UNIVERSITÉ DE TOURS

ÉCOLE DOCTORALE MIPTIS

Laboratoire d'Informatique Fondamentale et Appliquée de Tours (EA 6300)

THÈSE présentée par :

Chloé Martineau

soutenue le : 13/11/2019

pour obtenir le grade de Docteur de l'Université de Tours

Discipline/Spécialité : Informatique

Deep learning onto graph space : application to image-based insect recognition

Thèse dirigée par :

Donatello Conte	Maître de Conférences, HDR, Université de Tours

Rapporteurs :

Luc Brun	Professeur des Universités, École Nationale Supérieure d'Ingénieurs de Caen
Kaspar Riesen	Professeur, University of Applied Sciences and Arts Northwestern Switzerland

Présidente du jury :

Christel Vrain	Professeure des Universités, Université d'Orléans

Jury :

Luc Brun	Professeur des Universités, École Nationale Supérieure d'Ingénieurs de Caen, France
Donatello Conte	Maître de Conférences, HDR, Université de Tours
Pierre Héroux	Maître de Conférences, Université de Rouen
Romain Raveaux	Maître de Conférences, HDR, Université de Tours
Kaspar Riesen	Professeur, University of Applied Sciences and Arts Northwestern Switzerland
Gilles Venturini	Professeur des Universités, Université de Tours
Christel Vrain	Professeure des Universités, Université d'Orléans

Acknowledgement

Here are the persons I would like to thank, for being part more or less directly in this PhD thesis:

- Luc Brun and Kaspar Riesen, for having accepted to review my thesis and for their thorough and rigorous reading
- Pierre Héroux and Christel Vrain for being part of my PhD jury and for their interesting questions and remarks
- Donatello Conte, Romain Raveaux and Gilles Venturini for giving me their trust, their great supervision and guidance
- All my colleagues at LIFAT, for their warm fellowship and interesting discussions
- My friends from Polytech master degree and the doctoral school, for their kindness and friendship
- My parents and sister for the flawless support they gave me
- My girlfriend Marie for all we shared together (among them quality kibble sharing and laughs)

Résumé

Le but de cette thèse est d'étudier la reconnaissance d'insectes comme un problème de reconnaissance des formes basé images. Ce problème a été étudié en profondeur au long des trois dernières décennies autant par les communautés de la science informatique que de la biologie/entomologie. Une tendance émerge des travaux existants : les *a priori* sur les images ont été au fur et à mesure abandonnés, écartant les méthodes adhoc au profit de méthodes génériques et orientées données.

Un aspect reste, selon nous, toujours à expérimenter à ce jour : les approches profondes. À cet effet, la première contribution de cette thèse consiste à déterminer la faisabilité de l'application des réseaux de neurones convolutifs profonds (CNN) au problème de reconnaissance d'images d'insectes. Les limitations majeures sont les suivantes: les images sont très rares (la quantité de données étant critique à l'application des CNN car reposant uniquement sur celles-ci) et les cardinalités de classes sont hautement déséquilibrées (certains insectes étant beaucoup plus rares que d'autres.) Pour atténuer ces limitations, le *transfer learning* et la pondération de la fonction de coûts ont été employés.

Finalement, une question subsiste de l'état de l'art : les images d'insectes sont-elles spécifiques dans certaines de leurs propriétés et doivent-elles être formalisées en ce sens ? Dans ce cadre, nous étudions l'importance de la structure des images d'insectes pour leur classification à l'aide des représentations sous forme de graphes.

Deux tentatives sont présentées. La première consiste en la conception d'un classificateur de graphes de type *perceptron*. Le perceptron utilise l'appariement de graphes comme composant principal. Cependant, un appariement de graphes n'a de sens que lorsque deux graphes sont à comparer alors que, lors d'une classification, nous ne disposons que d'un seul graphe. Pour appliquer l'appariement de graphe dans un contexte de classification, le graph-perceptron comprend un graphe modèle qui est utilisé comme le deuxième opérande à l'appariement de graphe. Une version paramétrée du problème d'appariement de graphes ainsi qu'un algorithme de graph perceptron sont proposés pour apprendre l'appariement utilisé lors de la classification. Lorsque comparé à des approches similaires, notre approche est en moyenne 60 fois plus rapide pour une perte de 6% de taux de reconnaissance.

Le second travail basé sur les graphes de cette thèse est la définition d'un opérateur de convolution pour construire un modèle de réseaux de neurones convolutifs s'appliquant sur les graphes (GCNN.) Notre méthode utilise également l'appariement de graphes. Contrairement au graph-perceptron, notre couche de convolution sur graphe utilises l'appariement de graphes à une échelle locale sur le graphe: un graphe filtre est successivement apparié au graphe de voisinage de chacun des sommets du graphe. La valeur de l'appariement pour le voisinage d'un sommet donné est la valeur locale du signal convolué. Notre méthode se comporte de façon similaire à des approches équivalentes sur des problèmes de tailles réduites de type MNIST à deux classes.

Le dernier chapitre de la thèse s'applique à utiliser les méthodes mentionnées précédemment à des problèmes de reconnaissance d'images d'insectes. Deux bases d'images sont ici proposées. La première est constituée d'images prises en laboratoire sur arrière-plan constant. La seconde base est issue de la base ImageNet. Cette base est composée d'images prises en contexte naturel. Les

CNN entrainés avec transfer learning sont les plus performants sur ces bases d'images (uniquement en utilisant le transfer learning).

Abstract

The goal of this thesis is to investigate insect recognition as an image-based pattern recognition problem. This problem has been extensively studied along the previous three decades by both computer scientists and biologists/entomologists. Along the previous pieces of work, a trend could be observed : images were processed more and more in generic ways, discarding handcrafted adhoc methods for generic and data-driven approach.

An element is, to the best of our knowledge, still to be experimented as of 2017: deep approaches. Therefore, a contribution is about determining to what extent deep convolutional neural networks (CNNs) can be applied to image-based insect recognition. The most important limitations are the following ones : images are very rare (which is critical to deep neural networks as their classification abilities rely on profusion of data) and classes cardinalities are highly unbalanced (some insects being rarer than others.) In order to mitigate these limitations, transfer learning and cost function weighting were applied.

Finally, a question remains from the review of literature: are insect images specific in certain ways and should they be modeled accordingly? In such a context, we investigate the importance of structure in insect image classification through graph-based representations.

Two attempts are presented. The former consists in designing a graph-perceptron classifier. The perceptron uses graph matching as the main operator for classification. However, graph matching makes sense only when two graphs are compared whereas classification is about giving only one item as input. To apply graph matching in such a context, the graph-perceptron embeds a model graph that is used as the second operand. A parametrized graph matching problem and a graph perceptron algorithm are proposed to learn the matching used in classification. When compared to similar approaches, ours is 60 times faster on average for a 6% accuracy loss.

The latter graph-based work in this thesis is on defining convolution on graphs to build graph convolutional neural networks. Our method uses a graph matching problem. Compared to the previous graph perceptron approach, our graph convolutional layer uses graph matching at a local scale on the graph: a filter graph is successively matched to every vertex and its neighbourhood. The value of the matching at a given point is the local value of the convolved signal. Our approach behaves similarly on small MNIST-2class problems with equivalent approaches.

The last chapter of the thesis deals with applying most of the aforementioned methods to insect image recognition problems. Two datasets are proposed. The first one consists of lab-based images with constant background. The second one is generated by taking a ImageNet subset. This set is composed of field-based images. CNNs with transfer learning are the most successful method applied on these datasets (only when using transfer learning).

Contents

List of figures **14**

List of tables **16**

1 Introduction **17**
1.1 Insects and biology . . . 17
1.1.1 What insects are and why they are important . . . 17
1.1.2 How insects are visually identified by biologists . . . 17
1.2 How computer science can help with insect identification . . . 18
1.3 About artificial intelligence . . . 18
1.3.1 Pattern recognition . . . 18
1.3.2 Computer vision . . . 18
1.3.3 Machine learning . . . 19
1.3.4 About deep learning . . . 19
1.3.5 Structural vs euclidean pattern recognition . . . 20
1.4 Problem definition . . . 21
1.5 Manuscript organization . . . 22

2 State of the art in insect image recognition **23**
2.1 Introduction . . . 24
2.2 Scope and methodology . . . 24
2.2.1 Scope and previous surveys . . . 24
2.2.2 Methodology . . . 24
2.3 Image capture settings and apparatuses . . . 25
2.3.1 Advantages and drawbacks . . . 27
2.4 Feature extraction . . . 28
2.4.1 Foreground/Background detection . . . 28
2.4.2 Three general feature extraction methods . . . 29
2.4.3 Fixed/Handcrafted Feature Extractor . . . 29
2.4.4 Mid-level features . . . 31
2.4.5 Learning Hierarchical Representations . . . 31
2.4.6 Advantages and drawbacks . . . 32

2.5 Classification methods . . . 32
2.5.1 Monolithic classifiers . . . 33
2.5.2 Combinations . . . 34
2.5.3 Instance-based . . . 35
2.5.4 Methods comparison studies . . . 35
2.5.5 Advantages and drawbacks . . . 35
2.6 Datasets . . . 36
2.6.1 Discussion . . . 38
2.7 General discussion . . . 38
2.7.1 What is the least imaging effort to make? . . . 38
2.7.2 How to define the problem with respect to the taxonomy tree? . . . 38
2.7.3 Is image-based insect recognition more than an application of image classification? . . . 39
2.8 Research directions . . . 40

3 Fundamentals **41**
3.1 Foreword . . . 41
3.2 Neural networks . . . 41
3.2.1 Biological inspiration . . . 41
3.2.2 Models . . . 42
3.2.3 Training neural networks . . . 45
3.2.4 Towards differential programming . . . 47
3.3 Graphs and graph comparison . . . 48
3.3.1 Graphs . . . 48
3.3.2 Graph comparison methods . . . 49
3.3.3 Graph comparison through matching . . . 50
3.3.4 Solving graph edit distance . . . 51
3.3.5 Solving error-tolerant graph matching problems . . . 52
3.4 About the manuscript . . . 53
3.4.1 Description . . . 53
3.4.2 Common methodology . . . 53

4 Convolutional neural networks and transfer learning **55**
4.1 Introduction . . . 55
4.2 Problem definition . . . 56
4.2.1 Image-based arthropod classification . . . 56
4.2.2 Transfer learning . . . 56
4.3 State of the art . . . 57
4.4 Proposed approach . . . 57
4.4.1 Transfer learning adapted VGG-16 architecture . . . 58
4.4.2 Data preprocessing . . . 58
4.4.3 Learning rule . . . 59

4.4.4 Hyper-parameter optimization . . . 60
4.5 Experimental work . . . 60
4.5.1 Data . . . 60
4.6 Follow up . . . 60

5 Graph perceptron **61**
5.1 Introduction . . . 61
5.2 Problem statement . . . 62
5.2.1 Parameterized graph matching . . . 62
5.2.2 Graph classification problem . . . 63
5.2.3 Fixed parametrized graph matching for machine learning . . . 64
5.3 State of the art . . . 64
5.4 Proposal: a multiclass graph-based perceptron . . . 66
5.4.1 The perceptron choice . . . 66
5.4.2 2-class graph-based perceptron . . . 67
5.4.3 Multiclass graph-based perceptron . . . 68
5.4.4 Complexity analysis . . . 69
5.4.5 Discussion . . . 69
5.5 Experiments . . . 70
5.5.1 Experimental setting . . . 70
5.5.2 Results . . . 72
5.6 Conclusions and Perspectives . . . 73
5.7 Follow up . . . 74

6 Graph convolutional neural networks **75**
6.1 Introduction . . . 75
6.2 Review of the literature . . . 76
6.3 Proposal . . . 78
6.3.1 Notation . . . 78
6.3.2 Graph matching . . . 78
6.3.3 Graph convolution based on graph matching . . . 79
6.3.4 Convolution layer . . . 80
6.3.5 About graph matching differentiation . . . 84
6.3.6 A "no edge matching" version of the graph convolution layer . . . 84
6.3.7 Graph pooling . . . 85
6.3.8 Hyperparameterization . . . 85
6.3.9 Choosing the graph matching solver . . . 86
6.4 Experimental work . . . 86
6.4.1 Baselines . . . 86
6.4.2 Data . . . 87
6.4.3 Parameterization . . . 87
6.4.4 Protocol . . . 88

6.4.5 Results . . . 89
6.5 Conclusion and perspectives . . . 90
6.6 Follow-up . . . 91

7 Experimental results on insect recognition **93**
7.1 Introduction . . . 93
7.2 Datasets . . . 93
7.2.1 IRBI . . . 94
7.2.2 ImageNet-arthropods . . . 95
7.3 Models . . . 95
7.4 Model parameterization . . . 96
7.5 Protocol . . . 96
7.6 Results . . . 97
7.6.1 Experiment 1: Euclidean classification . . . 97
7.6.2 Experiment 2: Advanced study on transfer learning . . . 97
7.6.3 Experiment 3: Graph classification . . . 99
7.6.4 Experiment 4: Misclassification analysis . . . 99
7.7 Conclusion . . . 100

8 Conclusions and perspectives **101**
8.1 Conclusion . . . 101
8.2 Perspectives . . . 102
8.2.1 To go further . . . 102
8.2.2 Other potential research directions . . . 103

List of publications **105**

Bibliography **107**

List of Figures

1.1 Evolution of number of layers (those with weights) along time in neural networks . 20
1.2 The taxonomic scales relevant to arthropod classification 21

2.1 Lab-based samples 25
2.2 Field-based samples 25
2.3 Multi-individuals samples 25
2.4 Feature extraction evolution [164] 27

3.1 Perceptron visualized as of its composing functions 42
3.2 Neuron-wise visualization of a multi-layer perceptron 43
3.3 Visualizing convolution Dumoulin and Visin [83] 46
3.4 A convolutional layer with three 3×3 layers. Image and kernels are in blue and violet (blue and violet respectively denote first and second channels) and the resulting feature maps are in green [83] 46
3.5 Example of computational graph for a neural network 48

4.1 Layer translating volumes into 1-D vector to take as input of the fully-connected end of a CNN 58
4.2 Transfer learning architecture with modified VGG-16 59

