

(c) BM3D (29.54 dB) (d) Proposed deep network (29.06 dB)

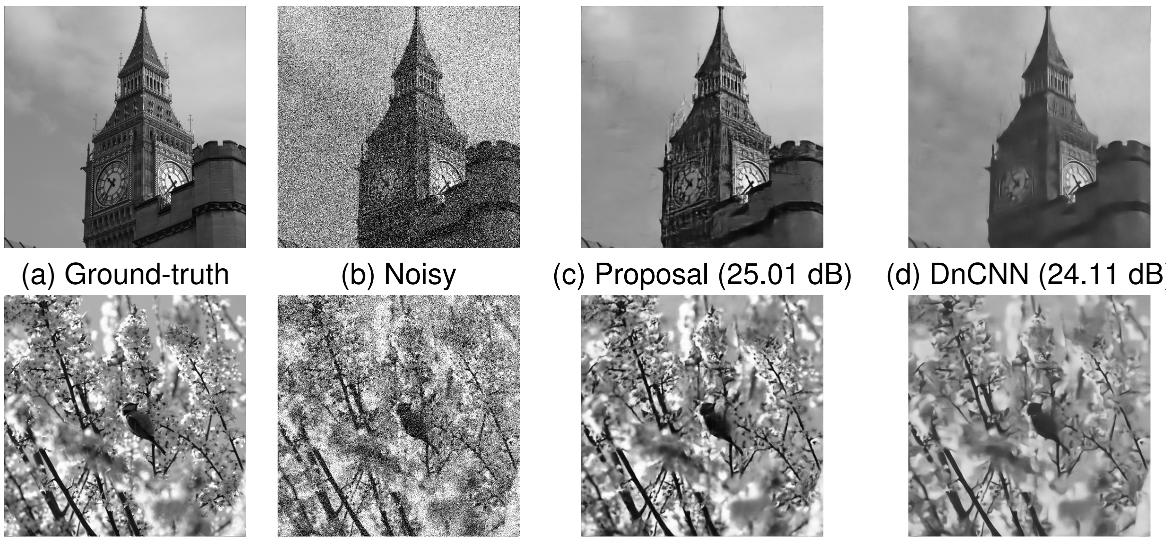
Figure 5.3: AWGN denoising results (PSNR) of one image with noise level $\sigma = 30$.



(a) Ground-truth (b) Noisy - $\sigma = 70$ (12.46 dB)

(c) BM3D (24.36 dB) (d) Proposed deep network (25.01 dB)

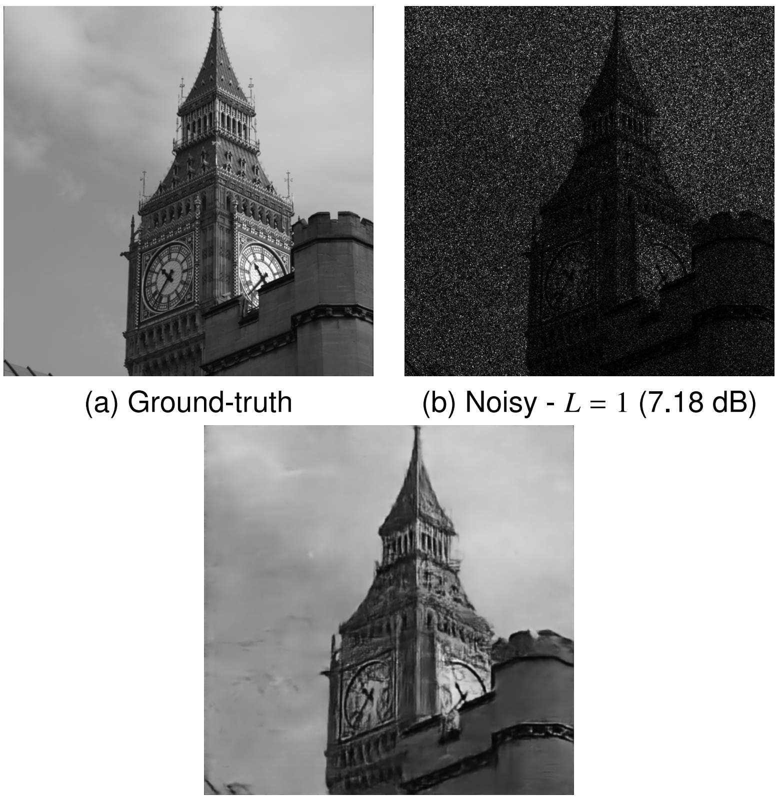
Figure 5.4: AWGN denoising results (PSNR) of one image with noise level $\sigma = 70$.



(a) Ground-truth (b) Noisy (c) Proposal (25.01 dB) (d) DnCNN (24.11 dB)

(e) Ground-truth (f) Noisy (g) Proposal (21.86 dB) (h) DnCNN (21.24 dB)

Figure 5.5: AWGN denoising results (PSNR) of two images with noise level $\sigma = 70$.



(a) Ground-truth (b) Noisy - $L = 1$ (7.18 dB)

(c) Proposed deep network (24.54 dB)

Figure 5.6: Speckle denoising results (PSNR) of one image with $L = 1$.

6

Echo State Networks-based Reservoir Computing for MNIST Handwritten Digits Recognition

In this chapter we will present an application of Reservoir Computing (RC), a brain-inspired paradigm quite different from learning point of view. Two considerations motivated this work. First, we wanted to study a concept which has been implemented in photonic hardware by colleagues of the OPTICS department from the FEMTO-ST Institute and, second, to apply it to a computer vision problem. Therefore, in collaboration with Raphaël Couturier we decided to co-supervise a MSc student in order to investigate the ability of Echo State Networks (ESN), a machine learning implementation of Reservoir Computing, to classify the digits of the MNIST database.

This work has led to a conference paper [138] which is summarized in this chapter. In a first part (Section 6.1) we give a short introduction to Reservoir Computing, then the two next sections describe, respectively, the ESN implementation of RC (Section 6.2) and the study we made on the classification of digits with it (Section 6.3). Finally we conclude in Section 6.4. Even if the studied ESNs are not able to outperform state-of-the-art convolutional networks, we show that they can reach low error thanks to a suitable preprocessing of images and use of the reservoir.

6.1/ Introduction

In Part I and in the previous chapters we have presented research works considering almost exclusively feedforward neural networks, but such networks are not designed to naturally deal with random variables whose samples are not independently and identically distributed. For tasks that exhibit a temporal dynamics within the data, such as speech recognition or time series forecasting (for example financial market time ones), Recurrent Neural Networks (RNN) are more suited to capture temporal correlations thanks to their memory capacity. Unfortunately, even if different approaches are available for training standard RNNs, like the gradient-based method BackPropagation Through Time (BPTT) which unfolds the recurrent network in time into a feedforward one trained using the back-propagation process, training is usually a difficult task. In that case the recurrent network can also be seen as a deep network with shared synaptic weights. Among the reasons why the training is hard are stability ones due to bifurcations in the course of learning

[139] and the handling of long-term dependencies as shown by Bengio *et al.* in [140]. In fact, the former work has shown that the longer the temporal span of dependencies the less efficient the gradient descent is, due mainly to vanishing and exploding gradients.

Fortunately, over the past many years several key improvements have led to an increasing use of RNNs with incredible successes in various real applications. A first and major example of architecture enhancement is the kind of RNN called Long-Short Term Memory (LSTM) networks originally proposed by Hochreiter and Schmidhuber [141], which has received a lot of interest leading to different variants (see [142] and [143] for comparative surveys) being now widely used. LSTM networks usually outperform classical RNNs and Hidden Markov Models, being able to learn long-term dependencies using a new structure called a memory cell. The key component is a gating mechanism that is embedded in each memory cell and permits to control the evolution of its content. In comparison with classical RNNs the benefits of this gating mechanism is twofold. First, the gates can prevent modifications for multiple time steps and thus preserve the information and propagate errors for a longer time. Second, by allowing a independent control of the way information flows in, out, and stays in a memory cell, the gates can be trained to focus on some parts of the input signals. Overall, LSTM networks address the problem of vanishing gradients in a way that can be seen similar to residual networks. In this chapter, we study another approach to solve this problem, namely the Reservoir Computing concept [69, 70] which overcomes the problem of vanishing gradients by giving up the training of the neural network's recurrent part. The key idea in RC is to extract features from time series using a high-dimensional dynamical system that is randomly initialized.

Originally, Reservoir Computing referred to Echo State Networks [144, 145], a machine learning method, and to Liquid State Machines (LSM) [146] in computational neuroscience. In the machine learning context, the key idea can be formulated as feeding a large and sparsely connected recurrent network, called the reservoir, with input data to produce nonlinear signals among the neurons, which are then combined linearly together in the output layer to obtain, after training, the expected outputs. The training of the RNN is thus not only easier, since it is focused on the output layer, but also because it results in solving a system of linear equations. Indeed, the projection in a higher-dimensional space done by the reservoir is supposed to give such a rich representation of the history of the inputs that a simple linear readout is sufficient to obtain relevant outputs after training. More recently, the effectiveness of deep networks with the rise of deep learning has resulted in deep reservoir computing, which consists in the stacking of several reservoirs. A deepESN model has been thus introduced in [147, 148] and its ability to perform a hierarchical processing of temporal data studied. The extension of the Echo State Property, the key property in ESNs, has been particularly discussed in this context.

Another great interest of RC is the possibility of a hardware implementation, providing thus a platform for neuromorphic computing [149]. In fact, such neuromorphic processors are very appealing because they can reduce the power consumption and process the input data at high-speed. Different physical systems implementing reservoir computing have been already proposed, from concept proof one which uses water ripples [150] to more evolved platforms using optoelectronic [151, 152] and fully optical devices [153]. A review of the two main approaches for the design of reservoir computer in optics, namely spatially extended photonic networks and the delay-line based RC, is made in [154].

We have started our research works on RC by investigating a delay-line approach, namely the optoelectronic implementation proposed by Larger *et al.* in [151], and more particularly its computer simulation with an Ikeda type nonlinear delay differential equation. The result of this first work, a parallel version using the Message Passing Interface for the communication, allowed to obtain the same classification performance on the considered speech recognition problem but with a reduced computation time. In this chapter we will not develop this first work, but rather present another one that studies the use of ESNs for the classification of images representing handwritten digits, and thus show that they can fulfill computer vision tasks.

6.2/ Echo State Networks-based reservoir computing

Let us start with the description of the ESN model and then briefly discuss the Echo State Property (ESP), which is the property having a major impact on the model accuracy.