5.1 Illustration of the parametrized score function computation. 63
5.2 Parametrized matching function where G_1 has 7 graph components (4 nodes and 3 edges). 65
5.3 Overview of the perceptron and our proposal of a modified perceptron for graph . 67
5.4 Letter-HIGH : Impact of the learning rate on the convergence. 73

6.1 Computing graph convolution 81
6.2 Network structure used for graph convolution experiments 87
6.3 MNIST graphs. Left is $\frac{1}{4}$ grid, right is 75 superpixels RAG. Red symbolizes vertex frontiers and green shows edges. 88
6.4 MNIST graph conv examples (respectively original, convoluted and rotated convoluted versions) 90

7.1 IRBI image examples 94

7.2 ImageNet-arthropods image examples 95
7.3 Network structure used for graph convolution network 96
7.4 Learning curves for VGG-16 (blue for train accuracy, orange for valid accuracy). . 97
7.5 Recognition rates on the testing set depending on the number of training examples and the number of layer finetuned 98
7.6 Misclassified samples. Left big image is the query image and right small images are the 5-nearest neighbors. 100

List of Tables

1.1 Complementary properties of representation spaces [241] . . . 21

2.1 Taxonomy of reviewed papers on the basis of image capture setting . . . 26
2.2 Taxonomy of the reviewed papers on the basis of segmentation . . . 28
2.3 Feature taxonomy for insect recognition . . . 29
2.4 Taxonomy of the reviewed papers on the basis of classification . . . 33
2.5 Recognition rates in comparison studies (bold figure for best rate) . . . 36
2.6 Datasets used in the selected studies . . . 37

5.1 Papers published about learning graph matching . . . 66
5.2 On the left side, summary of graph data set characteristics. On the right side, best parameters according to learning strategies R-1NN and C-1NN taken from [244] and from [201] . . . 70
5.3 Best parameters according to learning strategies R-1NN and C-1NN taken from [244] and from [201] . . . 70
5.4 Taxonomy of the compared methods . . . 71
5.5 Classification results. The best classification rates are marked in blue while the best processing times are in red. "Best" means an improvement over other methods statistically significant according to a z-test with a confidence level of 70% . . . 73
5.6 Classification analysis. The top-2 of deviations are marked in blue while the 2 best time ratios (Time) are in red. Accuracy deviations (Acc) are in % (The lower the better). Time ratio is the best when equal to one (The lower the better). . . . 74

6.1 Frequently used notations . . . 78
6.2 Different MNIST-based graph datasets . . . 87
6.3 MNIST representations . . . 89
6.4 Recognition rates on MNIST 2class . . . 89
6.5 Recognition rates for different neighbourhood sizes on MNIST reduced 2 class . . . 89
6.6 Epoch durations on MNIST 2class (Models use different implementations/hardware: CNN is Keras on GPU, MoNet is Theano on GPU and Ours is Keras on CPU) . . . 91

7.1 The image datasets . . . 95
7.2 Recognition rates on 5-fold experiments . . . 98
7.3 Recognition rates on 5-fold VGG16 transfer learning and mean epoch times . . . 99

7.4 Graph representations in use for insect recognition 99
7.5 Recognition rates on graph cnn experiments 99

8.1 Comparisons of contributions proposed in this PhD thesis 102

Chapter 1

Introduction

Building accurate knowledge of the identity, the geographic distribution and the evolution of living species is essential for a sustainable development of humanity as well as for biodiversity conservation.

1.1 Insects and biology

1.1.1 What insects are and why they are important

Insects are a class of invertebrates within the arthropods that have a exoskeleton, a three-part body (head, thorax and abdomen), three pairs of jointed legs, compound eyes and one pair of antennae. This is a subclass of the Arthropods class which also includes Arachnids, Myriapods and Crustacean and they have been the dominant component of animal species diversity for all of the past 520 million years, since the main burst of the Cambrian radiation. They have been considered to be the earliest colonizers of land [222, 175]. Fossil remains demonstrate that Arthropod species are robust over long periods of time, and that, given the opportunity, they migrate with changing conditions rather than evolving new species [84]. Arthropods are of exceptional value in the reconstruction of paleoenvironments because they provide detailed, precise information on vegetation, soils, water quality, vertebrate species composition, forest composition and degree of stress [84]. Arthropods represent up to 80% of animal phylum [213] and make up for the largest proportion of species richness at any spatial scale [127]. With 1,5 millions of species, they are more representative for wholesale organism biodiversity than any other group. Arthropods have been recognized as efficient indicators of ecosystem function and recommended for use in conservation planning [256, 154, 95, 300]. Many researchers have assessed habitat quality and measured habitat differences using Arthropods within different ecosystems as well as agricultural, forest and urban landscapes [211, 223, 264, 151, 116]. More specifically, arthropod diversity has been used to indicate the impacts of habitat modification [154, 6, 161] to measure the effects of human disturbance [149].

1.1.2 How insects are visually identified by biologists

Insect are more or less easy to discriminate depending on their taxa. Whereas it is obvious to discriminate a bee from a wasp. It is way more tedious to discriminate between asian and european hornet for example. For those more tedious discrimination tasks, biologists came up with step-by-step identification methods. These are called taxonomical keys and consist of decision trees. Experts have to recognize features on the unknown specimen following these decision trees until they reach an answer.

1.2 How computer science can help with insect identification

Species identification of Arthropods is a fundamental part of recognizing and describing biodiversity. Traditionally, identification has been based on morphological diagnoses provided by taxonomic studies. Only experts such as taxonomists and trained technicians can identify taxa accurately, because it requires special skills acquired through extensive experience.

However, the number of taxonomists and other identification experts has drastically decreased. In some cases a trained technician could make routine identifications using morphological "keys" (step-by-step instructions of what to look for), but in most cases an experienced professional taxonomist is needed. The use of Arthropods as biotic indicator is slown down by the high level of expertise required to identify insects. It is an obstacle to a broad and easy application [35]. Consequently, alternative and accurate identification methods non-experts can possibly use are required.

Finding automatic methods for such identification is an important topic with many expectations. The first element to be determined is the digital method to describe our individuals. Several data representations are available: several works were proposed using a variety of these data types such as 3D images [267], sound [49, 229], genomic data [15], etc.

Due to its reduced cost and easy capture process, images are an accessible source of data that suits almost any use case very well. Images of arthropods can be acquired and further processed by an image classification system. Such images also have some specificity and might require adaptation of existing image recognition methods, or new methods.

1.3 About artificial intelligence

Our goal is to replace a manually performed task by a program. For an unknown specimen, this program will output its taxon (or namely *class*) given an image of this specimen. This type of goal calls for computer science and most peculiarly for artificial intelligence. Artificial intelligence, according to [224], is the study of the design of intelligent agents. An intelligent agent is an entity which acts according for its circumstances and goals, more particularly to fulfill a given task. Artificial intelligence is opposed to natural intelligence: tasks to be fulfilled were often done by natural entities such as human beings.

1.3.1 Pattern recognition

Pattern recognition is the engineering discipline which consists in designing programs that are able to identify patterns in data and make a decision based on those patterns based on a given task to be fulfilled.

1.3.2 Computer vision

We are more precisely interested in reproducing vision-related tasks such as detecting or identifying objects in the environment. Computer vision is the subdomain of computer science that is insterested at designing intelligent agents which operate on vision tasks. Formally speaking, [226] describes computer vision as infering on a state of the world y (originally w in the book) from visual data x. x can be of different types (3D or 2D images), so as y:

- In case of regression, y is the value of a given variable: $y \in \mathbb{R}$).
- In case of object detection, y is a list of object instances and their types (classes), bounding

boxes sizes and positions $(\text{posx}, \text{posy}, w, h)$: $y \in \mathcal{C}^n \times \mathbb{R}^{4\times n}$, with n the number of objects in the environment.

- In case of image classification, y is the class of the image: $y \in \mathcal{C}$.

$\mathcal{C}$ is the set of object types (or classes).

In order to solve a task using a computer program, it is necessary to design an algorithm able to model and execute the expected behaviour. Most classical computer science tasks (as typical as sorting a list of numbers) are solved using explicit intructions (for which the reasoning is well defined). This is the path that has been followed for solving computer vision problems. The classical reasoning is to extract meaningful information (commonly called features) from images. Features rely on observed properties of images such as shapes, colours or textures. They can be extracted at a local or global level. A more thorough study of such features is described in Chapter 2 Section 2.4. These features are then used as a basis to operate on visual data.

1.3.3 Machine learning

Some algorithms rely solely on reasoning to fulfill their purpose. For instance, historical segmentation algorithms (such as watershed) do follow logical postulate to separate a subject in image from its background. But these algorithms often turn out to be too simplistic. For other tasks, no existing logical steps even seem to exist to fulfill them. This is the case for image classification. The key to solving image classification problems is to rely on example data (already labelled images set). Image classification algorithms are then designed to learn to classify based on these already labelled images.

This brings us to the area of computer science which deals with learning tasks: machine learning. Machine learning can indeed be defined as the study and design of algorithms that can learn fulfilling a task without explicit instructions [25]. Such algorithms rely on infering on available example data. They are generally made up of three components:

- A model: the structure on which is built the decision/action process. The model embeds the learnt knowledge as its parameters.
- A criterion: a measure of how well an agent (here: the model) behaves.
- A learning algorithm: the optimization method that leads to getting relevant parameters according to the criterion (parameters that allow the algorithm to behave as needed).

The classical generic computer vision pipeline is built by associating a form of feature extraction with a set of these three elements: data is processed through feature extraction and the resulting features are learnt upon with a given model optimized to maximize a given criterion.

1.3.4 About deep learning

Apart from these algorithms, example data is needed for the learning process. Since beginning of the century, both computational power and amount of data available increased drastically. This is critical to learning algorithms especially regarding the previous remark: the more data and computational power are available, the more these algorithms can learn. This has led to the generalization of using learning algorithm instead designing handcrafted algorithms. In newest computer vision algorithms, there is mostly no more conceptual difference between extracting features and learning to fulfill the task since both are learnt. In such a context, the term *deep learning* has emerged and is used to designate those type of approaches although it is difficult to give a clear definition of it.

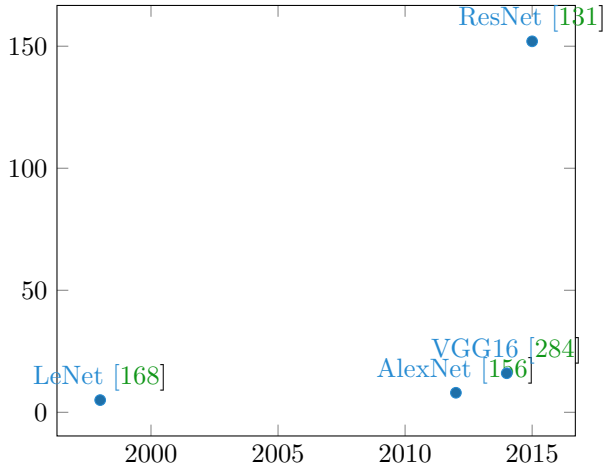


Figure 1.1: Evolution of number of layers (those with weights) along time in neural networks

Originally, *deep* was used to characterize neural networks that had a big number of layers (as opposed to shallow neural networks). The number of layers in neural networks did increase drastically along time (see Figure 1.1) The usage of the term deep learning did begin at the same time. However, deep learning usage has evolved such that it is short-sighted to restrain its definition to the size of models in use.Several definitions have been proposed. Yoshua Bengio's is centered around its goal (« Deep learning is about learning good representations by composition of learned functions. » [23]) while Yann Le Cun's is more about the models and optimization method it uses (« Deep learning is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization. » [163]).

As an alternative attempt, we define deep learning as seeing a task as a learning problem from end to end. Deep approaches indeed tend to minimize the amount of a priori knowledge they rely on, focusing on the learning power of their model.

1.3.5 Structural vs euclidean pattern recognition

A question remains unanswered: How to represent visual data in a computer vision context? To answer this question, we might ask what properties are peculiar of images: these are mainly spatiality and color. This is why 2D color images are usually seen as 3D tensors ($x \in \mathbb{R}^{w \times h \times 3}$). Another way to conceptualize such images is to see them as 3D metrics defined on a finite 2D euclidean space $x : [0, 1]^2 \rightarrow [0, 1]^3$.

Euclidean spaces are regular (with homogenous topology). In more technical terms, this means every pixel on the image has a neighbourhood which is of constant topological structure. If euclidean representations are the most used in computer science, they are not the only ones used. Some type of data cannot fit into the euclidean paradigm because their topology is not regular (such as molecules, networks, 3D meshes, …). Methods that deal with these non-euclidean spaces are called structural because since topology is neither fixed nor regular, it can express the data as well as the metric values does.

However, this expressivity ability comes with a price: euclidean spaces are particular cases of non-euclidean spaces (a 2D image can be defined as a grid graph or a regular manifold). As a consequence, complexity of processing defined on euclidean can be optimized since topology is constant. There is therefore a trade-off between data expressivity (representational power) and efficiency [241] as depicted in Table 1.1.

Table 1.1: Complementary properties of representation spaces [241]

	Euclidean	**Non-euclidean (graphs, manifolds, ...)**
Representational power	Low	High
Efficiency	High	Low

1.4 Problem definition

Image-based arthropod classification could be seen as an application of image classification. Given a photograph depicting the specimen, its biological identity is to be determined. The peculiarities of the problem are three-fold : taxonomy, image variations and incrementality.

Arthropod species, as for any living species, are grouped into taxons at several scales. These scales constitute the biological taxonomy which can be seen as a tree grouping species based on genetic similarity. In Figure 1.2, the scales relevant to arthropod classification are depicted. The biologists might be willing to use a scale or another depending on their scopes and applications. For instance, biodiversity studies need very fine information (down to species precision) and, conversely, pest management applications require less precise information (say to a genus information). The problem can therefore be seen differently along granularity in the taxonomy.

Aside from classical object image variations (such as rotation, scale, parallax, background or lightning), insect images have more particular properties such as pose (because specimen appearance varies with the orientation they are been shown) and deformation (because the specimens are most of the time composed of articulated parts). These aforementioned variations can be referred to as *capture-specific* variations in the sense they only depend on capture factors. About the objects themselves (*object-specific* variations), age, sex and genetic mutations are the main factors of visual variations. The most instructive example is that of lepidoptera (commonly referred to as butterflies) which can have extremely different visual aspects along time (being successively caterpillars, chrysalises and butterflies).

Finally, given the considerable diversity amongst arthropods, classes are highly numerous and, besides, new classes remain undiscovered. This property can be seen as incrementality.

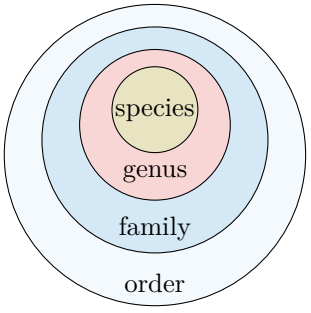


Figure 1.2: The taxonomic scales relevant to arthropod classification

1.5 Manuscript organization

This thesis tackles the insect image identification problem as follows: In Chapter 2, a thorough review of the litterature is proposed and sets research directions. Chapter 3 presents fundamental notions that are used in this thesis contributions. Chapters 4, 5 and 6 present the aforementioned contributions and Chapter 7 presents image datasets and results.

Chapter 2

State of the art in insect image recognition

Contents

2.1 Introduction . . . 24
2.2 Scope and methodology . . . 24
2.2.1 Scope and previous surveys . . . 24
2.2.2 Methodology . . . 24
2.3 Image capture settings and apparatuses . . . 25
2.3.1 Advantages and drawbacks . . . 27
2.4 Feature extraction . . . 28
2.4.1 Foreground/Background detection . . . 28
2.4.2 Three general feature extraction methods . . . 29
2.4.3 Fixed/Handcrafted Feature Extractor . . . 29
2.4.4 Mid-level features . . . 31
2.4.5 Learning Hierarchical Representations . . . 31
2.4.6 Advantages and drawbacks . . . 32
2.5 Classification methods . . . 32
2.5.1 Monolithic classifiers . . . 33
2.5.2 Combinations . . . 34
2.5.3 Instance-based . . . 35
2.5.4 Methods comparison studies . . . 35
2.5.5 Advantages and drawbacks . . . 35
2.6 Datasets . . . 36
2.6.1 Discussion . . . 38
2.7 General discussion . . . 38
2.7.1 What is the least imaging effort to make? . . . 38
2.7.2 How to define the problem with respect to the taxonomy tree? . . . 38
2.7.3 Is image-based insect recognition more than an application of image classification? . . . 39
2.8 Research directions . . . 40

2.1 Introduction

This chapter offers an overview of the existing approaches to image-based insect identification. It is structured as follows: Section 2.2 describes with more precision our scope and the methodology used to analyze the literature. Section 2.3 considers the first step of all approaches, i.e., image acquisition. Section 2.4 is devoted to features extraction, some of them being specific to arthropods. Section 2.5 describes the classification of the data into known classes. Section 2.6 details the existing datasets and, finally, Section 2.7 provides a discussion on this domain, with some conclusions and perspectives.

2.2 Scope and methodology

2.2.1 Scope and previous surveys

The kind of animals we deal with are arthropods. In a vernacular manner, we will abusively use the term insects to denote these animals. The data acquired from these animals are 2D images. We are especially interested in methods that can process 2D images with "normal lighting and shooting" conditions, that is when images are acquired by standard cameras (even mobile or embedded devices). Finally, we consider the systems that extract features from images and that perform a classification of the animals into known classes.

Two previously published surveys are relevant to these scope and aims. In [113], the authors take the general point of view of species identification and discuss the current state of the field and possible perspectives. They underline the key difficulties to solve this problem and the reasons why it is yet no so much developed. They discuss examples of successful achievements that were initially made and provide motivations for further studying this problem. In [10], the authors list a few papers from recent years whereas we provide more references and a common framework to analyze and compare them. We adopt a more focused and detailed level of description than in [113]. Our approach is also more centered on computer science point of view than the cited surveys. Consequently, it can be very useful for researchers in Pattern Recognition field. Furthermore, final discussions provide valuable questions that could be the starting points for researcher who intend embarking on automatic arthropods identification. Also, this state of the art will help positionning the thesis and will suggest research directions.

2.2.2 Methodology

To describe this state of art on image-based recognition of insects, we used the following methodology. Starting from a set of fourty-four papers, we described each paper along several analytic points of view. These analytic points of view are related to a classical image recognition system. Such a system can be decomposed into three sequential phases: a) Image capture; b) Feature extraction and c) Classification. Image capture consists of the tools and methods used to take photos of the insects (discussed in Section 2.3). Features extraction consists of extracting useful visual information out of the pictures (discussed in Section 2.4). Finally, classification is the actual step at which recognition is performed based on the extracted information (discussed in Section 2.5). The datasets are the images used to train and test the system (discussed in Section 2.6).

A set of criteria were chosen to analyze the literature from different perspectives and summarize it into useful information for further use. Are images capture with or without constraints? In the wild or in the lab? Are features domain-dependent or conventional from the computer vision community? What are the classifiers used in the literature? How is the performance evaluation carried out? What are the number of classes or the data set size? Are data sets publicly available

or not ? All these criteria can be seen as different dimensions or angles to categorize, summarize and identify relationships between approaches. Once these pieces of information are obtained, the papers were clustered based on them. Both dimensions and clusters are further exposed along the next sections.

2.3 Image capture settings and apparatuses

A first categorization which can be made is based on the type of image capture setting and on the context of the acquisition. In fact, the acquisition mode might influence the techniques used for the identification of the insects, but also the context of application of such algorithms. In this context, several analytic dimensions can be considered. The first is about acquisition conditions, that basically deals with the insects (both in the way they are captured and taken care of) and the way the images are produced (the capture tools and the overall protocol). The second is what we name as pose constraint, that is how much we allow the point of view on the insect to be variating. A constrained pose setting specifies we want to see, for instance, only the front side of the insect we capture the image of. An unconstrained pose setting, in the contrary, induces variability in orientation of the individual and therefore in their apparent shapes, details and colours. Next, closely related to the pose, the part or area of the individual is to be considered. Some insects are better recognized from their wings as others from their entire body.



(a) [304]



(b) [305]

Figure 2.1: Lab-based samples



(a) [321]



(b) [196]

Figure 2.2: Field-based samples



(a) [112]



(b) [228]

Figure 2.3: Multi-individuals samples

Therefore we have divided the considered works into two broad categories: lab-based setting and field-based setting. In a lab-based setting there is a fixed protocol for image acquisition. This protocol governs the insect trapping, its placement and the material used for the acquisition (capture sensor, lighting system, etc.). Lab-based setting is mainly used by entomologists bringing the insects to the lab to inspect them and to identify them. Therefore, they have the possibility to manage the image acquisition system. By field-based setting we mean a capture of insects directly in cultivated fields, without any particular constraints to the images capture system. Typically the acquisition is made by a mobile device and the insect is alive when the picture is taken. The application context of this type of setting is that of a farmer who wishes to identify an insect to know, for example, if it belongs to a devastating species or not. In this context no, or little, knowledge of the identification system is required by the user other than the user interface. At these two categories we add a third, smaller, which includes the works whose acquisition system is such that in a picture there are more insects than one. In such setting, a more complex segmentation phase might be required before the identification to isolate each insect.

A taxonomy of the reviewed papers is shown in Table 2.1. For each paper the taxonomy reports the corresponding bibliographic reference. We will highlight, in the next paragraphs, the most interesting acquisition systems from the considered papers (or a group of papers using similar settings).

In lab-based setting (see examples on Figure 2.1(a) and Figure 2.1(b)), most of the acquisition

Table 2.1: Taxonomy of reviewed papers on the basis of image capture setting

Category Levels			Refs
Lab-based	Manual Positioning	Constrained Pose	[309, 215, 78, 310, 308, 8, 307, 291, 263, 104, 267, 334, 185, 313, 322, 335, 305, 304, 266, 184, 281, 324, 93, 91, 314, 75, 76, 216]
		Unconstrained pose	[29, 159, 173, 189, 228, 290, 312, 311, 314]
	Automatic Positioning		[330, 159, 189]
Field-based			[196, 89, 321]
Multi-individuals			[112, 287, 323, 228]

systems are manually manipulated. In [173, 215, 78, 308, 310, 8, 307, 291, 263, 104, 311, 266, 314] the acquisition is made manually on a microscope slide, with some standard illumination systems (fiber optic light sources, desk lamps, etc). In [173], to change the pose of the insect, an interesting trick has been used: body rotation was achieved by first placing two short nylon fishing lines on either side of the specimen and gently moving a cover slip placed over them by using the tip of the index finger. In [215, 308, 310, 8, 266, 314, 91] some zoom lenses are used to have more details of the insect or to capture only some parts (wing, genitalia, etc.). In [8] live bees captured in the wild are cooled down using an icebox: this makes them immobile for a time sufficient for the recognition image to be acquired without causing the bee long-term harm. Also in [307] active moths were refrigerated to prevent them escaping while being imaged. In [91] a stereoscope is used in order to avoid a view with obstructed parts. In [312] the insects were frozen for 20 min and then randomly and manually placed on a white balance panel. The images were taken under different orientations and two poses (top and side view).

In [330] the main goal of the paper is not the insect identification but the detection of insects in bulk wheat samples. Nevertheless, it is interesting to note that their acquisition system could be used also for insect identification. Samples, that are a blend of wheat kernels and dockage including some grass seeds, are placed in a crate cell, are captured by a color capture device, with a backlight illumination. To reduce reflection, a dome with flat matte white paint inside was placed over the samples.

There are also semi-automatic systems. In [159, 189] the authors propose a semi-automated mechanical manipulation and imaging system that allows the positioning of the specimen and its rotation to obtain the best view for classification. The decision of the view is taken by the human operator. In [267], a similar but more complex system is proposed: a mirror system is providing a split image, with two views of the insect approximately 90° apart. Providing two simultaneous views increases the likelihood of good dorsal views for 2D classification, and it affords greater robustness for 3D reconstruction since the camera viewpoints of the two views are fixed.

In [334, 335, 324] the acquisition system is similar to the above-described ones, but the insect is captured with a classic color digital camera without microscope, so the insect has a "natural" size in the image. In [324] a transparent plastic stick was fixed to a piece of foam and held at the far end of a needle that was inserted into the specimen, in order to eliminate the shadow due to the fill lights used.

Typically in field-based setting (see Figure 2.2(a) and Figure 2.2(b)) there is no particular acquisition hardware and the insects are captured by standard cameras or mobile devices in open field. In [321], even if the acquisition is performed on open fields, pheromone traps are used and cameras are being integrated to the trap setup in order to improve the precision of insect information captured and facilitate the subsequent segmentation phase.

In multi-individuals (see examples on Figure 2.3(a) and Figure 2.3(b)) setting [228, 323, 287, 112], the positioning of the insect is also manual and is similar to the setups described for manual

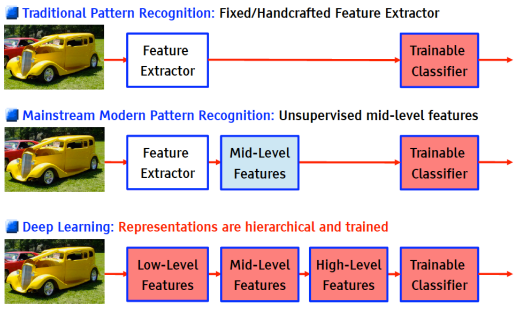


Figure 2.4: Feature extraction evolution [164]

setting in lab-based category. In [228] the rice pests caught in traps are spread on the glass plate with as little overlap as possible, which will make the following image segmentation and identification easier. The acquisition setting is quite standard, but an interesting particularity is that two digital cameras above and below the glass plate are fixed on a stainless holder in order to have multiple poses of the insect without manipulating either the insect or the acquisition system.

In conclusion we can say that in field of insect identification, the lab-based setting is still the most widely used setup and the positioning of the insects is made mostly manually with a constrained pose (see Table 2.1). More recently, the study is done considering several individuals in a single image, but the acquisition system is still largely manual. There are still few works that develop insect identification systems in open field, probably because of a more difficult recognition context (cluttered background, live insect, etc.). In fact in some open-field setting some expedients are used to have an image simple to process.

2.3.1 Advantages and drawbacks

With the large increase of mobile devices, an equivalent growing open-field captures is expected. In fact such a system is more flexible and it can be used by a wider range of users. Open-field captures do not require special hardware and there are no constraints on insect pose. However this kind of systems are more complex in terms of algorithm design and it is hard to obtain good performances.

Anyway, lab-based settings still remain a configuration that will be studied in the future. Indeed, biologists have some specific research that need insect to be studied in laboratory. Therefore, while open-field capture will be used by non-expert people, lab-based setting will continue to grow because of specific need of biologists. The advantage of lab-based setting, from the point of view of the classification system, is the possibility of constraining the insect so that the recognition task is less complex thanks to less variability. Nevertheless, the main drawback of lab-based setting is the manual manipulation of the capture system (posing of the insect, illumination fixing, etc.). This manual task is very tedious and time-consuming and researcher ask more and more for automatic manipulation systems.

In order to speed up the process of recognition, multi-individuals setting are conceived. In this case, the time is reduced, because of multiple classification in a single capture. However, again the classification system has to deal with more image processing problems, like detection of insects that is a difficulty not present in original lab-based setting.

2.4 Feature extraction

Focusing on feature extraction, we listed the two following aspects: the segmentation which is the data extraction phase such that the insect is isolated (foreground/background detection) and the types of features used (whether they are local, global and what information they model).

2.4.1 Foreground/Background detection

Before the features can be extracted, the system needs to know which zone of the image is important. This part is taken care of by segmentation. This segmentation problem was tackled in several ways, as listed in Table 2.2.

Table 2.2: Taxonomy of the reviewed papers on the basis of segmentation

Category Levels		Refs
No segmentation		[330, 321, 266]
User segmentation		[307, 304, 305, 281, 263, 216, 215, 104, 78]
Thresholding	Manual or static	[29, 76, 91, 112, 196, 286, 291, 304, 313, 314, 322, 324, 334, 335]
	Clustering	[89, 196, 286, 311, 334, 335]
	Edge detection	[173, 196, 286, 304]
Other	Image apriori	[8, 93, 291]
	Background subtraction	[228, 309, 308, 310]
	Snake	[313, 323]

In some cases, segmentation is learned in a supervised manner as in [330, 321, 266]. [266] feeds the entire image to the classifier, which is supposed to learn to discriminate only on foreground and therefore to recognize it. In [321], the approach relies on learning on both positive and negative example images. Negative example images are images where there is no insect but only foreground. Learning on such images enables the system to ignore the background.