6.2.1/ Echo State Network model

The original ESN basic model has been proposed by Herbert Jaeger in [144]. Formally, the update equation of the reservoir's state vector x_t for an input signal u_t, which is a N_u-dimensional vector, is typically defined by:

$$x_t = f\left(\overset{in}{W} u_t + W x_{t-1}\right), \qquad t = 1, ..., T \tag{6.1}$$

where $x_t \in \mathbb{R}^{N_x}$ denotes a vector of reservoir neuron activations at discrete time t for a reservoir consisting of N_x interconnected neurons, f is a Lipschitz continuous sigmoidal function, while $\overset{in}{W} \in \mathbb{R}^{N_x \times N_u}$ and $W \in \mathbb{R}^{N_x \times N_x}$ are the input and reservoir weight matrices respectively. The reservoir usually starts with a null state ($x_0 = 0$). The output vector, or readout, $\hat{y}_t$ is then obtained thanks to a linear combination of neuron activations:

$$\hat{y}_t = g\left(\overset{out}{W} x_t\right), \tag{6.2}$$

where $\overset{out}{W} \in \mathbb{R}^{N_y \times N_x}$, with N_y the number of readouts, is the output weight matrix and g is the output activation function (usually the identity or a sigmoid). This basic model can be extended to include bias weights and feedback between the output signal and the reservoir, as follows:

$$x_t = f\left(\overset{in}{W} u_t + W x_{t-1} + \overset{ofb}{W} y_{t-1} + \overset{bias}{W}\right), \qquad t = 1, ..., T \tag{6.3}$$

where $\overset{ofb}{W} \in \mathbb{R}^{N_x \times N_y}$ is the weight matrix of the connections between the outputs and the reservoir's neurons. Feedback connections are used in a mode called *free-run* where the ESN is used to generate an output signal one step ahead. The matrix $\overset{bias}{W} \in \mathbb{R}^{N_x}$ represents the biases of the reservoir neurons.

Skip connections, which skip the reservoir and connect directly the inputs to the outputs, can be added, leading to the following expression for the readout:

$$\hat{y}_t = h\left(\overset{out}{W} [x_t; u_t]\right), \text{ which becomes for linear outputs } \hat{y}_t = \overset{out}{W} [x_t; u_t]. \tag{6.4}$$

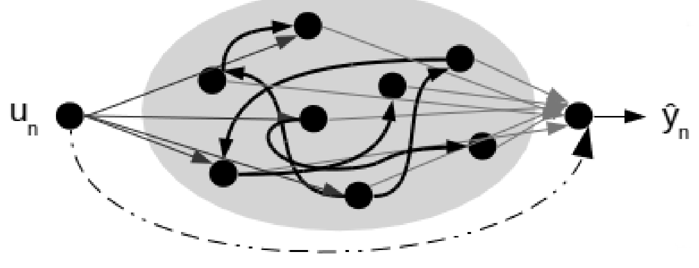


Figure 6.1: ESN architecture: the inputs u_t are connected to a recurrent network of sigmoid neurons, while the output vector $\hat{y}_t$ is a combination of these neurons.

The output weight matrix is then $\overset{out}{W} \in \mathbb{R}^{N_y \times (N_x + N_u)}$ and $[\cdot\,;\cdot\,]$ stands for a vertical vector concatenation. Figure 6.1 gives an architecture overview of the ESN basic model, showing that the input signal is first projected in a space of higher dimension thanks to the reservoir ($N_u << N_x$) from which the outputs are then extracted. Given an input signal, the reservoir's role is thus twofold: perform a nonlinear expansion in a higher dimensional space, and a memory.

A model named Leaky-integrator ESN (Li-ESN) has been proposed later in [145], in order to adapt the network to the temporal characteristics of the target signal y_t. In this model, which is popular due to the good results obtained in practice, each neuron takes into account its weighted past state and has thus a more smooth state evolution. The dynamic of a Li-ESN in continuous time is defined by:

$$\dot{x} = \frac{1}{c}\left(-ax + f\left(\overset{in}{W} u + Wx + \overset{ofb}{W} y\right)\right), \tag{6.5}$$

where $c > 0$ is a time constant shared by all neurons, $a > 0$ is the leaking rate of the neurons, and f is a sigmoidal function (typically $\tanh(\cdot)$). Then, if we discretize the previous equation with a time step δ and a sampled input signal $u_{t\delta}$, the reservoir state is governed by the following recurrent equation:

$$x_{t+1} = \left(1 - \frac{a\delta}{c}\right)x_t + \frac{\delta}{c} f\left(\overset{in}{W} u_{((t+1)\delta)} + Wx_t + \overset{ofb}{W} y_t\right) \tag{6.6}$$

and by setting $\delta = c$ we obtain finally the Li-ESN reservoir update equation:

$$x_{t+1} = (1-a)x_t + f\left(\overset{in}{W} u_{t+1} + Wx_t + \overset{ofb}{W} y_t\right), \tag{6.7}$$

where $a > 0$ is the leaking rate. This parameter allows to control the feedback of past states and the reservoir's dynamics (the lower the value of a, the slower the dynamics).

The ease of training, which is to minimize the squared error between the output $\hat{y}_t$ and the target signal y_t, comes from the fact that it consists in the computation of a linear regression on the weight matrix $\overset{out}{W}$. Therefore, with $X \in \mathbb{R}^{N_x \times T}$ the matrix containing all reservoir states during training phase, and $Y \in \mathbb{R}^{N_y \times T}$ the matrix containing the corresponding outputs (with $N_x < T$), the training is defined by the following equation:

$$\overset{out}{W} X = Y. \tag{6.8}$$

To compute $\overset{out}{W}$, Ridge Regression (or Tikhonov regularization) [155] is commonly used in order to bound the amplitude of output weights, and thus reduce the sensibility to noise and overfitting [156]:

$$\overset{out}{W} = YX^T(XX^T + \beta I)^{-1} \tag{6.9}$$

where β is the regularization coefficient and I is the identity matrix. The optimal value for β depends on the reservoir instance and the training data. Another way to solve Equation (6.8), which is frequently chosen (its offers in particular high numerical stability), is to use the Moore-Penrose pseudoinverse X^+:

$$\overset{out}{W} = YX^+. \tag{6.10}$$

The two previous approaches to calculate $\overset{out}{W}$ are used in a offline training context. That means that the training is done in one shot by concatenating in matrix X reservoir states and in Y the associated target outputs for an input signal u_t.

Considering the previous model, the Reservoir Computing method with ESN given in [144] consists of the following steps:

1. Generate randomly an input matrix and large reservoir ($\overset{in}{W}$,W,a);
2. Compute reservoir states for each time step using the input signal u_t and save them successively in matrix X;
3. Compute the output matrix $\overset{out}{W}$ from states collected in step **2** with a linear regression method, minimizing the mean squared error (MSE) $E(y_t, \hat{y}_t)$ between the output signal $\hat{y}_t$ and the target signal y_t;
4. Use the trained network on a new input signal u_t and computes the output $\hat{y}_t$ with the $\overset{out}{W}$ output matrix.

For a given task, several parameters have an impact on the ESN performance: the reservoir size, its sparsity, and so on. An important parameter is the reservoir spectral radius, because it was first used to define a necessary condition ensuring the so-called Echo State Property in most of the situations. Indeed, the ESP which can be formulated as: past states x_t and past inputs u_t should both affect the future states x_{t+k} in a way that is gradually vanishing with k, should be satisfied to have a working ESN-based RC method. In fact, the state of the reservoir should be uniquely defined by the fading history of the input. In other words, for a long enough input, the reservoir state should not depend on the initial conditions that were before the input. Intuitively, one also understands that two close input signals must not lead to very different reservoir states, otherwise the ESN would be useless.

6.2.2/ Echo State Property

As said above, an ESN should have the ESP in order to provide suitable performance. Therefore, it is important to have, for a given ESN setup, a practical way to estimate if the ESP holds. Several research works have investigated the formulation of conditions to get the ESP, for example in [70] and more recently in [157], leading to a sufficient and a necessary condition related to ESP [158]:

- *Sufficient condition*: if $\sigma(W) < 1$, where $\sigma(W) = \rho(WW^T)^{\frac{1}{2}}$ is the largest singular value of W, then the ESP is satisfied for every input;
- *Necessary condition*: $\rho(W) \leq 1$ is necessary to have the ESP for all inputs.

Even if the necessary condition only applies to autonomous systems that contain the zero input sequence ($u_t = 0$), in practice the reservoir weight matrix W is usually generated so that it fulfills this condition as follows:

1. Generate a random matrix W;
2. Divide W by the spectral radius $\rho(W)$.

Sometimes to improve the ESN performance the reservoir matrix must be set such that its spectral radius is above 1, which means that the ESP is no more ensured for all inputs. Even if no method to create an optimal reservoir for a given task is currently available, some conditions and measures have been proposed, most of them based on the border of chaos and the Lyapunov exponent [157].

The border, or edge, of chaos, is a region of parameters of a dynamical system in which the neural network operates at the limit between ordered and chaotic behaviors. There is no proof that being between ordered and chaotic regimes is always the best choice, but since it has been shown that for some applications requiring a lot of memory it is the optimal regime for a RNN [159], a common method is to set W so that its spectral radius is slightly below one. Nonetheless, setting systematically W such that the reservoir dynamics is at the border of chaos is questionable, because ESNs with small spectral radius can also perform well.

6.3/ Application on the MNIST problem

The MNIST problem is classically used as a benchmark for evaluating the performance of a classifier in the field of machine learning. It is a classification problem where the objective is to recognize a handwritten digit 0 to 9 (problem with ten classes) in a small grayscale image. The data set consists of 60,000 training images and 10,000 testing images, with the corresponding label data. Currently, the best-known classifier is a committee of 5 convolutional neural networks [64] (final prediction obtained by averaging the output probabilities of the networks) with a final error rate of 0.21%. Each CNN is composed of two convolutional layers having, respectively, 32 and 64 feature maps, followed by a fully connected layer of 150 ReLU neurons sparsely connected with DropConnect, while the output prediction vector is given by a softmax classification layer.

In the following subsections we first present how the network is fed with an input image to be classified. Then, different image transformations and the classical output layer are described. Finally, the classification performance given by several reservoir setups are compared between each other and with other machine learning approaches.