When segmentation is irrelevant, it can be asked to the user as in [307, 304, 305, 281, 263, 216, 215, 104, 78]. An interface is shown to the user so that he or she can select the region of interest by drawing its outlines. Of course, manual segmentation might be a tedious and time-consuming task considering how numerous the images can get. That is why segmentation is worth automated most of the time.

Some segmentation methods rely on thresholding which is basically splitting the image histogram into several groups which correspond respectively to the object and the background. The simplest way of doing thresholding is to set the intensity value that separates the two groups. This value can be set statically in the program or by the user who can select the one which yields the best result as in [76, 75].

Another way of performing thresholding is to see it as a clustering problem where two or more clusters (which are the regions) have to be formed [89, 196, 286, 311, 334, 335]. Otsu's method criterion is about choosing the clusters such that the intra-cluster variation is minimized while the inter-cluster one is maximized [334, 335, 286]. k-means is used onto the color space to search for centroids representing the different regions in the image based on color similarity [89, 311]. [196] uses ISODATA, a clustering algorithm that builds clusters with a given standard deviation threshold. [334, 335] use meanshift clustering in the color space as a preprocessing step to Otsu.

Some articles build their segmentation over edge detection. Most of them use Sobel filters to get the gradient of the image and use filling operators to get a first mask [196, 286, 304]. [173] uses order-statistic filters to get the edges of the images.

There are other techniques used to segment the images. [313, 323] use active contours (snake)

that take a simple thresholding mask as a seed point to get a more accurate segmentation. [228, 309, 308, 310] use background segmentation (which implies the background to be constant).

Finally some studies base their segmentation pipelines on image apriori. [291] assumes the object in the image yields the longest outline while [8] searches for lines on wing images and [93] uses the symmetry of lepidoptera as a detection criterion.

Table 2.3: Feature taxonomy for insect recognition

Category Levels			Refs
Handcrafted features	Domain-dependent	Wing' Venations	[8, 291, 281]
		Geometry	[304, 75, 76, 305, 104, 324, 330]
	Global and generic image features	Shape	[323, 112, 173, 29, 313, 228, 314, 322, 305]
		Color	[29, 334, 313, 196, 228, 314, 305]
		Texture	[78, 334, 313, 228, 263, 314, 91, 305]
		Raw Pixel	[266]
	Local features	SIFT	[267, 185, 287, 184, 159, 290, 312, 189]
		Others	[196, 93, 321, 335]
Mid-level features	Unsupervised representations	BoW	[267, 159, 290, 312]
		PCA	[215, 216, 309, 308, 310]
	Supervised representations	MLP	[29, 78, 266]
		Sparse Coding	[321, 185, 184, 89]
Hierarchical representations	Auto-encoder		[314]

2.4.2 Three general feature extraction methods

For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. To increase feature performance, learning mid-level features on top of handcrafted ones has been put forward. During the last decade, the amount of human labor has been reduced by representation learning. Representation learning is a set of methods that allows a machine to be fed with raw data (pixels) and to automatically discover the representations needed for detection or classification. Figure 2.4 illustrates the concepts of handcrafted, mid-level and learned features. In Table 2.3, the papers from the literature are organized according the following criteria: a) Handcrafted features; b) Mid-level features; c) Learning Hierarchical Representations.

2.4.3 Fixed/Handcrafted Feature Extractor

2.4.3.1 Domain-depend features

In [8], the features are based on venations present in the bees' wings, and the cells they enclose, to generate a numerical feature vector. Authors in [8] could say the venations in the bees' wings are almost as fingerprints. In the same direction, [291] the coordinates of some characteristic points on the wing are used to compare vein patterns. In [324], owlflies wings are described by Elliptic Fourier coefficients. The feature extraction algorithm incorporates prior expert knowledge about wing veination. More recently, [281] studied the impact of feature selection on top of descriptors presented in [291].

In [75, 76, 305, 304], series of geometrical features including area, perimeter, holes number, eccentricity and roundness have previously been tested. All these features are intuitive because they can be directly measured or simply calculated from images. However, these precisely extracted features are easily affected by factors such as the pose of insects or the shooting angle. Furthermore, it is usually difficult to compute the real size of insects from the images because of the lack of some shooting parameters such as object distance.

2.4.3.2 Global Image features

General image features are generic descriptors that can be used to represent any kind of images not specifically insects. However, in the literature there is distinction between local and global features whether features ares extracted locally on specific area of the image or globally on the whole image.

In [323] Hu moments (Hu), Elliptic Fourier Descriptors (EFD), Radial Distance Functions (RDF) and Local Binary Patterns (LBP) are extracted as shape descriptors. LBP features seem to outperform the rest of the features in recognition rate based on the individual performance of each method. The results from the underlying features are then fused using weighted majority voting to obtain a decision. In [305], a series of shape, color and texture features was developed that draw on CBIR and allow the identification of butterfly images to the taxonomic scale of family. In [89], the objective is to detect pests into an image, and $L^*a^*b^*$ color space is used as features.

In [313, 334, 228, 29] global features are used to describe the entire insect. A set of 54 features including geometric features of insect, contour features, moment features, texture features and color features are extracted. In [112, 173] features were extracted from the shape descriptors computed on the binary mask of the Region Of Interest (ROI). The measurements taken were area, convex area, eccentricity, major axis length, minor axis length, perimeter, solidity, roundness, compactness, extent, aspect ratio. In [78, 91] wavelet decomposition of the images is performed.

2.4.3.3 Local features

A crucial step for identification or classification and retrieval is to describe images by extracting a set of local feature descriptors, encoding them into high-dimensional vectors and then fusing them into image-level signatures. In [196], images are preprocessed to detect insect location and then global color features are measured (color frequency histograms). Finally, local color features are extracted from patches. In [93], the core of the system is a probabilistic model that infers Semantically Related Visual (SRV) attributes from low-level visual features of moth image. The main motivation is to obtain a human-readable representation of the image. In [335], the approximate shift invariant property of the dual tree complex wavelet transform (DTCWT) and its good directional selectivity in 2D make it a very appealing choice for insect image recognition. These local and global approaches are combined: a first distance between images is obtained by conventional region-based bag of words techniques and a second distance is computed through DTCWT. The two distances are then merged together. Finally, the very well-known SIFT (Scale Invariant Feature Transforms) [182] features are extracted in [267, 159, 312, 185, 184, 287, 189]. SIFT features were completed by multiple order gradient histogram (MOGH) in [290]. In the same direction, [321] SIFT features are also completed by with color, texture and shape features. In [189, 287], SIFT features are extracted and directly matched between paired-images. The number of positive putative matches is used to estimate the similarity of the two images.

2.4.4 Mid-level features

In the Digital Automated Identification System (DAISY) [308, 310, 215, 309, 216], images are processed using filters to detect edges (top-hat filer) and then noise is removed by applying a Gaussian blur. It adopts a principal component analysis (PCA) method to acquire image features that contain nearly all the information of an image. PCA features are more suitable for species identification because the great amount of detailed information collected with PCA can weaken differences between high-level categories such as orders. The basic algorithm of DAISY is based on the methods for human face detection and recognition via eigen-images. DAISY requires user interaction for image capture and segmentation, because specimens must be aligned in the images. In Russell et al. [263], SPIDA (Species IDentification Automated) was created as a ground spiders identification system. SPIDA's feature vector is built using images of the spider's external genitalia from a subset of components of the wavelet transform using the Daubechies 4 function [67]. This method also has the drawback that the specimen has to be manipulated by hand, and the image capture, preprocessing, and region selection also require direct user interaction.

There is a large body of literature [267, 159, 312, 290] showing the successful application of the bag-of-keywords approach. In this approach, local-feature data is extracted from the available images and characterized by computing a feature vector called a descriptor. A visual dictionary is created by clustering a subset of the training data in the descriptor space, creating clusters that define dictionary words. This dictionary is then used to map the region descriptors of a novel image into words, and to map the bag of features for an image into a bag of words that is used to construct a representation of the image. Histograms of these words have been successfully used as intermediate image representations. Once an intermediate representation is constructed, the remainder of the training data is used to train a classifier to recognize object classes.

In this series of studies regarding insect identification, the progression of recognition methods from simple pattern recognition approaches to unsupervised and later discriminative dictionaries can be seen.

To improve the classification accuracy, [185, 184, 321] develop an insect recognition system using advanced multiple task sparse representation and multiple-kernel learning (MKL) techniques. As different features of insect images contribute differently to the classification of insect species, the multiple-task sparse representation technique can combine multiple features of insect species to enhance the recognition performance.

Another way to build discriminative mid-level features is to feed a Multi Layered Perceptron (MLP) with handcrafted features as in [29, 78] or directly from the raw pixels as in [266]. However the number of feature abstraction layers is limited to two because of the vanishing gradient problem.

2.4.5 Learning Hierarchical Representations

Over the last few years, a large amount of research on visual recognition has focused on learning low-level and mid-level features using unsupervised learning, supervised learning, or a combination of both. The ability to learn multiple levels of good feature representations in a hierarchical structure helps to construct sophisticated recognition systems.

In [314], series of geometric, shape, invariant moment, texture and color features are given as an input to a Stacked Denoising Autoencoder. This deep learning architecture consists of multiple levels of non-linear operations and is an effective way to represent high-level abstractions. Automatically learning features at multiple levels of abstraction allows a deep learning system to learn complex functions mapping the input to the output directly from data [20]. The goal in [314] is to solve the pose variety problem. Compared with general auto-encoder (AE) architecture, DAE architecture adds the noise to the training data, which means that the DAE learns to detach noise from real data. It makes the encoder learning more robust to the expression of the input signal.

2.4.6 Advantages and drawbacks

The first attempts of image-based recognition [75, 76] used morpho-metrics as features. Morpho-metrics are distances or ratios between venations present on the wings. These morpho-metrics were extracted and used to separate species using Linear Discriminant Analysis. Morpho-metrics being specific to winged species, more generic approaches were to be found. [308, 310, 215, 322, 8] established the DAISY system to classify wasp insect images using PCA. To improve classification accuracy, [216] applied DAISY to recognize insect images by analyzing their wing patterns and shapes. [8] proposed an improved ABIS (Automatic Bee Identification System) system using support vector machine (SVM) and kernel discriminate analysis based on geometric features of wings (such as length, angle, and area). Moreover, many literature works have focused on constructing object appearance models, a key part of object classification. Generally, based on their appearance models, most object feature descriptors can be categorized as either global features or local features. [263] adopted global features (including color, texture, and geometry) to classify insect images and obtained good results using high-quality images. However, because the features are very sensitive to rotation, scale, translation, and viewpoint changes, this classification method did not work well on objects with large intra-species variation or high inter-species similarity. To address these issues, [312] developed a local feature-based insect identification scheme to account for variations in insect appearance. Furthermore, [311] devised an image-based automated insect identification and classification method using three models: an invariant local feature model, a global feature model, and a hierarchical combination model. [287] extended the LOSS algorithm [286] for analyzing the geometrical characteristics of insects to improve insect classification. A drawback is the limited expressiveness of these models because insect appearances can only be represented by hand-crafted descriptors which makes it difficult to handle significant insect view and pose changes simultaneously. For the same reason, some local features based on SIFT such as CFH [159], bag of words [312, 267, 312, 290] or ScSPM [185] that are currently used in insect species recognition are not quite suitable in high-level (order level) insect identification. It is well known that natural images can be sparsely represented using a sparse linear combination of a few elements from a trained dictionary. In contrast to most existing insect-classification methods that directly operate on low-level features or cues, sparse coding can learn insect appearances from raw features to quantify insect appearances by means of sparsity [185, 184, 321]

Over the last few years, in [314] the ability to learn multiple levels of good feature representations in a hierarchical structure helps to construct sophisticated recognition systems. The benefit of the sophistication of recognition systems is the release of the imaging constraints. The perspective is therefore about handling more and more noise and variations so that both the capture step gets less and less tedious and the classification gets improved in handling more complex class configurations.

2.5 Classification methods

From the perspective of classification, we used the following pieces of informations: the granularity of the classification (the scale in the taxonomy tree whether it's to a species, families, order or another level precision), the number of classes, classifier(s) used, type of validation and the accuracy of the results.

The original pattern recognition framework assumes there is a clear boundary between extracting information from raw data and classification. The algorithms fulfilling these two tasks are thought as independent processes and seen as blackboxes from one another point of view. However, the boundary is becoming blurrier with some recent work (see representation learning [22] for example). In these approaches, the features and the classifier function are learnt at the same time during the learning stage. However, most of the papers studied in this article keep a sharp boundary between feature extraction and classification. In this section, the classification stage is

Table 2.4: Taxonomy of the reviewed papers on the basis of classification

Category Levels			Refs
Monolithic	Discriminative	Linear	[76, 75, 291, 104, 330]
		SVM	[228, 324, 184, 91, 185, 321, 312, 196, 281, 304]
		Neural nets	[314, 78, 173, 266, 322, 281, 304]
		Decision trees	[196, 281, 159]
	Generative	Parametric	[159, 313, 196, 281]
		Density estimation	[312, 313]
		Other	[309, 215, 308, 310, 89]
Combinations	Committees		[323]
	Boosting and bagging		[267, 159, 196]
	Other		[78, 263, 112, 311]
Instance-based	k-NN		[323, 312, 313, 196, 305, 335, 334, 287]
	Other		[93]

investigated as well as the way it has been applied to insect image recognition until now.

As for the previous sections, we consider the papers under a certain categorization, which is depicted in Table 2.4: a) Monolithic classifiers; b) Combinations; c) Instance-based.

2.5.1 Monolithic classifiers

The basic way to do classification in pattern recognition is to train a single system on the actual data.

Classification is the process of actually recognizing an individual as belonging to a certain population among others. There are two ways to see classification conceptually: discriminatively and generatively speaking. This will be discussed about along the next paragraphs.

2.5.1.1 Discriminative methods

As a discriminative process, classification is setting boundaries on a given feature space between the different populations in order to distinguish them. Stochastically speaking, it is said that discriminative approaches try to model $p(C = c|x)$, i.e. the probability of occurence of class c given x (features vector).

A first approach to discriminative classification is to find a linear boundary between classes. [312, 313] uses the least squares approximation method to find this boundary. In [76, 75, 291, 104, 330], Fisher criterion (Linear Discriminant Analysis) is used to the same end on morphometrics. However, data may not be linearly separable and therefore linear discrimination is likely to fail. In [228, 324, 184, 91, 185, 321, 312, 196, 281, 304] SVMs are in use. While [91, 184, 185, 196] use linear SVM, [312, 228, 324, 281] use kernelized SVMs. [312, 281] use SVMs with the polynomial kernel. [228, 324, 304] use the gaussian Radial Basis Function of standard deviation σ as a kernel. σ is considered as a free parameter in the frame of such a use. The usual way to tune σ is trial and error. However, [124] proposes a genetic algorithm to find this parameter which is used in [324]. [321] uses Multiple Kernel Learning, which is a SVM using an optimally chosen combination of different kernel types in order to efficiently learn on different feature modalities.

As used in [314, 78, 173, 266, 322, 281, 304], neural networks are another discriminative approach where non-linearity is induced through activation functions. In [266], a neural network is trained with scaled conjugate gradient-based backpropagation. Several parameters and network sizes (from no to two hidden layers) are tested on the following recognition task: discriminate a single pest from many non-damaging insects. While a one-hidden-layer network is proven to be

a universal approximator, the authors couldn't get satisfying result with one single hidden layer. A possible explanation is, because the inputs are raw pixel, the MLP must learn a feature level. Finally, [322] uses Quality Threshold ARTMAP, which is an incremental learning network technique. An ARTMAP network is a combination of unsupervised modules (which are self-organizing maps): one operating on features, one on the classes and a last one in between.

Decision trees are used in [196, 281, 159]. [196, 281] use classic decision tree generated with C4.5 algorithm. [189] uses Logistic Model Trees which differ from classic trees in that its nodes are logistic regression classifiers.

2.5.1.2 Generative methods

From a generative perspective, classification is trying to estimate the likelihood for an individual to belong to a given population. To keep using the probabilistic point of view, generative approaches are modeling $p(x|C = c)$, i.e. the probability of occurence of feature vector x if considering the class c.

[189, 313, 196, 281] use parametric probability density functions to estimate $p(x|C = c)$ for each class c. [189, 281, 196, 313] use Naïve Bayes classifiers which assume the features are statistically independent. Logistic classifiers (used in [281]), however, don't postulate on statistical independance. Rather than trying to find a distribution by tuning probability density functions, non-parametric methods rely on the instances themselves. [312, 313] use Parzen-Rosenblatt window classifiers to estimate the probabilities.

Other generative methods are used besides probability density estimation. In its first version, DAISY [309, 215, 308, 310] uses a non-parametric statistical test to evaluate the quality of the PCA reconstructed image with respect to the original image. The test used is Kendall rank correlation coefficient (more commonly referred to as the τ-test). [89] uses a correspondence filter to detect a given shape and localize it on image. It is a frequency domain filter which is computed through an optimisation process such as a linear optimisation program. The idea is to create such a filter so that the peak of correlation and the localisation of the object we search in the image match up.

2.5.2 Combinations

But when it comes to issues such as overfitting, a monolithic classifier is somehow not enough and that is here where ensemble learning is relevant. The idea of ensemble learning is to use a combination of said weak classifiers to get a strong classifier [255]. Classifier combinations also aim at improving simple classifiers performances.

In [323] four k-Nearest Neighbour classifiers are trained on different feature descriptors. Their decisions is combined into a weighted majority vote: it is a committee of k-NN classifiers. [267, 159] combine decision trees in more complex ways. In [159], logistic model trees are boosted with AdaBoost. The classifiers are incrementally trained, each classifier being trained on classification error weighted according to the previous classifier error so that it can learn more where it was weaker. [290, 112] use several classifiers to perform dichotomies on the dataset. In [290], the decision is made by making SVMs compete against each other in a tennis-tournament-like process.

SPIDA [78] not only experiments flat neural network classifiers but also tests multi-granularity discrimination: a first neural net discriminates individuals at the genus level before a second one performs likewise at the species-level in the actual genus. The system is found to have no difficulty to discriminate between genuses but some between species inside different genera. In [263] a different structural approach is adopted: A set of neural nets were trained (one per taxon). Each neural network has two outputs: one that is trained to be stimulated when the individuals belongs to the taxon (known as pro-species output) and the other when it's not (the anti-species output). The final decision is given as follows: the top-3 scores from all the pro neurons are considered and

those who are above a given threshold are considered as answers and shown to the user. In [112], a stacked ensemble of neural networks is used to get satisfying classification results (compared to those of a single neural net classifier). It is in fact a decision-tree where every node decision is given by a neural network. [311] uses a different framework which is based on recognition relevance. If a first classification correlation level is not high enough, a second classification is operated. It stacks up a global feature classifier (based on Bayes classifiers) on top of a local feature classifier (with Nearest Mean classifier) in order to become invariant to pose variabilities. The global classifier is first called and, if the matching probability is too low (under a given threshold), the local feature classifier is called to give the final decision.

2.5.3 Instance-based

The previously mentioned methods rely on characterizing classes in the feature space either generatively or discriminatively. Another way of tackling the classification problem is to compare the unknown image not to classes themselves but to known occurences (which turn out to be the learning images).

[323, 305, 335, 334, 287] use nearest neighbour classification to get a final decision. [305] uses a nearest neighbour classifier with a weighted L_2 norm: the authors want to test the classification within modifying the weights of the different features.[334, 335, 287] base their approaches on refining a candidate list at each level. A first classifier gives a match list based on some similarity measure while a second one refines that list. [334] experimented a two-level classification workflow. The first level is a coarse matching step where a histogram similarity measure is done and candidates are filtered based on a given similarity-threshold value. The second level classification is done by a k-NN classifier which performs on some texture features. [335] first uses an integrated region matching process on clustered image to find k image candidates. The final answer is given by a Nearest Neighbour classifier.

[93] relies on a CBIR approach which is not about getting a final answer but just providing a match list to the user. The decision task falls therefore to them. In the scope of this study, Relevance Feedback is used as a performance booster. The user sends a query and is asked about the answer relevance. If trust has been granted to the user (because he passed a recognition test successfully), the system distorts the feature spaces so that the non-relevant items are further away from each others.

2.5.4 Methods comparison studies

In some studies, features are extracted from the images and the choice of a classifier is not really guided. They usually take a set of classifiers to apply to extracted features and keep the one that maximizes the recognition rate on sample data. Results of some of the tested classifiers are somehow investigated. Table 2.5 details the different methods used in each study and gives the recognition rates per method.

2.5.5 Advantages and drawbacks

Although discriminative methods are powerful tools to accurately split the feature spaces, their lack of incrementality can be a restriction in the scope of broad insect recognition where new species can be encountered. Generative methods are more fitted to this end since the classes probabilities are considered independently. Adding a class is then not a problem.

When it comes to evolving classes (unknown individuals encountered that belong to already existing classes), instance-based are handier since the new individuals can be integrated to the instance. Going further, the nature of insect recognition as a pattern recognition problem can be

Table 2.5: Recognition rates in comparison studies (bold figure for best rate)

	[312]	[313]	[196]	[281]	[304]
Least Square	84.2	83.3			
LDA				64.0	
SVM	88.4		**85.0**	60.0	**92.0**
Decision tree			58.3	36.0	
MLP				58.5	76.0
Bayes		**89.9**	65.9	**65.2**	
Logistic				63.2	
Parzen density	86.3	84.7			
Random forest			83.2		
***k*-NN**	77.4	83.8	71.6		
Nearest Mean	**89.5**	63.6			

called into question. Given the potentially high cardinality of the classes set and the high intra-class variability, it might be more relevant to try to match the unknown individual to a known one than to an entire class. This feature is interesting from the user perspective because they get the closest known individuals instead of the class information. The problem could then be seen more as a metric learning one. In such a frame, the similarity function is to be learnt so that the individuals in a class are close to each others and as far as possible to other classes in the similarity space. The learning process would therefore be class-agnostic in the sense the learning phase only needs to know which individuals pairs are similar and which ones aren't. The class is induced by the individuals and their labels.

2.6 Datasets

To characterize the datasets, we noted the number of instances (images) per class and overall as well as per training, validation and testing subset. The granularity is a good indicator as well. Granularity is the scope in the taxonomy where a study is located. It can be as specific as to a few species in a single family or as wide as over several orders (see taxonomical scales in Figure 1.2). Every dataset used in the studied literature are listed in Table 2.6.

The first comment one can make is that the granularity is often very thin and the variability very subtle. We most of the time talk about species in a few specific orders or families. In fact, studied sets are tightly bound to the projects biological and environment-related problems such as recognizing a very specific group of pests for a given orchard. Some studies are in the frame of Integrated Pest Management [115] Pests for different orchards were investigated such as rice [228], palm [266], apples [29] cotton [112] or miscellaneous vegetables [89, 287, 286] pests. [314, 312, 313, 307, 323] focus on moths. Some others deal with biodiversity assessments like the BugID project [203, 202, 267, 189, 159] and [290] which focus on aquatic myriapods. These insects, when found in rivers, are used to perform bioassessments regarding water quality. [173] does likewise to assess marine life by recognizing copepods populations. ABIS [8] aims at quantifying bees populations regarding their different species. Bees are the main pollinators and are therefore essential for crops to grow.

Some studies deal with coarser problems [304, 305, 29] such as families of butterflies [305] or different orders [304] while some studies test the performances of their methods on different granularity levels [78].

Table 2.6: Datasets used in the selected studies

Insects	Granularity	Number of taxons	Avg. Nb. of images per taxon	Capture type	Constrained pose?	Part of insect	Publicly available	Used in	Best reco rate (* means avg reco rate)
Lepidopterans	species	100	4	lab-based	Yes	Wings	No	[334, 335]	85.00%
Ichneumonidae	species	5	50	lab-based	Yes	Wings	No	[309, 310, 215]	94%
Midges	species	49	20	lab-based	Yes	Wings	No	[308, 215]	86%
Macromoths	species	237	3	lab-based	Yes	Whole	No	[307, 196]	85.00%
Moths, butterflies and caterpillar	species	4500	8	lab-based	Yes	Whole	Yes (http://janzen.sas.upenn.edu/)	[93] (partial use)	53.00%
Leafhoppers	species	3	N/A	lab-based	Yes	Whole	No	[76]	73.00%
Copepods	species	8	30	lab-based	No	Whole	No	[173]	98.00%
Miscellaneous orchard insects (multiple poses)	species	9	162	lab-based	Yes	Whole	No	[312, 311, 314] (partial use) [313]	96.60%
Miscellaneous orchard insects (trap-based)	species	5	128	lab-based	No	Whole	No	[311, 314] (partial use)	94.10%
Tephritidae	species	20	7	lab-based	Yes	Whole or part	No	[185, 184]	88.50%
Many orders	orders	9	123	lab-based	Yes	Whole	No	[305]	32%
Apple pests	species	6	200	lab-based	No	Whole	No	[29]	99.50%
Butterflies	species	9	25	lab-based	Yes	Whole	No	[304],[321]	90.30%
Miscellaneous insects	species	20	12	lab-based	Yes	Whole	No	[322]	99%
Benthic macro-invertebrates	species	8	191	lab-based	No	Whole	No	[290]	88%
Palm tree insects	species	21	20	lab-based	Yes	Whole	No	[266]	95%*
Cotton insects	species	12	N/A	multi-individuals	No	Whole	No	[112]	75%*
Rice insects	species	4	195	multi-individuals	No	Whole	No	[228]	97.5%*
Stoneflies	species	9	425	lab-based	No	Whole	Yes (http://web.engr.oregonstate.edu/~tgd/bugid/stonefly9/)	[189, 267, 159]	96.50%
Ephemeroptera, Plecoptera and Trichoptera	species	29					Yes (http://web.engr.oregonstate.edu/~tgd/bugid/ept29/)		
Spiders	species	13	974	lab-based	Yes	Whole	No	[263]	100%
Lycosidae	species	6	18	lab-based	Yes	Whole	No	[78]	100%
Several pests	species	6	160	multi-individuals	No	Whole	No	[287]	N/A
Field crop insects	species	24	26	field-based	No	Whole	No	[321]	89.50%
Moths	species	N/A	6	field-based	No	Whole	Yes (http://www2.ahu.edu.cn/pchen/web/insectRecognition.htm)	[323]	88%
Several pests	species	N/A	N/A	field-based	No	Whole	No	[89]	N/A
Wasps and dragonflies	species	3	8	lab-based	Yes	Wings	No	[291]	N/A
Honeybees	subspecies	26	70	lab-based	Yes	Wings	No	[281]	65%
Bombus bees	species	8	65	lab-based	Yes	Wings	No	[8]	95%
Various bees	subspecies	5	448	lab-based	Yes	Wings	No	[104]	97.50%
Owlflies	species	7	17	lab-based	Yes	Wings	No	[324]	100%
Fruitflies	species	72	25	lab-based	Yes	Wings	No	[91]	86.20%
Mosquitoes	species	79	100	lab-based	Yes	Wings	No	[91]	80.30%

2.6.1 Discussion

There are almost a dataset per article, in most cases because every study has its focus on both the capture of images and the insect taxons involved. This research method apriori makes sense in a biological context since it is an important element in the biologists point of view. However, this goes against the ability to produce comparable results, as methods are tested on different datasets.

It is therefore necessary to come up with a benchmark dataset. As shown in Table 2.6, most of existing datasets are not publicly available. Even though some are, legal terms are to be determined when it comes to using them as benchmark datasets.

As an attempt to offer a simple solution to this problem, we propose to build a specialized dataset from an existing public one. Details are presented in Subsection 7.2.2 (Chapter 7).

2.7 General discussion

Automated insect identification has been intensively studied over the last three decades, including computer vision-based systems for the classification of insect species.