6.3.1/ Encoding images as temporal series

Each image of 28×28 pixels is first normalized by scaling the pixel values to $[-1, 1]$, leading to the matrix $I_n \in \mathbb{R}^{N_i \times N_i}$ where n is the image index and $N_i = 28$ is its size. Then, as

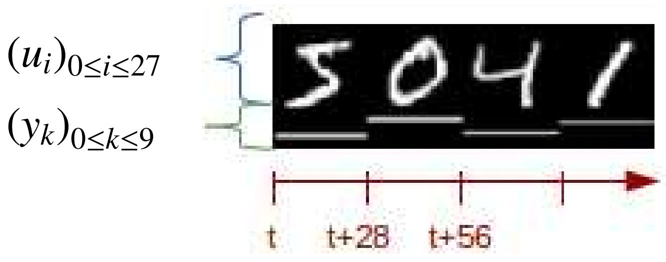


Figure 6.2: Inputs and outputs of a ESN for image classification, with $N_i = 28$.

an ESN processes time series, we must transform such a image into a temporal signal. Therefore, a spatial dimension is converted to a temporal one, passing thus images column by column or row by row according to the chosen spatial dimension. Considering a column-wise feeding, or horizontal scanning, we have $N_u = N_i$ inputs connected to the reservoir and each input satisfies $(u_i)_t = (I_n)_{ij}$, where $t = n \times N_i + j$. That means that the i-th reservoir input receives, at time step t, the value of the (i, j) element of the array I_n representing the pixel in position (i, j). The underlying idea is to store an image in the reservoir one column after another, provided that the reservoir has enough memory for the complete image.

An image is thus seen as a set of N_i time series of length N_i, and the images are presented one after another. The output binary vector y, which identifies the represented digit by a single one in the vector component having for index the digit value, is transformed in a similar way in ten time series of length N_i. Figure 6.2 illustrates this temporal encoding of an image ($u_t = (u_i)_t$, with $0 \leq i \leq 27$) and its associated label ($y_t = (y_k)_t$, with $0 \leq k \leq 9$), we can see that the network will start to process an image every N_i time steps.

6.3.2/ Image transformations

A way to improve the classification performance is to increase the quality of the input data and therefore two kinds of transformations are introduced. The first one, denoted by S_x, changes the original image to a new one of $x \times x$ pixels focused on the digit. It removes most of the background around the digit and then changes its size while preserving its aspect ratio. The second transformation, denoted by R_θ, is a centered counterclockwise rotation of angle θ of the digit without changing the image size. The idea is to provide the reservoir with a different point of view of the digit and thus to increase the input features associated to a digit value.

The two transformations can be combined to produce a new image and rather than a single image, the reservoir can receive several points of view of the same original image by stacking transformed images. To assess the relevance of these transformations, their combination and stacking, in addition to the configuration which uses the original image, we have defined four other input configurations:

- Configuration A corresponds to a single image issued by S_{15};
- Configuration B is A plus the image resulting of $S_{15} + R_{30}$;
- Configuration C is obtained by adding $S_{15} + R_{90}$ to B;
- Configuration D consists of the 3 images from C and $S_{15} + R_{150}$.

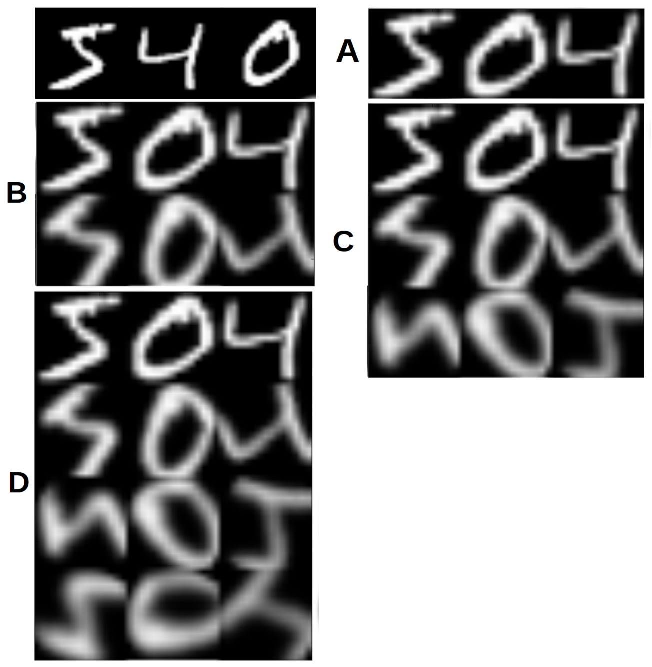


Figure 6.3: Examples of image deformations and stacking.

On the one hand the original size of 28×28 pixels of the images is reduced to 15×15 pixels and on the other hand the number of temporal series defining a digit instance will range from 15 for configuration A up to 60 with configuration D.

As the dynamics induced by each configuration should be different, in each case the best classification performance should be observed by using specific combination of spectral radius and leaking rate values. As we expect to see an increase of the dynamics with the transformations, a higher leaking rate should give better results.

6.3.3/ Output layer

The output $\hat{y}_t$ is composed of 10 outputs $(\hat{y}_k)_t$ $(0 \leq k \leq 9)$, one per possible digit value, which are trained to match the target label y_t ones, with $(y_k)_t = 1$ if the digit corresponding to the input image I_n is k and 0 otherwise. In the case of the classical output layer, the final prediction for I_n is obtained by taking the index of the output having the highest averaged value over the time duration of the image. In other words, I_n is the digit d satisfying:

$$\overline{\hat{y}_d} = \max \left\{ \overline{\hat{y}_k} \;\middle|\; 0 \leq k \leq 9 \right\} = \max \left\{ \frac{1}{N_i} \sum_{t=t_{begin}}^{t_{end}} (\hat{y}_k)_t \;\middle|\; 0 \leq k \leq 9 \right\}. \tag{6.11}$$

6.3.4/ Experimental results

A suitable parameter setup for the classical Li-ESN was found after some preliminary experiments, leading to a spectral radius of 1.3, a proportion of zeros in $\overset{in}{W}$ of 90% and 10% in W, an input scaling of 0.6, bias values equal to 1.0, and a leaking rate of 0.2. Notice the high spectral value, above one, and the low leaking rate showing that this

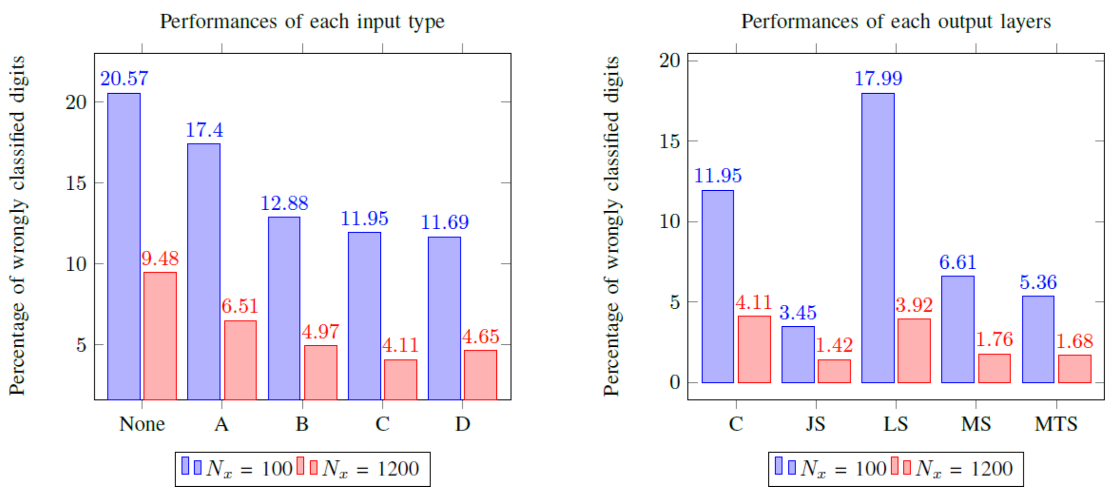


Figure 6.4: Error rate of different input and output configurations for reservoirs of 100 and 1,200 neurons.

task requires a low dynamics. The classification performances presented thereafter for two reservoir sizes ($N_x = 100$ and $1,200$ neurons) are obtained by averaging the results obtained from 30 Li-ESN for each configuration.

The left part of Figure 6.4 presents the performance of each input configuration, while the right part is devoted to the impact of the reservoir states used to compute the output considering the best input configuration. In this last case, apart from **C**lassical output layer, the output configurations range from considering as a single state all the reservoir states (**J**oin **S**tates) from $x_{t_{begin}}$ to $x_{t_{end}}$, a concatenation of two (**M**ixed **S**tates) or three (**M**ixed **T**hree **S**tates) states, to only the final state (**L**ast **S**tate). Besides the clear impact of the reservoir size, two remarks can be made. First, we can see that the proposed transformations allow to obtain better performances, with input configuration C which is the best choice. As expected, we have found different optimal spectral radius values for each configuration (from 1.2 for A to 0.1 for D) and a higher leaking rate $a = 0.4$ than for the classical Li-ESN. Second, the strategy used to extract the outputs of the reservoir by combining all the states (JS) is also beneficial. Regarding the computation time, for the 1,200 neurons reservoir the training time is about 20 minutes on a typical personal computer.

Table 6.1 shows a comparison with other machine learning approaches. The ESNs we have investigated (denoted in bold in the table) are not able to outperform the state-of-the-art CNN, but compared to earlier deep learning approaches [160] the classification performances are fair. First, the reservoir of 1,200 neurons gives a lower error rate than LeNet-1: the ESN yields 1.42% while it is 1.7% for LeNet-1. Second, a large ESN instance using 4,000 neurons has an error rate of 0.93%, slightly lower than that of LeNet-4 and LeNet-5 [66]. Obviously, the generic architecture of RC, which allows to apply it easily to very different tasks, explains that a network architecture suited for computer vision tasks as CNNs are one, offer better results.

Like for other classifiers, a committee of ESNs is a fast and common way to improve performance. Besides, the three places on the podium are occupied by committees of 7, 35, and 5 CNNs, yielding respective error rates of 0.27%, 0.23%, and 0.21%. This last value is the best performance so far and is nearly the human capacity ($\approx 0.2\%$). However,

Table 6.1: Comparison of different machine learning approaches.

Classifier	Error rate	Add/mult
LeNet-1 (deep learning)	1.7%	1.6×10^5
Li-ESN with MTS 1,200 neurons	**1.68%**	2.5×10^5
Li-ESN with JS 1,200 neurons	**1.42%**	4.5×10^5
21 × Li-ESN with JS 1,000 neurons	**1.25%**	15×10^6
SVM degree 4	1.1%	14×10^6
LeNet-4	1.1%	1.6×10^5
LeNet-5	0.95%	4.0×10^5
Li-ESN 4,000 neurons	**0.93%**	
CNN 551 neurons	0.35%	
7 × CNN 221 neurons	0.27%	
35 × CNN 221 neurons	0.23%	
5 × CNN with DropConnect	0.21%	

despite nowadays GPU implementations, CNNs can have huge training time, stretching from a few hours to several weeks. The ESN model on the contrary has a training time of only about 20 minutes for a reservoir of 1,200 neurons.

Better performances can be achieved with the ESN model by extending the learning set with affine and elastic transformations. Our method also shows an impressive capacity to learn quickly with a small training set. Figure 6.5 shows the error rate for the network using transformations of type C as input and each specific output layer. The error rate drops quickly as the learning set size grows and reaches values near its minimum with only one third of the set. This demonstrates a huge capacity of generalization.

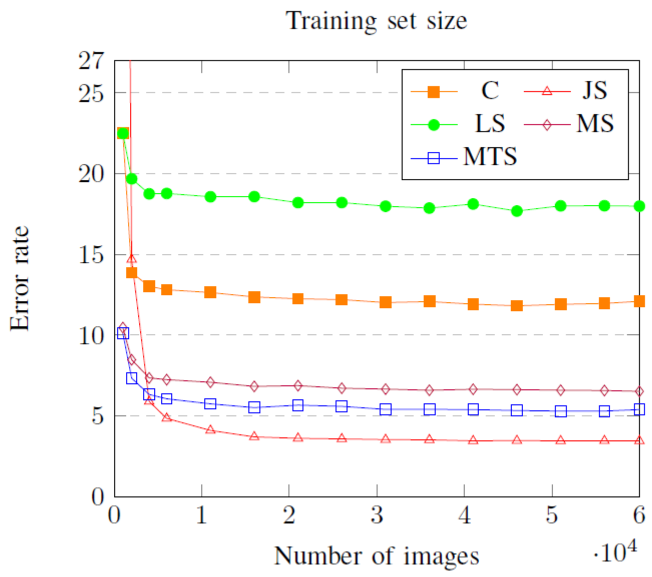


Figure 6.5: Error rate of different input and output configurations according to the learning set size.

6.4/ Conclusion

We have shown that the Reservoir Computing paradigm and more particularly Echo State Networks are able to deal with a computer vision task. In the case of the MNIST problem, this kind of recurrent neural networks, which is interesting due to the ease of training and existing neuromorphic implementations, gives a fair classification performance. To reduce the error rate, two approaches have been investigated with success. First, we have fed the reservoir with different points of view of a same image, in order to increase the features representing the corresponding digit. Second, the way the reservoir is exploited to compute the output can also greatly affect the obtained results. We think that the very generic architecture of ESNs allows an easy and fast use on a wide range of tasks, but as only the output weights are trained for the target task, it might be outperformed by fully optimized approaches like CNNs in the handwritten digits recognition case study.

Conclusion and Future Work

1/ Summary of the main contributions

Today, the neural network field enjoys a resurgence of interest due to the many breakthrough results obtained with deep learning architectures, but more traditional networks are not necessarily outdated. The research works we have presented show that various neural network architectures are able to provide a relevant solution for a wide range of problems, provided that the data set is sufficiently representative to let them generalize.

A shallow feedforward multilayer network is thus perfectly suited for function approximation, assuming that the domain of definition is well-chosen with the handcrafted inputs. Even if this is explained by the universal approximation theorem, the key is to be able to find the optimal network topology and more particularly the setting of the hidden part of the network. The practical cases described in the first part of this manuscript and solved with multilayer perceptron networks clearly support this conclusion.

First, we have designed neural networks for the simulation of lung motion in the context of image-guided radiotherapy. Therefore we have designed MLPs for two different tasks, networks to improve the quality and the size of the initial data set and another one to simulate the motion. Indeed, the data set consisted of anatomical feature points plotted on 4D-CT scans, a technology that captures the location and movement of a body's organs over time, but as the points were in limited number and their movements noisy, we had to increase their number and smooth their trajectories. After this data augmentation, a MLP was able to learn to simulate the motion of anatomical feature points of the lung. However, even if it provides accurate simulations, it does not allow to directly render a 3D view of the lung motion. An additional approach for reconstructing a dynamic 3D model of the lung from the feature points is needed for that.

Second, we have assessed the ability of the multilayer perceptron neural network to give a relevant modelling of the airflow evolution and then we have studied the application of Micro-Electro-Mechanical Systems (MEMS) for aerodynamical active flow control. After having established the relevance of a multilayer perceptron to predict the force corresponding to the flow induced by the MEMS, we have optimized the flow with respect to a numerical criterion. The case study which is considered is a dynamic flow over a backward facing step (a rough model of the back of a car) where MEMS actuators velocities are adjusted to maximize the pressure applied on the step surface and consequently the corresponding force.

Third, we have presented a way to build chaotic neural networks based on a rigorous theoretical framework that allows to model the considered problems as dynamical systems shown to be chaotic according to Devaney's definition. This is a major contribution since in most of the works dealing with chaotic neural networks the proposed networks are usually claimed to be chaotic without any mathematical proof. An extension to the dual case, checking whether an existing neural network is chaotic or not, is outlined. It consists in checking if the graph describing the network evolution is strongly connected or not. A further question we have addressed by studying the predictability of a natural process that exhibits a chaotic dynamics is the practical interest of neural networks in such cases, as they are often the first choice in many research works that simply assume the considered process to be predictable. Thus, after having established the chaotic behavior of protein folding, we have shown that using a MLP to predict the fold is obviously questionable.

In all the aforementioned works, the inputs and the inner hidden part of the multilayer networks have been tuned in order to identify the topology providing the best predictions. For the inputs, the replacement of the initial low dimensional handmade input vector by a Higher-order Process Unit structure with polynomial combinations systematically provided better function approximations. The benefit of using two hidden layers, together with their respective best sizes, has been investigated with a trial and error process. As for the single hidden layer setups an incremental training approach allowed to find the optimal number of hidden neurons. All the networks were trained using a quasi-Netwon method, the L-BFGS algorithm, to optimize the weights.

However, when the input data is unstructured and of high dimension, such as in the case of an image, finding suitable handcrafted features that are compact and discriminant is usually illusory. A main advantage of deep learning over all previous neural networks and other machine learning methods is precisely the ability to extract features from the training set. By searching during the training new ways to represent the input data, it allows also to work with more complex data. To find such features, the data set must be large and representative enough, and the training might take a long time even if increasingly powerful computing platforms are available. Among the different kinds of deep architectures some are known to be specifically suited for a type of problem, but as it is also the case for other neural nets, the question of finding the optimal architecture for a given problem is still open.

Motivated by the number of challenging tasks in computer vision for which deep learning has achieved breakthrough improvements, we have investigated the application of deep networks to two similar problems already studied in our research teams.

The first problem, which takes place in the information security context, is image steganalysis. We have thus studied the Convolutional Neural Network (CNN) architecture as an alternative to the classical steganalysis approach based on Spatial Rich Models (SRM), hand made features, and Ensemble Classifier (EC). In a first attempt we have tried to build a new architecture that offers a better detection performance, but our proposal was only limited to a single key embedding scheme. Then, considering a CNN architecture which, to the best of our knowledge, is the most competitive one, we have examined when this CNN architecture fails to propose a method improving the detection performance for different spatial steganography algorithms. We have been able to define a reliable criterion allowing to decide, for an input image, when to use the CNN or SRM+EC to obtain the most accurate prediction. The experiments done have validated the proposed criterion, since it has always led to improved detection performances, regardless of the studied embedding schemes. Another contribution of this work is to have designed a steganalyzer insensitive to the embedding process (blind detection).

The second problem is image denoising whose aim is to achieve both noise reduction and feature preservation. In many fields of application, image denoising is considered very important to improve visual appearances and facilitate subsequent processing. Medical imaging is a typical example with CT or ultrasound images. Therefore, we have conceived a fully convolutional neural network architecture that belongs to the encoder-decoder family. The first experiments show that the obtained deep network can remove additive white Gaussian noise and speckle noise, once trained for the targeted noise, whereas traditional denoising methods and even existing deep denoising methods usually only focus on additive white Gaussian noise.

The manuscript ends with the description of a work that is quite different from the ones described above. Indeed, Reservoir Computing is an attractive paradigm of recurrent neural network architecture due to the ease of training and existing neuromorphic implementations. It has been successively applied on speech recognition and time series forecasting, but few works have so far studied the behavior of such networks on computer vision tasks. To fill this lack, we have explored the ability of Echo State Networks to classify the digits of the MNIST database. We have shown that even if ESNs are not able to outperform state-of-the-art convolutional networks, they can provide low error thanks to an appropriate preprocessing of images.

2/ Future work

The possible directions for future works are twofold. On the one hand, some of the problems faced in the research works we have presented might be extended. On the other hand, the experience gained in deep learning and the continuous significant progress made in various areas thanks to neural networks encourage us to continue to study their application to new challenging problems. In the following we present some plans for future work in the fields of information security, medical image analysis, and bioinformatics.

2.1/ Information security

The work in the context of information hidding in images is limited to the detection of embedding schemes in the spatial domain, *i.e.* steganography methods that slightly change the value of some well chosen pixels. However, embedding methods have also been developed for the frequency domain, in which case the secret message is embedded in the transformed coefficients (typically issued from discrete cosine, Fourier, or wavelet transforms). As detecting changes in the frequency domain is known to be more difficult than in the spatial one, the design of a CNN-based steganalyzer for this case is obviously more complex. As an example, while CNNs for spatial steganalysis typically have at most five convolutional layers, the deep networks proposed for JPEG steganalysis merge the outputs of several CNNs or use a very deep structure like the ResNet one. Some researchers claim that the CNN architecture is the new state-of-the-art tool for steganalysis, but we think that classical statistical approaches can still bring valuable information. Therefore, as in our proposal for the spatial case, we think that it might be interesting to investigate an approach that mixes deep learning and classical statistical analysis.

Deep learning is also receiving more and more attention in the detection of cyberattacks that are executed to disclose confidential information, alter its integrity, and so on. Indeed, recent high-profile cybersecurity issues have shown that new approaches are needed to face the continuous growth of cyberthreats that can lead to potentially catastrophic, political, economic, and social implications. Moreover, many systems are prone to attacks as they have not been developed with security in mind, such as the ADS-B system that has been designed for the safe navigation of airplanes and air traffic control management. This system allows to continuously and precisely follow aircraft movements thanks to the broadcast of messages containing information like the aircraft number, speed, position, etc. but its security can be easily compromised due to the lack of basic security mechanisms such as authentication, integrity, and encryption. Therefore being able to detect anomalous ADS-B messages is of great importance to mitigate False Data Injection Attacks (FDIA) which represent a major threat. To solve this problem, we plan to work in collaboration with colleagues of the VESONTIO team from DISC in order to build and train a neural network using data obtained with automatic test pattern generation. To detect if a message is a false one, its correctness will be established relatively to the preceding messages. Indeed, considering a time series of messages, anomalous messages will more or less appear as outliers. The need for a temporal analysis calls for a solution based on a recurrent neural network architecture.