From all the considered works emerge the following questions:

1. What is the least imaging effort to make in order to get satisfying results?
2. How to define the problem with respect to the taxonomy tree?
3. Is image-based insect recognition more than an application of image classification?

2.7.1 What is the least imaging effort to make?

The first approaches of insect-image classification used very constrained and low-tolerance imaging protocols such as with flatbed wing images. Then came methods using region-of-interest segmentation on plain or weakly cluttered backgrounds. Finally, the plunge towards dealing with complex-background images has yet to be taken although some local methods have been used to this end.

Considering generic image classification, Convolutional Neural Networks have been acknowledged as state of the art in a quite short span of time regarding their high and surpassing performance in terms of both accuracy and invariances. In the set of studies being scrutinized here, none has experimented on-pixel classification further than through classical shallow MLP [266] on one hand. On the other hand, deep hierarchical approaches aren't represented further than Stacked Denoising Autoencoders in [314].

However, important characteristics regarding deep learning in general should be taken into account regarding the insect-image classification problem: learning deep hierarchical representations requires a notably larger set of example images than for shallow learning. Assuming the insect images harvest is a more tedious task than for generic images (which is tedious enough for getting millions of images), this issue is the main limitation to the use of CNNs. Finally, no article applied the transfer learning paradigm in order to take the benefit of a pre-trained deep hierarchical representation. Using knowledge gained on larger but generic dataset such as ImageNet as a learning basis to recognize insects could be a way to tackle the image volumetrics problem.

2.7.2 How to define the problem with respect to the taxonomy tree?

As for any other classification task, the problem has to be modeled considering both the classes and the relations between them. Are the classes close to each other? Do they cover variability?

Does that variability make the classes cross each other? To sum up, what are the variabilities both between the categories (*inter-class*) and inside (*intra-class*) them?

The first intuition is these properties vary along the taxonomy tree (see Figure 1.2). The smaller the scale is, the finer the intra-class variations. Small scale insect recognition problems tend to be fine grained object recognition problems. At this scale, the main issue is the intra-class variations can be as strong as and even stronger than the inter-class ones [312, 311, 314, 159, 263].

Does it go to less fine-grained when we climb up the tree towards a coarser level? Even though macro-taxons are not formed on morphological criteria, the genetical proximity induces likelihood of visual similarity. [78] genus-level results are way superior than the species-level ones. Even further, their hierarchical experiment seems to yield good results: a first set of neural networks identifies the genus and a second set deals with contained species. This assertion has to be taken carefully due to the low number of classes that are tested here.

On another point of view, natural phenomena (biodiversity resulting in inter-class similarities and intra-class differences) bring more difficulties for insect identification at the order level than that at the species level [304]. However, it is most often found that the similarities in species within a single family and the differences among species of separate families are prominent enough to [allow] identification [304]. This assumption calls another one: To make upscaled insect recognition efficient, every sub-class (or sub-taxon) must be collected. Same observation is made in [305] at the family level: higher results are observed for higher taxon coverage.

The macro-groups cannot be considered as uniform populations where the individuals are equivalent. These groups must be learnt by considering their respective subgroups equally.

2.7.3 Is image-based insect recognition more than an application of image classification?

This question could be rephrased as "Are there intrinsic properties to insect images so that insect recognition is a specific computer vision problem in itself?" To provide the beginning of an answer, the characteristics regarding insect recognition must be considered. Aside from classical object image variations, pose is a fairly critical variation type that is tackled in [314]. Deformation is one potential characteristics of these objects due to articulated parts [159, 290]. However, these variations can occur under equivalent forms in other objects categories. To take the example of face recognition, age is an object-specific variation factor which can be compared to the insect age. Regarding the visual statistical and structural properties of such images, apparently no study investigated these aspects.

Observations on the classification perspective emerge from the conclusions in Subsection 2.7.2: taxonomy does not reflect visual similarity at any location. This heterogeneity creates complex classes configurations at a coarse level as previously shown but also brings about the dynamics of a potential evolving classification tool. In a context where specimens are being encountered along time, i.e. the construction of the classes is incremental in some sense, new groups of specimens might appear at any moment. As a consequence on a fine level, class are likely to add up when unencountered taxons appear. A dual assertion can be made at the coarse level: when sub taxons add up and since taxonomy is not heterogeneous, the macro-taxon is incrementally defined by its sub-taxons. Depending on the granularity, the problems can reveal two kinds of incrementality : class add-ups and incremental class definitions. These systemic concerns have not been addressed yet in the literature since the datasets are fixed.

Even though this problem has interesting properties, its singularity regarding pattern recognition and image analysis remains unsure. Besides, generic methods are the most used methods in the recent works and it will not be possible to compare methods as long as the experiments are not performed on a common dataset.

2.8 Research directions

In the light of this review of literature, a two-fold research orientation could be determined:

Firstly, this thesis being an overview of the problem as of 2019, experimenting the current state of art pattern recognition methods is inevitable. Convolutional neural networks (CNNs) outperformed in few years every previous approaches at least in the field of image recognition. CNNs have, to the best of our knowledge, rarely been applied to insect image recognition. It is critical to determine whether it can perform well in our frame of study. Moreover, it is a generic approach, which helped having clues about whether insect images have to be taken care of in specific ways.

Once the state of art tested, an other path was explored with the help of graph-based representations. This path was determined by considering the fact that insects are multi-part objects. These objects can be described as a set of parts (nodes) spatially connected. These graph-based representations were manipulated with analogous tools as for the first research axis: convolution on the graph domain.

Chapter 3

Fundamentals

Contents

3.1 Foreword . . . 41
3.2 Neural networks . . . 41
3.2.1 Biological inspiration . . . 41
3.2.2 Models . . . 42
3.2.3 Training neural networks . . . 45
3.2.4 Towards differential programming . . . 47
3.3 Graphs and graph comparison . . . 48
3.3.1 Graphs . . . 48
3.3.2 Graph comparison methods . . . 49
3.3.3 Graph comparison through matching . . . 50
3.3.4 Solving graph edit distance . . . 51
3.3.5 Solving error-tolerant graph matching problems . . . 52
3.4 About the manuscript . . . 53
3.4.1 Description . . . 53
3.4.2 Common methodology . . . 53

3.1 Foreword

This chapter is about listing and explaining every concepts and notions in use in this thesis work. This chapter goes through introducing two mathematical objects: artificial neural networks and graphs.

3.2 Neural networks

3.2.1 Biological inspiration

Artificial neural networks are bio-inspired models used in pattern recognition to fulfill classification and regression tasks. They are rooted in most vertebrate animals way of analyzing information: more-or-less complicated brains. These brains are mostly made up of neurons working in synergy to process signals.

Neurons are connected through synapses that make the signal pass through either in electrical or chemical form. A given neuron that receives information from other neurons can be activated or not depending of its sensitivity regarding the occuring stimulus. Artificial neurons aim at mimicking those two mechanisms which are aggregation of other neurons signals and activation.

3.2.2 Models

3.2.2.1 Perceptron

The perceptron is a canonical neural network formalised by McCulloch and Pitts [197], Rosenblatt [258] and [257]. It consists of a single neuron. This neuron aggregates weighted input data by summation and passes it through an activation function. Its output is calculated as (3.1) and depicted in Figure 3.1.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1a}$$

$$= f(\mathbf{w} \cdot \mathbf{x} + b) \tag{3.1b}$$

where $\mathbf{x}$ is the input vector (3.1c)

$\mathbf{w}$ is the weight vector (3.1d)

f is the activation function (3.1e)

b is the bias. (3.1f)

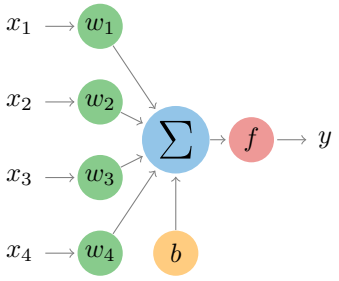


Figure 3.1: Perceptron visualized as of its composing functions

3.2.2.2 Multi-layer perceptron

The main limitation to the perceptron use is due to the fact it is a linear classifier: it cannot solve non-linearly separable problems. To address this limitation, perceptrons neurons are organized in layers. Those layers are then stacked up one on top of the other, the input of the neurons in the k-th layer is composed by every outputs of the neurons in the $(k-1)$-th layer. (3.2) describes how is the output for each layer calculated.

$$y_q^k = f\left(\sum_{i=1}^{I} w_{q,i}^k y_i^{k-1} + b_q^k\right) \tag{3.2a}$$

$$= f(\mathbf{w}_q^k \cdot \mathbf{y}^{k-1} + b_q^k) \tag{3.2b}$$

$$\mathbf{y}^k = f(\mathbf{W}^k \cdot \mathbf{y}^{k-1} + \mathbf{b}^k) \tag{3.2c}$$

where for the kth layer: (3.2d)

$$\mathbf{y}^k \quad \text{is the output vector} \tag{3.2e}$$

$$\mathbf{W}^k = (w^k_{q,i})_{q,i} \quad \text{is the weight matrix} \tag{3.2f}$$

$$\mathbf{b}^k \quad \text{is the bias vector} \tag{3.2g}$$

This type of perceptron stacking is called a multi-layer perceptron (MLP). The input of the network is considered as a layer even though it has no weights (for the sake of recursion). The layers that are neither input or output are called hidden layers. Not only has it the ability to learn to solve non-linearly separable problems, but according to [137] it also has universal approximation abilities: it can approximate arbitrarily closely any continuous function with no more than one hidden layer given enough neurons.

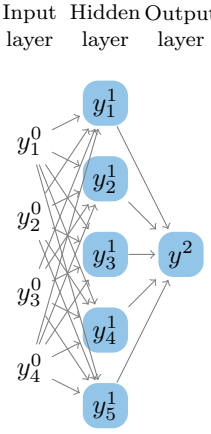


Figure 3.2: Neuron-wise visualization of a multi-layer perceptron

Perceptron-based models can handle n-D vectors. To represent a human being, one can create a 2D vector out of a human's weight and height. It is then possible to learn on that representation and infer based on this piece of data. In other words, a human is represented by a 2D signal defined on a 0D space (i.e. a singularity). Perceptrons, among other classifiers, can only handle signals defined on 0D space.

This is a critical limitation when it comes to images (among other types of data). Images are indeed n-D signals (3D in the case of color RGB images) defined on a 2D space. Hence, it is impossible to apply MLP classification on such data without space transformation: one can for instance express images as a vector of global informations. The consequence of such a representation is the loss of any local information. An alternative is to build a signal defined on a 0D space from the image. This signal will have as many dimensions as there are points on the original 2D space.

This tilt from space to signal dimensions results in the fact that the signal is processed in a different way depending on the location on the space: this transformation is said not to respect the equivariance property because a given pixel value will be interpreted differently depending on the pixel's location.

3.2.2.3 Convolutional neural networks

Convolutional neural networks (CNNs) are pattern recognition models that can learn on signals defined on any euclidean spaces. The most frequent case is 2D images but it can be generalized to process signals defined on n-D spaces such as videos, volumic (3D) images, sound, etc. Neurons

base their aggregation function on the convolution operator. Convolution is an operation on two signals f and g that generates a third signal which is the summation of every product between g at a given location and f at every location. This can be seen as sliding f on g. This operation has been defined by [136] for 1D signals on 1D space (continuous or discrete) as in (3.3).

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t)g(t)\,\mathrm{d}t \tag{3.3a}$$

$$(f * g)(x) = \sum_{m=-\infty}^{+\infty} f(x-m)g(m) \tag{3.3b}$$

Convolution can be easily generalized to 1D signals on N-D spaces $f, g : \mathbb{R}^N \to \mathbb{R}$ as stated by (3.4): f is slided on g along every dimensions of the space at the same time.

$$(f * g)(\mathbf{x}) = \sum_{\mathbf{m}\in\mathbb{R}^N} f(\mathbf{x}-\mathbf{m})g(\mathbf{m}) \tag{3.4}$$

There are two known ways to extend convolution to multidimensional signals $f, g : \mathbb{R}^N \to \mathbb{R}^C$. The first one is, considering each dimension of the signal as independent 1-D signals, to convolve them and pointwise multiply these 1-D convolved signals. This extension (described in (3.5)) is called depthwise separable convolution.

$$(f \circledast g)(\mathbf{x}) = \prod_{c=1}^{C} (f * g)(\mathbf{x}, c) \tag{3.5a}$$

$$= \prod_{c=1}^{C} \left[\sum_{\mathbf{m}\in\mathbb{R}^N} f(\mathbf{x}-\mathbf{m}, c)g(\mathbf{m}, c) \right] \tag{3.5b}$$

The second way is the most common way of defining convolution on multidimensional signals. This approach considers the signal as a whole by replacing multiplication of the two 1D values by dot product between signals value vectors (as shown in (3.6)).

$$(f * g)(\mathbf{x}) = \sum_{\mathbf{m}\in\mathbb{R}^N} f(\mathbf{x}-\mathbf{m}) \cdot g(\mathbf{m}) \tag{3.6}$$

Since this work focuses solely on 2D images, let us define the convolution operator for image signals $f, g : \mathbb{R}^2 \to \mathbb{R}^C$. Although previous theoretical context could allow to work on infinite spaces, images have bounded sizes: f is the image to convolve and is of sizes (W_f, H_f). g is the slided signal, also referred to as the convolution kernel. Its sizes are (W_g, H_g) The classical multidimensional signal convolution will be used here although depthwise separable has also been used on images in the literature.

This operation is calculated as in (3.7) and Figure 3.3(a). A noticeable effect is that the output space size is smaller than the input space size. This is due to the fact the kernel cannot be entirely slid over the image as there is no value beyond the edge of the image to be mapped with the kernel coefficients. An example would be computing the convolution at the upper left corner pixel of the image: the upper left corner pixel would have no value to be multiplied with.

To solve this limitation, it is possible to extend the image space so that the kernel can be slid sufficiently enough to cover the entirety of the image space and therefore preserving input-output size consistency. This solution is called padding (shown in Figure 3.3(b)). The most straightforward approach to padding is padding with zero value. This is also the most logical way of padding in the scope of CNNs because when part of the kernel is mapped with an external zero, that means that part of the kernel did not interact at all and therefore its contribution to the output signal is zero.

$$(f * g)(x, y) = \sum_{x'=0}^{W_f-1} \sum_{y'=0}^{H_f-1} f(x - x', y - y') \cdot g(x', y') \tag{3.7}$$

Convolution has been introduced in neural network architectures by [166]. A convolutional layer consists of a kernel (also called filter) bank. Each filter is convolved over the image and the resulting signal is part of the output. If the number of filters in the layer is p, the output signal will be p-dimensional (see Figure 3.4).