2.2/ Medical image analysis

The design of automated image analysis tools, among which machine learning based ones, is a longstanding issue in the medical imaging field. Indeed, Computer-Aided Diagnosis (CAD) from medical images has been used to help radiologists for a long time and new solutions are continually proposed. Meanwhile, in the race for improving the quality of image diagnosis and interpretation, deep neural networks have become the methodology of choice for analyzing medical images. Hence, over the past few years, medical image analysis problems have been increasingly addressed with deep learning concepts, in particular convolutional networks, leading to solutions providing state-of-the-art results. A typical example showing the profound and lasting impact of deep learning on healthcare diagnostics is the detection of diabetic retinopathy, which is the leading cause of blindness in the developed world. Following this trend, we will take advantage of our skills in deep learning to revisit the lung motion simulation problem and to propose a solution to help doctors to evaluate the state of the heart after myocardial infarction.

For the simulation of lung motion, we think that a Generative Adversarial Network (GAN) is a good starting point for two reasons. First, as shown by some research works, a GAN has the potential to predict pseudo-CT images from their corresponding MR images, even if the quality of a pseudo-CT image must be enhanced. Anatomical boundaries are more or less well-reconstructed, but cavities usually exhibit discrepancies. Second, GANs have also been proposed to predict video frames. A first challenge would be to identify a GAN architecture for simulating the lung motion in a single slice of a 4D-CT during a respiratory cycle, and for that two kinds of networks will have to be devised and trained: a generator and a discriminator. The generator produces samples in order to fool the discriminator, whereas the latter tries to distinguish between real and generated samples. Some studies have also shown that tumor motion is anatomic (lobe) location dependent, thus a further improvement of the precision could be obtained by working independently on each lung lobe. Finally, let us notice that the issue of respiratory motion compensation during external-beam radiation therapy is not limited to the lungs, since respiration causes significant motion of other organs such as the liver.

Assessing myocardial viability is of particular interest to estimate the state of the heart after Myocardial Infarction (MI), as it helps to determine whether the segment can recover functionally from revascularization procedures. To address this question, Cardiovascular MRI (CMRI), which is a non-invasive imaging technique, is the appropriate medical imaging modality. Compared to other modalities, it provides high-resolution images of the heart, and without radiation, adequate for the discrimination of viable and non-viable myocardium. Therefore, considering CMRI scans, we have started a preliminary study in the perspective of a deep learning based CAD tool dedicated to diagnostics in order to help cardiologists to choose the best treatment for the heart defects. Once designed, this tool is expected to automatically detect the different relevant areas (the myocardial contours, the infarcted area, the permanent microvascular obstruction area, and the border zone of the myocardial infarction) from a series of short-axis DE-MRI covering the left ventricle and then to make a quantification of the MI, in absolute value (mm3) or percentage of the myocardium. Semantic segmentation will be the key deep learning concept in the proposal by identifying the infarcted area. This future work will take place in a collaboration with colleagues of the LE2I lab from the University of Burgundy (one of them who is a hospital practitioner will provide us with images) and benefits from a financial support as a three year UBFC ISITE-BFC / Industry project starting at the end of 2018.

2.3/ Bioinformatics

Biological data are known to be difficult to analyze due to their complex and heterogeneous nature. To overcome this difficulty, computational pipelines are usually developed, as was the case during the PhD thesis that we have co-supervised and which addressed the problem of reconstruction of ancestral DNA sequences. To automatically extract knowledge from data, conventional machine learning tools have also been investigated, but to obtain meaningful results a lot of energy is spent in transforming the raw data into high-level features. Deep learning can bring a valuable contribution by extracting high-abstraction level features on condition that the biological data can exhibit a structure from which existing deep learning concepts can take advantage. For example, when dealing with images, CNNs exploit their neighborhood structure by convolving subsets of input pixels, thus to apply such deep networks to biological features, an analogous structure must be available.

Considering the perspective of further investigating ancestral genome reconstruction of microorganisms, we have identified a first task for which deep learning solutions might be studied. To explore how organellar genomes have evolved over time, well annotated genomes are required. This annotation task, which consists in the transformation of each complete genome in an ordered list of genes, starts with an inspection to find the location of the coding DNA sequences followed by their clustering in genes. Several tools for (semi-)automated annotation are already available, like DOGMA (a tool for annotating plant chloroplast and animal mitochondrial genomes) that has been used in the PhD thesis we have co-supervised. However, they suffer from limited customizability of reference sequences by the user, which means that sequence annotation is still a bottleneck. We think that deep learning can help to detect the location of the coding DNA sequences by identifying interesting features in genomes.

Bibliography

[1] P. Giraud, T. Lacornerie, and F. Mornex. "Radiothérapie des cancers primitifs du poumon". In: *Cancer/Radiothérapie* 20 (2016). Recorad : Recommandations pour la pratique de la radiothérapie externe et de la curiethérapie, S147–S156. ISSN: 1278-3218. DOI: https://doi.org/10.1016/j.canrad.2016.07.009. URL: http://www.sciencedirect.com/science/article/pii/S1278321816301305.

[2] Rémy Laurent et al. "Respiratory lung motion using an artificial neural network". In: *Neural Computing and Applications* 21.5 (2012), pp. 929–934. DOI: 10.1007/s00521-011-0727-y. IF Web of Science JCR 1.492, SJR Q2 in Artificial Intelligence.

[3] Rémy Laurent, Régine Gschwind, Michel Salomon, Julien Henriet, and Libor Makovicka. "Perspective de la plate-forme NEMOSIS dans le cadre d´une réduction de doses en imagerie". In: *Radioprotection* 47.4 (2012), pp. 599–617. DOI: 10.1051/radiopro/2012030. IF Web of Science JCR 0.508, SJR Q3 in Nuclear Energy and Engineering.

[4] Rémy Laurent et al. "Data Processing using Artificial Neural Networks to Improve the Simulation of Lung Motion". In: *Biomedical Engineering: Applications, Basis and Communications (BME)* 24.06 (2012), pp. 563–571. DOI: 10.4015/S1016237212500524. IF Scopus 0.233, SJR Q4 in Bioengineering.

[5] Jean-François Bosset, Julien Henriet, Rémy Laurent, Libor Makovicka, and Michel Salomon. *NEMOSIS V1.0 du 21/09/11*. Produit logiciel. Numéro de dépôt APP : IDDN.FR.001.170023.000.S.P.2012.000.31230 (logiciel oeuvre de l'Université de Franche-Comté). 2012.

[6] Gitte Fredberg Persson et al. "Deviations in delineated GTV caused by artefacts in 4DCT". In: *Radiotherapy and Oncology* 96.1 (2010), pp. 61–66.

[7] Jan Ehrhardt et al. "An optical flow based method for improved reconstruction of 4D CT data sets acquired during free breathing". In: *Medical Physics* 34.2 (2007), pp. 711–721.

[8] David Sarrut et al. "Nonrigid registration method to assess reproducibility of breath-holding with ABC in lung cancer". In: *International Journal of Radiation Oncology*Biology*Physics* 61.2 (2005), pp. 594–607. ISSN: 0360-3016. DOI: http://dx.doi.org/10.1016/j.ijrobp.2004.08.007. URL: http://www.sciencedirect.com/science/article/pii/S0360301604022540.

[9] D. Sarrut et al. "A Comparison Framework for Breathing Motion Estimation Methods From 4-D Imaging". In: *IEEE Transactions on Medical Imaging* 26.12 (Dec. 2007), pp. 1636–1648. ISSN: 0278-0062. DOI: 10.1109/TMI.2007.901006.

[10] Pierre-Frédéric Villard. "Simulation du mouvement pulmonaire pour un traitement oncologique". PhD thesis. Université Claude Bernard-Lyon I, 2006.

[11] Vlad Boldea, Gregory C Sharp, Steve B Jiang, and David Sarrut. “4D-CT lung motion estimation with deformable registration: Quantification of motion nonlinearity and hysteresis”. In: *Medical physics* 35.3 (2008), pp. 1008–1018.

[12] Alexandre Hostettler, Stéphane A Nicolau, Clement Forest, Luc Soler, and Yves Rémond. “Real time simulation of organ motions induced by breathing: First evaluation on patient data”. In: *ISBMS* 4072 (2006), pp. 9–18.

[13] H Wu et al. “Hysteresis analysis of lung tumor motion in radiation treatment”. In: *International Journal of Radiation Oncology* Biology* Physics* 72.1 (2008), S443–S444.

[14] Yuehui Chen, Yan Wang, and Bo Yang. “Evolving hierarchical RBF neural networks for breast cancer detection”. In: *Neural information processing*. Springer. 2006, pp. 137–144.

[15] Yubin Yang, Shifu Chen, Zhihua Zhou, Hui Lin, and Yukun Ye. “An intelligent medical image understanding method using two-tier neural network ensembles”. In: *Innovations in Applied Artificial Intelligence* (2005), pp. 143–155.

[16] Libor Makovicka et al. “Avenir des nouveaux concepts des calculs dosimétriques basés sur les méthodes de Monte Carlo”. In: *Radioprotection* 44.1 (2009), pp. 77–88. DOI: 10.1051/radiopro/2008055.

[17] Marcus Isaksson, Joakim Jalden, and Martin J Murphy. “On using an adaptive neural network to predict lung tumor motion during respiration for radiotherapy applications”. In: *Medical physics* 32.12 (2005), pp. 3801–3809.

[18] Martin J Murphy and Damodar Pokhrel. “Optimization of an adaptive neural network to predict breathing”. In: *Medical physics* 36.1 (2009), pp. 40–47.

[19] Iffat A Gheyas and Leslie S Smith. “A neural network-based framework for the reconstruction of incomplete data sets”. In: *Neurocomputing* 73.16 (2010), pp. 3039–3065.

[20] Seonyeong Park, Suk Jin Lee, Elisabeth Weiss, and Yuichi Motai. “Intra-and inter-fractional variation prediction of lung tumors using fuzzy deep learning”. In: *IEEE journal of translational engineering in health and medicine* 4 (2016), pp. 1–12.

[21] Jef Vandemeulebroucke, David Sarrut, Patrick Clarysse, et al. “The POPI-model, a point-validated pixel-based breathing thorax model”. In: *XVth international conference on the use of computers in radiation therapy (ICCR)*. Vol. 2. 2007, pp. 195–199.

[22] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (1989), pp. 303–314.

[23] Jacques M Bahi, Sylvain Contassot-Vivier, and Marc Sauget. “An incremental learning algorithm for function approximation”. In: *Advances in Engineering Software* 40.8 (2009), pp. 725–730.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. New York: Springer, 2006.

[25] Pierre-Emmanuel Leni et al. “Development of a 4D numerical chest phantom with customizable breathing”. In: *Physica Medica: European Journal of Medical Physics* 32.6 (2016), pp. 795–800. DOI: doi:10.1016/j.ejmp.2016.05.004. IF Web of Science JCR 1.99, SJR Q2 in Radiology, Nuclear Medicine and Imaging.

[26] Jean-François Couchot, Karine Deschinkel, and Michel Salomon*. “Suitability of artificial neural network for MEMS-based flow control”. In: *2012 Second Workshop on Design, Control and Software Implementation for Distributed MEMS, Proceedings*. Besançon, Apr. 2012, pp. 1–6. DOI: 10.1109/dMEMS.2012.17.

[27] Jean-François Couchot, Karine Deschinkel, and Michel Salomon*. “Active MEMS-based flow control using artificial neural network”. In: *Mechatronics* 23.7 (2013), pp. 898–905. DOI: 10.1016/j.mechatronics.2013.02.010. IF Web of Science JCR 1.871, SJR Q1 in Computer Science.

[28] P. Pernod et al. “MEMS magneto-mechanical microvalves (MMMS) for aerodynamic active flow control”. In: *Journal of Magnetism and Magnetic Materials* 322.9-12 (2010). Proceedings of the Joint European Magnetic Symposia, pp. 1642–1646. ISSN: 0304-8853. DOI: DOI:10.1016/j.jmmm.2009.04.086. URL: http://www.sciencedirect.com/science/article/B6TJJ-4WB3NGP-1/2/afb108de0880e448abbd099e4cff8159.

[29] FLUENT. *User’s Guide. Fluent 6.2*. Lebanon, USA. 2005.

[30] S. Scott Collis, Ronald D. Joslin, Avi Seifert, and Vassilis Theofilis. “Issues in active flow control: theory, control, simulation, and experiment”. In: *Progress in Aerospace Sciences* 40.4-5 (2004), pp. 237–289. ISSN: 0376-0421. DOI: DOI:10.1016/j.paerosci.2004.06.001. URL: http://www.sciencedirect.com/science/article/B6V3V-4D4JHMK-2/2/56c9adca9e7be0568a6765d2e18fac81.

[31] P.-Y. Pamart, J. Dandois, E. Garnier, and P. Sagaut. “NARX Modeling and Adaptive Close-Loop Control of a Separation by Synthetic Jet in Unsteady RANS computations”. In: *5th Flow Control Conference*. American Institute of Aeronautics and Astronautics, July 2010.

[32] J.-R. Frutos, D. Vernier, Y. Bailly, F. Bastien, and M. De Labachelerie. “« An Electrostatically Actuated Valve For Turbulent Boundary Layer Control.” Anglais. In: *actes de IEEE Sensors - Conference, Irvine, USA,* IEEE Sensors, 2005, pp135–142. URL: http://hal.archives-ouvertes.fr/hal-00135729/en/.

[33] Joydeep Ghosh and Yoan Shin. “Efficient Higher-Order Neural Networks for Classification and Function Approximation”. In: *Int. J. Neural Syst.* 3.4 (1992), pp. 323–350.

[34] Scott E. Fahlman and Christian Lebiere. “The Cascade-Correlation Learning Architecture”. In: *NIPS*. Ed. by David S. Touretzky. Morgan Kaufmann, 1989, pp. 524–532. ISBN: 1-55860-100-7.

[35] Lutz Prechelt. “Automatic early stopping using cross validation: quantifying the criteria”. In: *Neural Networks* 11.4 (1998), pp. 761–767. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(98)00010-0. URL: http://www.sciencedirect.com/science/article/pii/S0893608098000100.

[36] Christophe Guyeux. “Désordre des itérations chaotiques et leur utilité en sécurité informatique”. 2010BESA2019. PhD thesis. 2010, 1 vol. (244p.) URL: http://www.theses.fr/2010BESA2019.

[37] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems*. 2nd. Redwood City, CA: Addison-Wesley, 1989.

[38] Jacques M. Bahi, Christophe Guyeux, and Michel Salomon*. “Building a Chaotic Proved Neural Network”. In: vol. abs/1101.4351. 2011. URL: http://arxiv.org/abs/1101.4351.

[39] Jacques M. Bahi, Jean-François Couchot, Christophe Guyeux, and Michel Salomon*. “Neural networks and chaos: Construction, evaluation of chaotic networks, and prediction of chaos with multilayer feedforward networks”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1 (2012), p. 013122. DOI: 10.1063/1.3685524. IF Web of Science JCR 2.049, SJR Q2 in Applied Mathematics.

[40] Jacques M. Bahi, Nathalie Coté, Christophe Guyeux, and Michel Salomon*. “Protein folding in the 2D Hydrophobic–Hydrophilic (HP) square lattice model is chaotic”. In: *Cognitive Computation* 4 (1 2012), pp. 98–114. DOI: 10.1007/s12559-011-9118-z. IF Web of Science JCR 1.933, SJR Q2 in Computer Science Applications.

[41] KA Dill. “Theory for the folding and stability of globular proteins.” In: *Biochemistry* 24.6 (Mar. 1985), pp. 1501–1509. URL: http://ukpmc.ac.uk/abstract/MED/3986190.

[42] K. Aihara, T. Takabe, and M. Toyoda. “Chaotic neural networks”. In: *Physics Letters A* 144.6-7 (1990), pp. 333–340. ISSN: 0375-9601.

[43] Nigel Crook and Tjeerd Olde Scheper. “A novel chaotic neural network architecture”. In: *ESANN*. 2001, pp. 295–300.

[44] Nigel Crook, Wee Jin Goh, and Mohammad Hawarat. “Pattern recall in networks of chaotic neurons”. In: *Biosystems* 87.2-3 (2007), pp. 267–274. ISSN: 0303-2647. DOI: DOI:10.1016/j.biosystems.2006.09.022.

[45] Ilker Dalkiran and Kenan Danisman. “Artificial neural network based chaotic generator for cryptology”. In: *Turk. J.Elec. Eng. & Comp. Sci.* 18.2 (2010), pp. 225–240.

[46] Shiguo Lian. “A block cipher based on chaotic neural networks”. In: *Neurocomputing* 72.4-6 (2009), pp. 1296–1301. ISSN: 0925-2312.

[47] Yantao Li, Shaojiang Deng, and Di Xiao. “A novel Hash algorithm construction based on chaotic neural network”. In: *Neural Computing and Applications* 20.1 (Feb. 2011), pp. 133–141. ISSN: 1433-3058. DOI: 10.1007/s00521-010-0432-2.

[48] F. Robert. *Discrete Iterations: A Metric Study*. Ed. by Springer-Verlag. Vol. 6. Springer Series in Computational Mathematics. 1986.

[49] J. Banks, J. Brooks, G. Cairns, and P. Stacey. “On Devaney’s Definition of Chaos”. In: *Amer. Math. Monthly* 99 (1992), pp. 332–334.

[50] Christophe Guyeux and Jacques M. Bahi. “Hash functions using chaotic iterations”. In: *Journal of Algorithms & Computational Technology* 4.2 (2010), pp. 167–182.

[51] Gerald Böhm. “Protein folding and deterministic chaos: Limits of protein folding simulations and calculations”. In: *Chaos, Solitons & Fractals* 1.4 (1991), pp. 375–382. ISSN: 0960-0779. DOI: DOI:10.1016/0960-0779(91)90028-8. URL: http://www.sciencedirect.com/science/article/B6TJ4-46CBXVT-1X/2/370489c218e4c2732cd9b620ef50c696.

[52] Michael Braxenthaler, R. Ron Unger, Ditza Auerbach, and John Moult. “Chaos in protein dynamics”. In: *Proteins-structure Function and Bioinformatics* 29 (1997), pp. 417–425. DOI: 10.1002/(SICI)1097-0134(199712)29:4<417::AID-PROT2>3.3.CO;2-O.

[53] Alena Shmygelska and Holger Hoos. “An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem”. In: *BMC Bioinformatics* 6.1 (2005), p. 30. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-30.

[54] Md. Hoque, Madhu Chetty, and Abdul Sattar. “Genetic Algorithm in Ab Initio Protein Structure Prediction Using Low Resolution Model: A Review”. In: *Biomedical Data and Applications*. Ed. by Amandeep Sidhu and Tharam Dillon. Vol. 224. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009, pp. 317–342.

[55] R. Backofen, S. Will, and P. Clote. *Algorithmic Approach To Quantifying The Hydrophobic Force Contribution In Protein Folding*. 1999.

[56] Trent Higgs, Bela Stantic, Tamjidul Hoque, and Abdul Sattar. “Genetic algorithm feature-based resampling for protein structure prediction”. In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–8.

[57] Md. Kamrul Islam and Madhu Chetty. “Clustered memetic algorithm for protein structure prediction”. In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–8.

[58] Dragos Horvath and Camelia Chira. “Simplified chain folding models as metaheuristic benchmark for tuning real protein folding algorithms?” In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–8.

[59] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. “PREDICTING THE FUTURE: A CONNECTIONIST APPROACH”. In: *International Journal of Neural Systems* 01.03 (1990), pp. 193–209. DOI: 10 . 1142 / S0129065790000102. eprint: http://www.worldscientific.com/doi/pdf/10.1142/S0129065790000102. URL: http://www.worldscientific.com/doi/abs/10.1142/S0129065790000102.

[60] Li Deng and Dong Yu. “Deep Learning: Methods and Applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387. ISSN: 1932-8346. DOI: 10.1561/2000000039. URL: http://dx.doi.org/10.1561/2000000039.

[61] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: http://dx.doi.org/10.1016/j.neunet.2014.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0893608014002135.

[62] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[63] Andrej Karpathy and Li Fei-Fei. “Deep visual-semantic alignments for generating image descriptions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137.

[64] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. “Regularization of neural networks using dropconnect”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 1058–1066.

[65] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[66] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[68] Hyeon-Joong Yoo. “Deep Convolution Neural Networks in Computer Vision”. In: *IEEE Transactions on Smart Processing & Computing* 4.1 (2015), pp. 35–43.

[69] David Verstraeten, Benjamin Schrauwen, Michiel d’Haene, and Dirk Stroobandt. “An experimental unification of reservoir computing methods”. In: *Neural Networks* 20.3 (2007), pp. 391–403.

[70] Mantas LukošEvičIus and Herbert Jaeger. “Reservoir computing approaches to recurrent neural network training”. In: *Computer Science Review* 3.3 (2009), pp. 127–149.

[71] Azarakhsh Jalalvand, Glenn Van Wallendael, and Rik Van de Walle. “Real-time reservoir computing network-based systems for detection tasks on visual contents”. In: *Computational Intelligence, Communication Systems and Networks (CICSyN), 2015 7th International Conference on*. IEEE. 2015, pp. 146–151.

[72] Jean-François Couchot, Raphaël Couturier, and Michel Salomon*. “Improving Blind Steganalysis in Spatial Domain using a Criterion to Choose the Appropriate Steganalyzer between CNN and SRM+EC”. In: *ICT Systems Security and Privacy Protection: 32nd IFIP TC 11 International Conference, SEC 2017, Rome, Italy, May 29-31, 2017, Proceedings, IFIP Advances in Information and Communication Technology*. Ed. by Sabrina De Capitani di Vimercati and Fabio Martinelli. Vol. 502. Springer International Publishing, 2017, pp. 327–340. DOI: 10.1007/978-3-319-58469-0. Rank B.