3.2.3 Training neural networks

3.2.3.1 Problem definition

Training a neural network can be defined as the following optimization problem:

Let $\mathcal{L}$ be a function such that $\mathcal{L}(f(X, W), Y)$ is the error between the observed and expected inputs (resp. $f(X, W)$ and Y). Training a neural network on (X, Y) is equivalent to optimizing (3.8).

$$\min_W \mathcal{L}(f(X, W), Y) \tag{3.8}$$

$\mathcal{L}$ is also referred to as the loss function. Depending on the task on which the network is trained, it can be mean-squared error (to solve regression problems), Kullblack-Leibler divergence, , categorical cross-entropy (for classification tasks), etc.

3.2.3.2 Gradient descent

Now that the optimization context has been set, an optimization method must be chosen. The classical way of training neural networks is through gradient descent [261]. Every parameter w contribution in the loss $\mathcal{L}$ has to be evaluated. This is done by computing the partial derivative of the loss criterion with respect to this parameter $\frac{\partial \mathcal{L}}{\partial w}$. Now the weight can be updated more or less by tuning a free parameter α called the learning rate (3.9).

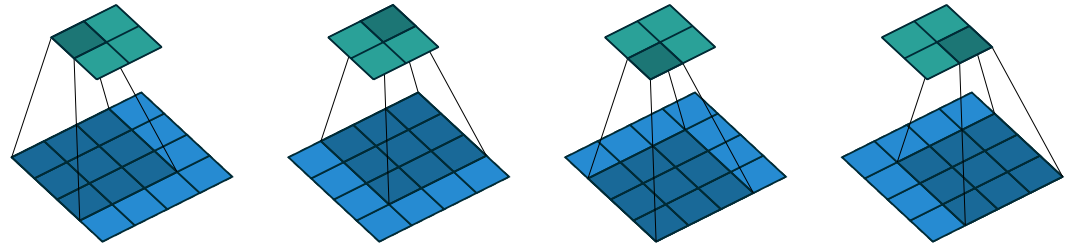
$$\hat{w} = w - \alpha \frac{\partial \mathcal{L}}{\partial w} \tag{3.9}$$

In the case of multi-layer neural networks, f is defined by composition of functions (corresponding to each layers and activation functions). Let's take the example of a two-layer network (with layers f_1, f_2 and f_3 with respective parameters W_1, W_2 and W_3):

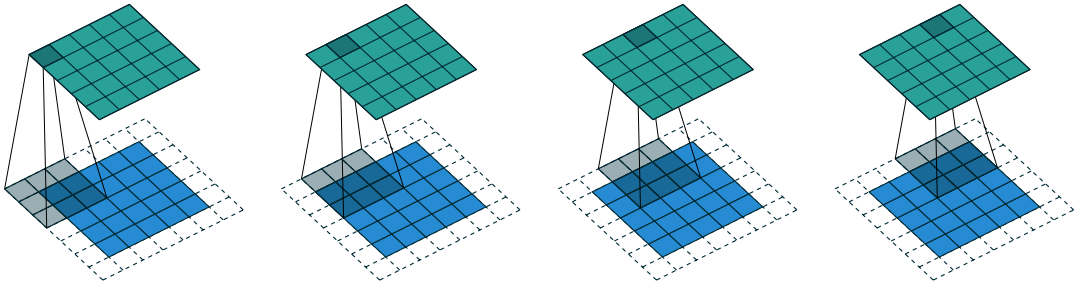
$$f = f_3 \circ f_2 \circ f_1 \tag{3.10a}$$

$$f(X) = f_3(f_2(f_1(X, W_1), W_2), W_3) \tag{3.10b}$$

According to the chain rule of derivative $\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$,



(a) Convolving a 3x3 kernel over a 4x4 image with no padding



(b) Convolving a 3x3 kernel over a 5x5 image with dimension-preserving padding

Figure 3.3: Visualizing convolution Dumoulin and Visin [83]

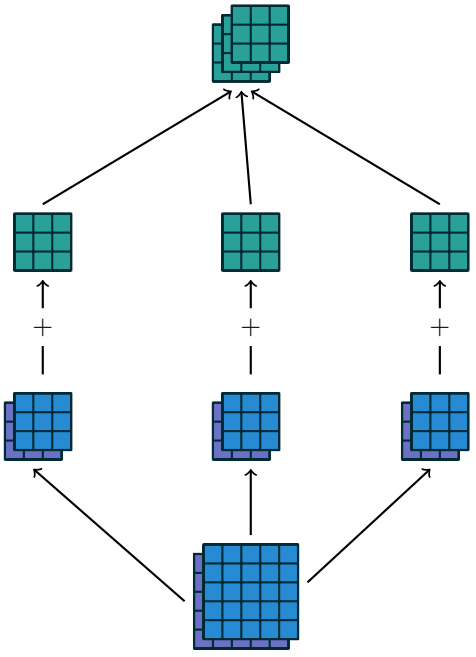


Figure 3.4: A convolutional layer with three 3×3 layers. Image and kernels are in blue and violet (blue and violet respectively denote first and second channels) and the resulting feature maps are in green [83]

$$\frac{\partial \mathcal{L}}{\partial W_3} = \frac{\partial \mathcal{L}}{\partial f_3} \cdot \frac{\partial f_3}{\partial W_3} \tag{3.11a}$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_2} \cdot \frac{\partial f_2}{\partial W_2} \tag{3.11b}$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial W_1} \tag{3.11c}$$

Equations (3.11) outline the error gradient computation mechanism: backpropagation. Backpropagating the error gradient is updating weights layer-by-layer from the end to the beginning of the function chain: $\frac{\partial \mathcal{L}}{\partial W_3}$ is firstly calculated as in (3.11a). Then $\frac{\partial \mathcal{L}}{\partial W_2}$ is computed as in (3.11b) but the first term was already calculated in (3.11a). Then $\frac{\partial \mathcal{L}}{\partial W_1}$ is computed as in (3.11c) but the two first terms were already computed in (3.11b). From this observation can be derived an iterative algorithm for backpropagation (see Algorithm 1).

Algorithm 1: The propagation and backpropagation algorithm

Data: (X, Y) the training set and $\mathcal{L}$ the loss function
Data: $(f_i)_{1 \leq i \leq n}$ the functions composing the network
Data: $(W_i)_{1 \leq i \leq n}$ parameters to be optimized
Data: α learning rate
Result: Optimized $(W_i)_{1 \leq i \leq n}$
// Propagation
$o_0 = X$
for $i \in 1..n$ **do**
  $o_i \leftarrow f(o_{i-1}, W_i)$
end
// Backpropagation
$\mathrm{g} = \frac{\partial \mathcal{L}}{\partial f_n}(X, Y)$
for $i \in n..1$ **do**
  $\mathrm{g} \leftarrow \mathrm{g} \cdot \frac{\partial f_{i+1}}{\partial f_i}(o_i)$
  $W_i \leftarrow W_i - \alpha \cdot \mathrm{g} \cdot \frac{\partial f_i}{\partial W_i}(o_{i-1})$
end

3.2.3.3 Hyperparameters

The process of building a neural network architecture is a non-trivial process. It can be seen as a separate optimization process even though this one is often empirically-operated. Among hyperparameters in neural networks are topological parameters (number and types of layers, shortcuts, …) and layer parameters (number of neurons for perceptron layers, filter size/padding for convolutional ones, pooling window size for pooling layers, activation functions, …). There are also numerous hyperparameters when it comes about the training process: choice of gradient descent algorithm, size of sample to compute the gradient, loss function, regularization parameters to prevent overfitting, etc.

3.2.4 Towards differential programming

Differential programming is introduced as an abstraction framework in an effort to generalize neural networks beyond multi-layer perceptron and convolutional neural networks. It is a canonical way of conceptualizing neural networks without assuming on which domain it operates and how it operates.

A neural network is a set of layers (or functions) $\{f_i\}_{1 \leq i \leq n}$ for which output computability and derivability is ensured. These layers can have parameters or not and are connected in a computational graph $\mathcal{G}$. If an edge exists from layer f_k to f_l, f_k must be calculated to compute f_l in the context of propagation and, conversely in the context of backpropagation, f_l must be computed to compute f_k.

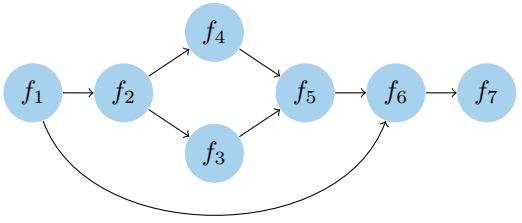


Figure 3.5: Example of computational graph for a neural network

Independently of the content of each layer, as long as both derivability and computability are ensured and the computational graph is known, it is possible to calculate the backpropagation stream at any point on the graph based on the chain rule of derivative. Figure 3.5 depicts an example.

3.3 Graphs and graph comparison

3.3.1 Graphs

A graph is an abstract model that can be used to represent data. It defines a given discrete data space through two concepts: points (with vertices) and adjacency (with edges).

Definition 1. *Graph*
A graph is a tuplet $G = (V, E)$
V *is a set of vertices*
$E \subseteq V \times V$ *is a set of edges*

The vertices set up locations that make up the graph space while the edges defines spatial (or neighbourhood) relations between these locations. The graph are said to be undirected when the defined relations are commutative (or symmetrical) and directed in the contrary.

Definition 2. *Directed graph*
A directed graph is a graph $G = (V, E)$
For which E *is a set of vertices ordered pairs*
(i.e. $\forall (i, j) \in E : (i, j) \neq (j, i)$*)*

Definition 3. *Undirected graph*
An undirected graph is a graph $G = (V, E)$
For which E *is a set of vertices un-ordered pairs (or vertices 2-sets)*
(i.e. $\forall (i, j) \in E : (i, j) = (j, i)$*)*

A graph defines the topological characteristics of the space it represents. It is analogous to the domain on which a signal is defined. Extending the analogy further, it is possible to define a signal on a given graph: the graphs becomes attributed (or labeled).

Definition 4. *Attributed Graph*
An attributed graph is a quadruplet $G = (V, E, \mu, \zeta)$ *with:*
V *is a set of vertices*
$E \subseteq V \times V$ *is a set of edges*
$\mu : V \rightarrow L_V$. μ *is a vertex labeling function which associates the label* l_{u_i} *to a vertex* u_i
$\zeta : E \rightarrow L_E$. ζ *is an edge labeling function which associates the label* $l_{e_{ij}}$ *to an edge* e_{ij}
where L_V *and* L_E *are the sets of labels respectively for vertices and edges. These sets can be of any type: numbers (integers, real, ...), symbolic, ...*

3.3.2 Graph comparison methods

Graph comparison methods can be categorized into two categories whether the comparison operates on graph or vector space.

3.3.2.1 Vector space

Explicit graph embedding ($\phi : \mathbb{G} \rightarrow \mathbb{R}^n$) A first one consists in transforming the initial structural problem in a common statistical pattern recognition one by describing the graphs with vectors in an Euclidean space [188]. In such a context, some features (vertex degree, labels occurrence histograms,etc.) are extracted from the graph. Hence, the graph is projected in a Euclidean space and classical machine learning algorithms can be applied. Such approaches suffer from a main drawback: to have a satisfactory description of topological structure and graph content, the number of such features has to be very large and dimensionality issues occur.

In this explicit embeddings family can be included graph neural networks: the embedding is learnt through gradient-based methods.

Implicit graph embedding ($k :< \mathbb{G}, \mathbb{G} >\rightarrow \mathbb{R}$) Another family of approaches also consists in using kernel-based machine learning algorithms. In the kernel approaches, an explicit data representation is of secondary interest. That is, rather than defining individual representations for each pattern or object, the data at hand is represented by pairwise comparisons only. The graphs are not explicitly but implicitly projected in a Euclidean space without defining the function ϕ. More formally, under given conditions, a similarity function can be replaced by a graph kernel function $k :< \mathbb{G}, \mathbb{G} >\rightarrow \mathbb{R}$ where $\mathbb{G}$ is the graph space. Most kernel methods can only process kernel values which are established by symmetric and positive definite kernel functions. Many kernels have been proposed in the literature [208, 114]. In most cases, the graph is embedded in a feature space composed of label sequences through a graph traversal. According to this traversal, the kernel value is then computed by measuring similarity between label sequences. Even if such approaches have proven to achieve high performance, they suffer from their lack of interpretability. In fact, it is very difficult to come back to graph space from the kernel space. This problem is also known as "pre-image".

Dissimilarity space embedding Another possible approach also consists in projecting the graphs in a Euclidean space of a given dimension but using a distance matrix between each pairs of graphs. In such cases, a dissimilarity measure between graphs has to be designed [38]. Kernels can be derived from the distance matrix. It is the case for multidimensional scaling methods proposed in [260].

3.3.2.2 Graph space ($d : \mathbb{G} \times \mathbb{G} \to \mathbb{R}$)

To compare unknown objects, one needs to define a metric that measures the distance between the unknown object and the elements in the learning database. The similarity or dissimilarity between two graphs requires the computation and the evaluation of the "best" matching between them. This is called graph matching. Graph matching is further detailed in the following sections.

3.3.3 Graph comparison through matching

Graph matching is a wide family of problems which are about dealing with graph similarity (or dissimilarity). In other words, it is about trying to identify whether two graphs are equivalent or not.

Formally-speaking, finding a matching between two graphs $G_1 = (V_1, E_1, \mu_1, \zeta_1)$ and $G_2 = (V_2, E_2, \mu_2, \zeta_2)$ is finding a function $f : V_1 \to V_2$ such that some more or less stringent similarity constraints are met.

Taxonomy is arranged regarding the problems matching tolerance: problems are said to be either exact or error-tolerant.

3.3.3.1 Exact graph matching

Exact problems need exact occurence of one graph's nodes and edges in the other graph.