[73] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. “Structural Design of Convolutional Neural Networks for Steganalysis”. In: *IEEE Signal Processing Letters* 23.5 (2016), pp. 708–712.

[74] Gustavus J Simmons. “The prisoners’ problem and the subliminal channel”. In: *Advances in Cryptology*. Springer. 1984, pp. 51–67.

[75] Tomás Pevný, Tomás Filler, and Patrick Bas. “Using High-Dimensional Image Models to Perform Highly Undetectable Steganography”. In: *Information Hiding - 12th International Conference, IH 2010, Calgary, AB, Canada, June 28-30, 2010, Revised Selected Papers*. Ed. by Rainer Böhme, Philip W. L. Fong, and Reihaneh Safavi-Naini. Vol. 6387. Lecture Notes in Computer Science. Springer, 2010, pp. 161–177. ISBN: 978-3-642-16434-7. URL: http://dx.doi.org/10.1007/978-3-642-16435-4.

[76] Vojtech Holub and Jessica J. Fridrich. “Designing steganographic distortion using directional filters.” In: *WIFS*. IEEE, 2012, pp. 234–239. ISBN: 978-1-4673-2285-0.

[77] Bin Li, Ming Wang, Jiwu Huang, and Xiaolong Li. “A new cost function for spatial image steganography”. In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 4206–4210.

[78] Jean-François Couchot, Raphaël Couturier, and Christophe Guyeux. “STABYLO: steganography with adaptive, Bbs, and binary embedding at low cost”. In: *Annales des Télécommunications* 70.9-10 (2015), pp. 441–449. DOI: 10.1007/s12243-015-0466-7. URL: http://dx.doi.org/10.1007/s12243-015-0466-7.

[79] V. Sedighi, R. Cogranne, and J. Fridrich. “Content-Adaptive Steganography by Minimizing Statistical Detectability”. In: *IEEE Transactions on Information Forensics and Security* 11.2 (Feb. 2016), pp. 221–234. ISSN: 1556-6013. DOI: 10.1109/TIFS.2015.2486744.

[80] Jessica Fridrich, Tomáš Pevny, and Jan Kodovsky. “Statistically undetectable jpeg steganography: dead ends challenges, and opportunities”. In: *Proceedings of the 9th workshop on Multimedia & security*. ACM. 2007, pp. 3–14.

[81] Linjie Guo, Jiangqun Ni, and Yun Qing Shi. “Uniform embedding for efficient JPEG steganography”. In: *IEEE Transactions on Information Forensics and Security* 9.5 (2014), pp. 814–825.

[82] Linjie Guo, Jiangqun Ni, Wenkang Su, Chengpei Tang, and Yun-Qing Shi. “Using statistical image model for JPEG steganography: uniform embedding revisited”. In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2669–2680.

[83] Vojtech Holub, Jessica Fridrich, and Tomás Denemark. “Universal distortion function for steganography in an arbitrary domain”. English. In: *EURASIP Journal on Information Security* 2014.1, 1 (2014). DOI: 10.1186/1687-417X-2014-1. URL: http://dx.doi.org/10.1186/1687-417X-2014-1.

[84] Tomás Denemark, Vahid Sedighi, Vojtech Holub, Rémi Cogranne, and Jessica J. Fridrich. “Selection-channel-aware rich model for Steganalysis of digital images”. In: *2014 IEEE International Workshop on Information Forensics and Security, WIFS 2014, Atlanta, GA, USA, December 3-5, 2014*. IEEE, 2014, pp. 48–53. ISBN: 978-1-4799-8882-2. DOI: 10.1109/WIFS.2014.7084302. URL: http://dx.doi.org/10.1109/WIFS.2014.7084302.

[85] Patrick Bas, Tomáš Filler, and Tomáš Pevný. “”Break Our Steganographic System”: The Ins and Outs of Organizing BOSS”. In: *Information Hiding: 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*. Ed. by Tomáš Filler, Tomáš Pevný, Scott Craver, and Andrew Ker. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 59–70. ISBN: 978-3-642-24178-9. DOI: 10.1007/978-3-642-24178-9_5.

[86] Jessica J. Fridrich and Jan Kodovský. “Rich Models for Steganalysis of Digital Images”. In: *IEEE Trans. Information Forensics and Security* 7.3 (2012), pp. 868–882. DOI: 10.1109/TIFS.2012.2190402. URL: http://dx.doi.org/10.1109/TIFS.2012.2190402.

[87] Vojtech Holub and Jessica J. Fridrich. “Random Projections of Residuals for Digital Image Steganalysis”. In: *IEEE Trans. Information Forensics and Security* 8.12 (2013), pp. 1996–2006. DOI: 10.1109/TIFS.2013.2286682. URL: http://dx.doi.org/10.1109/TIFS.2013.2286682.

[88] J. Fridrich and J. Kodovský. “Multivariate gaussian model for designing additive distortion for steganography”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. May 2013, pp. 2949–2953. DOI: 10.1109/ICASSP.2013.6638198.

[89] Vojtech Holub and Jessica J. Fridrich. “Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT”. In: *IEEE Trans. Information Forensics and Security* 10.2 (2015), pp. 219–228. DOI: 10.1109/TIFS.2014.2364918. URL: http://dx.doi.org/10.1109/TIFS.2014.2364918.

[90] Jan Kodovský, Jessica J. Fridrich, and Vojtech Holub. "Ensemble Classifiers for Steganalysis of Digital Media". In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 432–444. DOI: 10.1109/TIFS.2011.2175919. URL: http://dx.doi.org/10.1109/TIFS.2011.2175919.

[91] Hans Georg Schaathun. "Support Vector Machines". In: *Machine Learning in Image Steganalysis*. John Wiley & Sons, Ltd, 2012, pp. 179–196. ISBN: 9781118437957. DOI: 10.1002/9781118437957.ch11. URL: http://dx.doi.org/10.1002/9781118437957.ch11.

[92] Ivans Lubenko and Andrew D. Ker. "Steganalysis with Mismatched Covers: Do Simple Classifiers Help?" In: *Proceedings of the on Multimedia and Security*. MM&#38;Sec '12. Coventry, United Kingdom: ACM, 2012, pp. 11–18. ISBN: 978-1-4503-1417-6. DOI: 10.1145/2361407.2361410. URL: http://doi.acm.org/10.1145/2361407.2361410.

[93] Weixuan Tang, Haodong Li, Weiqi Luo, and Jiwu Huang. "Adaptive Steganalysis Against WOW Embedding Algorithm". In: *Proceedings of the 2Nd ACM Workshop on Information Hiding and Multimedia Security*. IH&MMSec '14. Salzburg, Austria: ACM, 2014, pp. 91–96. ISBN: 978-1-4503-2647-6. DOI: 10.1145/2600918.2600935. URL: http://doi.acm.org/10.1145/2600918.2600935.

[94] Weixuan Tang, Haodong Li, Weiqi Luo, and Jiwu Huang. "Adaptive Steganalysis Based on Embedding Probabilities of Pixels". In: *IEEE Transactions on Information Forensics and Security* 11.4 (2016), pp. 734–745.

[95] Tomáš Denemark Denemark, Mehdi Boroumand, and Jessica Fridrich. "Steganalysis Features for Content-Adaptive JPEG Steganography". In: *IEEE Transactions on Information Forensics and Security* 11.8 (2016), pp. 1736–1746.

[96] Vojtěch Holub and Jessica Fridrich. "Phase-aware projection model for steganalysis of JPEG images". In: *SPIE/IS&T Electronic Imaging*. International Society for Optics and Photonics. 2015, 94090T–94090T.

[97] B. Graham. "Fractional Max-Pooling". In: *ArXiv e-prints* (Dec. 2014). arXiv: 1412.6071 [cs.CV].

[98] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *ArXiv e-prints* (Nov. 2015). arXiv: 1511.07289 [cs.LG].

[99] *DreamUp Vision*. http://dreamupvision.com/. Accessed: 2016-04-14.

[100] Michel Salomon, Raphaël Couturier, Christophe Guyeux, Jean-François Couchot, and Jacques M. Bahi. "Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine". In: *European Research in Telemedicine / La Recherche Européenne en Télémédecine* 6.2 (2017), pp. 79–92. ISSN: 2212-764X. DOI: 10.1016/j.eurtel.2017.06.001. SJR Q3 in Health Informatics.

[101] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. "Deep Learning for Steganalysis via Convolutional Neural Networks". In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2015, 94090J–94090J.

[102] Lionel Pibre, Pasquet Jérôme, Dino Ienco, and Marc Chaumont. "Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch". In: *Media Watermarking, Security, and Forensics, EI: Electronic Imaging*. 2016.

[103] Shunquan Tan and Bin Li. "Stacked Convolutional Auto-encoders for Steganalysis of Digital Images". In: *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*. IEEE. 2014, pp. 1–4.

[104] Jean-François Couchot, Raphaël Couturier, Christophe Guyeux, and Michel Salomon*. "Steganalysis via a Convolutional Neural Network using Large Convolution Filters for Embedding Process with Same Stego Key". In: *ArXiv e-prints* (May 2016). arXiv: 1605.07946 `[cs.MM]`.

[105] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. "Ensemble of CNNs for Steganalysis: An Empirical Study". In: *ACM Workshop on Information Hiding and Multimedia Security*. 2016.

[106] Tomáš Denemark, Jessica Fridrich, and Pedro Comesaña-Alfaro. "Improving selection-channel-aware steganalysis features". In: *Electronic Imaging* 2016.8 (2016), pp. 1–8.

[107] Y. Qian, J. Dong, W. Wang, and T. Tan. "Learning and transferring representations for image steganalysis using convolutional neural network". In: *2016 IEEE International Conference on Image Processing (ICIP)*. Sept. 2016, pp. 2752–2756. DOI: 10.1109/ICIP.2016.7532860.

[108] J. Zeng, S. Tan, B. Li, and J. Huang. "Large-scale JPEG steganalysis using hybrid deep-learning framework". In: *ArXiv e-prints* (Nov. 2016). arXiv: 1611.03233 `[cs.MM]`.

[109] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[110] M. Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.04467 `[cs.DC]`.

[111] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning". In: *BigLearn, NIPS Workshop*. EPFL-CONF-192376. 2011.

[112] Gilles Perrot, Stéphane Domas, Raphaël Couturier, and Nicolas Bertaux. "Fast GPU-based denoising filter using isoline levels". In: *J. Real-Time Image Processing* 12.1 (2016), pp. 31–42. DOI: 10.1007/s11554-013-0381-y. URL: https://doi.org/10.1007/s11554-013-0381-y.

[113] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[114] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. "Weighted nuclear norm minimization with application to image denoising". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2862–2869.

[115] Harold C Burger, Christian J Schuler, and Stefan Harmeling. "Image denoising: Can plain neural networks compete with BM3D?" In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2392–2399.

[116] Junyuan Xie, Linli Xu, and Enhong Chen. “Image Denoising and Inpainting with Deep Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 341–349. URL: http://dl.acm.org/citation.cfm?id=2999134.2999173.

[117] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[118] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising”. In: *IEEE Transactions on Image Processing* PP.99 (2017), pp. 1–1. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2662206.

[119] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1706.05587 (2017). arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

[120] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (Apr. 2018), pp. 834–848. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2699184.

[121] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. “Learning deep CNN denoiser prior for image restoration”. In: *arXiv preprint* (2017).

[122] K. Zhang, W. Zuo, and L. Zhang. “FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising”. In: *IEEE Transactions on Image Processing* (2018), pp. 1–1. ISSN: 1057-7149. DOI: 10.1109/TIP.2018.2839891.

[123] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CoRR* abs/1611.07004 (2016). URL: http://arxiv.org/abs/1611.07004.

[124] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[125] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241.

[126] Damien François, Vincent Wertz, Michel Verleysen, et al. “Non-Euclidean metrics for similarity search in noisy datasets.” In: *ESANN*. 2005, pp. 339–344.

[127] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. “Loss Functions for Image Restoration With Neural Networks”. In: *IEEE Transactions on Computational Imaging* 3.1 (Mar. 2017), pp. 47–57. DOI: 10.1109/TCI.2016.2644865.

[128] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[129] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment". In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. Vol. 2. Ieee. 2003, pp. 1398–1402.

[130] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[131] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics". In: *Proc. 8th Int'l Conf. Computer Vision*. Vol. 2. July 2001, pp. 416–423.

[132] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.5 (May 2011), pp. 898–916. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.161. URL: http://dx.doi.org/10.1109/TPAMI.2010.161.

[133] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. 2016, pp. 2802–2810. URL: http://papers.nips.cc/paper/6172-image-restoration-using-very-deep-convolutional-encoder-decoder-networks-with-symmetric-skip-connections.

[134] Peng Liu and Ruogu Fang. "Wide Inference Network for Image Denoising". In: *CoRR* abs/1707.05414 (2017). arXiv: 1707.05414. URL: http://arxiv.org/abs/1707.05414.

[135] Yunjin Chen and Thomas Pock. "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2017), pp. 1256–1272.

[136] Kede Ma et al. "Waterloo Exploration Database: New Challenges for Image Quality Assessment Models". In: *IEEE Transactions on Image Processing* 26.2 (Feb. 2017), pp. 1004–1016.

[137] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[138] Nils Schaetti, Michel Salomon, and Raphaël Couturier. "Echo state networks-based reservoir computing for MNIST handwritten digits recognition". In: *19th IEEE International Conference on Computational Science and Engineering, CSE 2016, Paris, 24-26 August 2016, Proceedings*. 2016, pp. 484–491. DOI: 10.1109/CSE-EUC-DCABES.2016.229.

[139] K. Doya. "Bifurcations in the learning of recurrent neural networks". In: *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*. Vol. 6. May 1992, 2777–2780 vol.6. DOI: 10.1109/ISCAS.1992.230622.

[140] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

[141] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[142] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. “An Empirical Exploration of Recurrent Network Architectures”. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. Lille, France: JMLR.org, 2015, pp. 2342–2350. URL: http://dl.acm.org/citation.cfm?id=3045118.3045367.

[143] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* PP.99 (2016), pp. 1–11. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2016.2582924.

[144] Herbert Jaeger. “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note”. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (2001), p. 34.

[145] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. “Optimization and applications of echo state networks with leaky-integrator neurons”. In: *Neural Networks* 20.3 (2007), pp. 335–352.

[146] Wolfgang Maass. “Liquid state machines: motivation, theory, and applications”. In: *Computability in context: computation and logic in the real world* (2010), pp. 275–296.

[147] Claudio Gallicchio and Alessio Micheli. “Echo State Property of Deep Reservoir Computing Networks”. In: *Cognitive Computation* (2017), pp. 1–14. ISSN: 1866-9964. DOI: 10.1007/s12559-017-9461-9. URL: http://dx.doi.org/10.1007/s12559-017-9461-9.

[148] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. “Deep reservoir computing: A critical experimental analysis”. In: *Neurocomputing* 268 (2017). Advances in artificial neural networks, machine learning and computational intelligence, pp. 87–99. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.12.089. URL: http://www.sciencedirect.com/science/article/pii/S0925231217307567.

[149] C. D. Schuman et al. “A Survey of Neuromorphic Computing and Neural Networks in Hardware”. In: *ArXiv e-prints* (May 2017). arXiv: 1705.06963.

[150] Chrisantha Fernando and Sampsa Sojakka. “Pattern Recognition in a Bucket”. In: *Advances in Artificial Life: 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003. Proceedings*. Ed. by Wolfgang Banzhaf, Jens Ziegler, Thomas Christaller, Peter Dittrich, and Jan T. Kim. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 588–597. ISBN: 978-3-540-39432-7. DOI: 10.1007/978-3-540-39432-7_63. URL: http://dx.doi.org/10.1007/978-3-540-39432-7_63.

[151] L. Larger et al. “Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing”. In: *Opt. Express* 20.3 (Jan. 2012), pp. 3241–3249. DOI: 10.1364/OE.20.003241. URL: http://www.opticsexpress.org/abstract.cfm?%5C%5CURI=oe-20-3-3241.

[152] Laurent Larger et al. “High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification”. In: *Phys. Rev. X* 7 (1 Feb. 2017), p. 011015. DOI: 10.1103/PhysRevX.7.011015. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.011015.

[153] Daniel Brunner, Miguel C Soriano, Claudio R Mirasso, and Ingo Fischer. “Parallel photonic information processing at gigabyte per second data rates using transient states”. In: *Nature communications* 4 (2013), p. 1364.

[154] G. Van der Sande, D. Brunner, and M. Soriano. “Advances in photonic reservoir computing”. In: *Nanophotonics* 6.3 (May 2017), pp. 561–576. DOI: 10.1515/nanoph-2016-0132. URL: https://www.degruyter.com/view/j/nanoph.2017.6.issue-3/nanoph-2016-0132/nanoph-2016-0132.xml.

[155] Andrey Nikolaevich Tikhonov and Vasily Yakovlevitch Arsenin. *Solutions of ill-posed problems*. Winston, 1977.

[156] Mantas Lukoševičius. “Neural Networks: Tricks of the Trade: Second Edition”. In: ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Chap. A Practical Guide to Applying Echo State Networks, pp. 659–686. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_36. URL: http://dx.doi.org/10.1007/978-3-642-35289-8_36.

[157] Gilles Wainrib and Mathieu N. Galtier. “A local Echo State Property through the largest Lyapunov exponent”. In: *Neural Networks* 76 (2016), pp. 39–45. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2015.12.013. URL: http://www.sciencedirect.com/science/article/pii/S0893608015002828.

[158] Sebastian Basterrech. “Empirical Analysis of te Necessary and Sufficient Conditions of the Echo State Property”. In: *Proceedings of the 30th International Join Conference on Neural Networks*. IJCNN 2017. Anchorage, USA, 2017, x–y.

[159] Nils Bertschinger and Thomas Natschläger. “Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks”. In: *Neural Computation* 16.7 (2004), pp. 1413–1436. DOI: 10.1162/089976604323057443. eprint: http://dx.doi.org/10.1162/089976604323057443. URL: http://dx.doi.org/10.1162/089976604323057443.

[160] Léon Bottou et al. “Comparison of classifier methods: a case study in handwritten digit recognition”. In: *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*. Vol. 2. IEEE. 1994, pp. 77–82.

List of Figures

List of Tables

Abstract:

Since the introduction of the perceptron at the end of the 1950s, neural networks have passed through several boom-and-bust episodes, becoming today the state-of-the-art approach for solving many classification and prediction problems. The backpropagation algorithm has thus given rise to the multilayer networks era. More recently, there has been a revival with the emergence of various deep neural networks also known as deep learning architectures. The research works described in this manuscript reflect on the one hand this network architecture evolution and on the other hand highlight the suitability of neural networks for successfully solving tasks in different areas. In a first part we show that a multilayer perceptron neural network can simulate lung motion being able to predict the trajectory of several feature points, that it can be a substitute for a computational fluid dynamics software for active airflow control, and finally allows to build also a true chaotic neural network. In the second part, convolutional neural networks (a main flavor of deep learning) are used to detect whether an image embeds a hidden message or not, the obtained steganalysis approach was at the time of its publication the state-of-the-art one. Then we propose a fully convolutional neural network for image denoising, this latter, which is an encoder-decoder, can deal with different kinds of noise. Finally, considering, the Reservoir Computing paradigm studied by colleagues of our research institute, we study the application of Echo State Networks, a recurrent architecture, on a handwritten digits recognition problem (namely the well-known MNIST benchmark problem).

Keywords: Multilayer Perceptron, Deep Learning, Reservoir Computing

Résumé :

Depuis l'introduction du perceptron à la fin des années 1950, les réseaux de neurones ont alterné des périodes d'avancée et de stagnation, devenant aujourd'hui l'approche de référence pour la résolution de très nombreux problèmes de classification ou de prédiction. L'algorithme de rétropropagation du gradient donna ainsi lieu à l'âge des réseaux de neurones multicouches. Plus récemment, il y a eu une renaissance avec l'avènement de la famille des réseaux de neurones profonds plus communément connue sous le terme de deep learning. Les travaux de recherche décrits dans ce manuscrit reflètent, d'une part ces évolutions architecturales et d'autre part la grande capacité des réseaux de neurones à résoudre des problèmes dans des domaines variés. Dans une première partie nous montrons qu'un perceptron multicouche est une architecture permettant de simuler le mouvement pulmonaire, en étant capable de prédire la trajectoire d'un ensemble de points caractéristiques, de remplacer un programme de calcul de la mécanique des fluides pour le contrôle actif de flux d'air, et enfin de construire des réseaux de neurones chaotiques. Dans la seconde partie, nous exploitons des réseaux de neurones convolutionnels (une architecture phare du deep learning) afin de détecter si une image embarque ou non un message caché, l'approche de stéganalyse obtenue était lors de sa publication la plus performante. Nous proposons ensuite un réseau complètement convolutionnel pour le débruitage d'images, à savoir un encodeur-décodeur capable de traiter plusieurs types de bruits. Pour finir, dans le cadre des travaux sur le paradigme du Reservoir Computing menés au sein de notre institut de recherche, nous étudions l'application des Echo State Networks, une architecture récurrente, sur un problème de reconnaissance de chiffres manuscrits (le problème benchmark du MNIST).

Mots-clés : Perceptron Multicouche, Réseaux de Neurones Profonds, Reservoir Computing