The most straight-forward exact problem is graph isomorphism which is about determining whether two graphs are strictly equals or not.

Definition 5. *Graph isomorphism*
A bijective function $f : V_1 \to V_2$ is a graph isomorphism from G_1 to G_2 if:

1. $\forall i \in V_1,\ \mu_1(i) = \mu_2(f(i))$
2. $\forall (i,j) \in V_1 \times V_1, (i,j) \in E_1 \equiv (f(i), f(j)) \in E_2$
3. $\forall (i,j) \in E_1,\ \zeta_1((i,j)) = \zeta_2((f(i), f(j)))$

Subgraph isomorphism is about finding one graph as part of the other graph.

Definition 6. *Subgraph isomorphism*
An injective function $f : V_1 \to V_2$ is a subgraph isomorphism from G_1 to G_2 if:

1. $\forall i \in V_1,\ \mu_1(i) = \mu_2(f(i))$
2. $\forall (i,j) \in V_1 \times V_1, (i,j) \in E_1 \implies (f(i), f(j)) \in E_2$
3. $\forall (i,j) \in E_1,\ \zeta_1((i,j)) = \zeta_2((f(i), f(j)))$

A more restrictive variant can be found as *induced subgraph isomorphism* which takes care of edge preservation not only from G_1 to G_2 but also from G_2 to G_1 ($\forall (i,j) \in V_1 \times V_1, (i,j) \in E_1 \equiv (f(i), f(j)) \in E_2$).

The last exact graph matching problem is maximum common subgraph which is a generalization of subgraph isomorphism as it tries to identify a third graph to occur (as a subgraph) in the two graphs that are being matched.

Definition 7. *Maximum common subgraph*
Let G_1 and G_2 be two graphs. A graph G_s is said to be a common subgraph of G_1 and G_2 if there exists subgraph isomorphism from G_s to G_1 and from G_s to G_2. The largest common subgraph is called the maximum common subgraph (MCS), of G_1 and G_2.

3.3.3.2 Error-tolerant graph matching

Graph isomorphism rarely occurs in pattern analysis applications. Inexact problems allow matching to be approximated in terms of topology (as in subgraph-based exact problems) but also introducing substitution tolerance: not only it is possible for a node not to be matched, it can also be matched to a node with non-equal attributes.

In order to define the best matching solution possible in this context, inexact problems revolve around the notion of minimizing matching cost.

Graph edit distance [241] is a measure of how far are two graphs that are being matched. This measure of dissimilarity is based on edit operations that must be performed to transform one graph into another. These operations are given costs and the graph edit distance between two graphs is the minimal total cost of every edit operations needed.

Definition 8. *Edit Path*
A set $\{ed_1, \cdots, ed_k\}$ *of* k *edit operations* ed_i *that transform* G_1 *completely into* G_2 *is called a (complete) edit path* $\lambda \in \gamma(G_1, G_2)$ *between* G_1 *and* G_2.
$\gamma(G_1, G_2)$ *is the set of every possible edit paths.*

Definition 9. *Graph Edit Distance*
The graph edit distance between G_1 *and* G_2 *is defined as:*

$$d_{\lambda_{min}}(G_1, G_2) = \min_{\lambda \in \gamma(G_1, G_2)} \sum_{ed_i \in \lambda} c(ed_i) \tag{3.12}$$

Image signals are real valued and so are image graph attributes, therefore the probability for two images to have exactly the same labels is low. Therefore, graphs in a single class will be different in that they are impacted with pose and individual-related variations. That is why the problem that is the base of this thesis work on graphs must be error tolerant.

3.3.4 Solving graph edit distance

In [241, 31], the GED problem is shown to be equivalent to a Quadratic Assignment Problem (QAP).

We define the error-correcting graph matching problem to compute the GED of two graphs. Let $G_1 = (V_1, E_1, \mu_1, \zeta_1)$ and $G_2 = (V_2, E_2, \mu_2, \zeta_2)$ be two graphs, with $V_1 = \{1, \cdots, n\}$ and $V_2 = \{1, \cdots, m\}$. To apply removal or insertion operations on nodes, node sets are augmented by dummy elements. The removal of each node $i \in V_1$ is modeled as a mapping $i \to \epsilon_i^2$ where ϵ_i^2 is the dummy element that is associated with i. As a consequence, the set V_2 is increased by n dummy elements ϵ_2 to form a new set $N_2 = N_2 \cup \epsilon_2$. The node set V_1 is increased similarly by m dummy elements ϵ_1 to form $N_1 = V_1 \cup \epsilon_1$.

Note that N_1 and N_2 have the same cardinality : $n_1 = n_2 = n + m$. Each element of the two graphs can be edited only once. A graph matching solution is defined as a subset of possible correspondences $y \subset N_1 \times N_2$, which are represented by a binary assignment matrix $Y \in \{0,1\}^{n_1 \times n_2}$, where n_1 and n_2 denote the sizes of V_1 and V_2, respectively. If $i \in V_1$ matches with $a \in V_2$, then $Y_{i,a} = 1$, or $Y_{i,a} = 0$ otherwise. We denote by $y \in \{0,1\}^{n_1 n_2}$, a column-wise vectorized replica of Y. With this notation, the error-correcting graph matching problem can be expressed as the problem of finding the assignment vector y^* that minimizes a score function $d(G_1, G_2, y)$ as follows:

Definition 10. *Error-correcting graph matching formulation*

$$y^* = \underset{y}{\operatorname{argmin}} \quad d(G_1, G_2, y) \tag{3.13a}$$

$$\textit{subject to} \quad y \in \{0,1\}^{n_1 n_2} \tag{3.13b}$$

$$\sum_{i=1}^{n_1} y_{i,a} = 1 \quad \forall a \in [1, \cdots, n_2] \tag{3.13c}$$

$$\sum_{a=1}^{n_2} y_{i,a} = 1 \quad \forall i \in [1, \cdots, n_1] \tag{3.13d}$$

where constraints 3.13c and 3.13d indicate that each node of a graph must be matched with only one node of the other graph.
The function $d(G_1, G_2, y)$ measures the dissimilarity of graph attributes, and is typically decomposed into a first order dissimilarity function $d_V(\mu_1(i), \mu_2(a))$ for a node pair $v_{1,i} \in V_1$ and $v_{2,a} \in V_2$, and a second-order similarity function $d_E(\zeta_1(ij), \mu_2(ab))$ for an edge pair $e_{1,ij} \in V_1 \times V_1$ and $e_{2,ab} \in V_2 \times V_2$. Dissimilarity functions are usually represented by a symmetric dissimilarity matrix D, in which a non-diagonal element $D_{ia;jb} = d_E(\zeta_1(ij), \zeta_2(ab))$ contains the edge dissimilarity of two correspondences (ia, jb) and a diagonal term $D_{ia;ia} = d_V(\mu_1(i), \mu_2(a))$ represents the node dissimilarity of a correspondence (i, a). Thus, the objective function of graph matching is defined as:

$$d(G_1, G_2, y) = \sum_{y_{ia}=1} d_V(\mu_1(i), \mu_2(a)) + \sum_{y_{ia}=1} \sum_{y_{jb}=1} d_E(\zeta_1(ij), \zeta_2(ab)) = y^T D y \tag{3.14}$$

3.3.5 Solving error-tolerant graph matching problems

Error-correcting graph matching is an integer quadratic programming problem. More precisely, it is equivalent to solving a non-convex quadratic assignment problem, which is known to be NP-hard. Obtaining optimal solutions is therefore potentially impracticable. Methods can be divided into two categories: exact methods and heuristic methods. An exact method computes an optimal solution of a given problem. On the opposite, heuristic methods compute sub-optimal solutions.

Exact methods Nevertheless, some exact methods have been applied to error-tolerant graph matching which give exponential complexity solvers. Some approaches use tree-search based algorithms [294, 276] while others translate graph matching to Integer Linear Problem [147, 174, 66].

Heuristic methods Heuristics methods can be grouped into two families: problem reformulation and heuristic optimization. In the problem reformulation paradigm, the GM problem is reduced into a simpler problem. However, the optimal solution of the simpler problem is not the optimal solution of the original problem. In the heuristic optimization category, the GM problem is solved by an heuristic algorithms that only explore sub-parts of the solution space and thus leads to find near-optimal solutions. Each heuristic method can also be divided between deterministic and non-deterministic. The characteristic of deterministic strategies is that under the same conditions the same solution is always obtained.

Nearly all possible solver paradigms were put to the test for QAP. These include, but are not limited to, convex relaxations based on Lagrangean decompositions [292], linear [272], convex quadratic [7] and semi-definite [270] relaxations, which can be used either directly to obtain approximate solutions or just to provide lower bounds. To tighten these bounds several cutting plane methods were proposed [18]. On the other side, various primal heuristics, both (i) deterministic, such as local search [220], graduated assignment methods [120], fixed point iterations [171], spectral technique and its derivatives [62, 170] and (ii) non-deterministic (stochastic), like random walk [52] and Monte-Carlo sampling [169] were suggested to provide approximate solutions to the problem.

As is usual for NP-hard problems, no single method can efficiently address all QAP instances. Different applications require different methods and we concentrate here on problem instances specific for computer vision. Traditionally within this community predominantly primal heuristics are used, since demand for low computational time usually dominates the need to obtain optimality guarantees. However, a proposed solver [242, 241] based on Linear Sum Assignment has shown a good trade-off between speed and accuracy and surpassed numerous state-of-the-art primal heuristics. This family of methods is also known as Bipartite Matching (BP). To approximate the exact problem, bipartite methods formulate a linear sum assignment problem where only the nodes are to be matched. In order to take edge costs into account, additional estimation costs are added to each node costs. Classical LSAP solving algorithm such as the Hungarian algorithm is then used to get the node matchings.

3.4 About the manuscript

3.4.1 Description

The following chapters describe every contribution made during the PhD. Chapter 4 is about presenting a way to train convolutional neural networks on insect image recognition. CNNs are state of the art when it comes to most computer vision tasks including image classification. As stated in Chapter 2, CNNs are to our knowledge yet to be tested on insect image recognition. This is therefore a logical place to start working on this problem.

As an effort to explore peculiarities of image insect recognition, we question the relevance of a structural representation of insect images through graphs. To stay consistent with state of the art, Chapter 5 and Chapter 6 present approaches that apply neural paradigm to the graph domain. Chapter 5 presents a perceptron algorithm applied on graph and Chapter 6 defines a graph convolutional neural network approach.

3.4.2 Common methodology

Approaches presented in Chapter 5 and Chapter 6 are debut work. This means they have yet to be experimented on insect images but also generic images. Along these chapters, standard benchmark datasets are used to get preliminary results to give comparison with other similar approaches.

Chapter 4

Convolutional neural networks and transfer learning

Contents

4.1 Introduction . . . 55
4.2 Problem definition . . . 56
4.2.1 Image-based arthropod classification . . . 56
4.2.2 Transfer learning . . . 56
4.3 State of the art . . . 57
4.4 Proposed approach . . . 57
4.4.1 Transfer learning adapted VGG-16 architecture . . . 58
4.4.2 Data preprocessing . . . 58
4.4.3 Learning rule . . . 59
4.4.4 Hyper-parameter optimization . . . 60
4.5 Experimental work . . . 60
4.5.1 Data . . . 60
4.6 Follow up . . . 60

4.1 Introduction

Considering the hierarchical feature learning fashion in DNN, first layers are expected [327] to learn features for general simple visual building blocks, such as edges, corners and simple blob-like structures, while the last layers learn more complicated abstract task-dependent features. In general, the ability to learn domain-dependent high-level representations is an advantage enabling DNNs to achieve great recognition capabilities. However, DNN require large labeled dataset to efficiently learn their huge number of parameters, and insect image databases only contains a few hundred of labeled samples. To overcome this lack of data, transfer learning has been proposed which aims at transferring the knowledge from a more or less related task that has already been learned on a large dataset. Although transfer learning has been shown to be very efficient on many applications, its limits and its practical implementation issues has not been much studied. For example, it would be practically important to determine how much data on the target domain is required for domain adaptation with sufficient accuracy for a given task, or how many layers from a model fitted on the source domain can be effectively transferred to the target domain. Or more

interestingly, given a number of available samples on the target domain, what layer types and how many of those can we afford to learn. Moreover, there is a common scenario in which a large set of annotated data is available (ImageNet-arthropod subset), often collected in a time-consuming and costly process. To what extent these data can contribute to a better analysis of new datasets is another question worth investigating.

In this chapter, we aim towards answering the questions discussed above. To tackle the insect image recognition problem, we use transfer learning methodology for domain adaptation of models trained on scene images. The contribution of this paper is an effective learning methodology for the insect recognition problem. The role of pretraining in the DNN accuracy is detailed and the question is rised about how many insect samples are required to achieve a sufficient accuracy from the biologist viewpoint.

4.2 Problem definition

This section presents the problems tackled in this chapter.

4.2.1 Image-based arthropod classification

Image-based arthropod classification could be seen as an application of image classification. Based on some photograph depicting the specimen, its biological identity is to be determined. Even though deep convolutional neural networks were successfully applied to image classification, their efficiency might be limited for two reasons:

1. **Rarity of images**. Only experts such as taxonomists and trained technicians can identify accurately insect classes, because it requires special skills acquired through extensive experience. In lab-based setting, most of the acquisition systems are manually manipulated which increase the workforce amount (see examples on Figure 2.1(a) and Figure 2.1(b)).

2. **Large discrepancies among class cardinalities**. Insect capture campaigns are season-dependent impacting the number and the type of the captured insects. This fact can be translated in the pattern recognition domain as an imbalanced classification problem.

4.2.2 Transfer learning

Although many studies report excellent results on specific domains and image acquisition protocols, the generalization of these models on test data with different distributions are often not investigated and evaluated. Therefore, to ensure the usability of the trained models in real world practice, the domain adaptation remains a valuable field of study.

Mathematically, a domain $\mathcal{D}$ can be expressed by a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \cdots, x_n\} \in \mathcal{X}$ [218]. A supervised learning task on a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, consists of a pair of a label space Y and an objective predictive function $f(.)$ (denoted by $T = \{Y, f(.)\}$). The objective function $f(.)$ can be learned from the training data, which consists of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$. After the training process, the learned model denoted by $\widetilde{f}(.)$ is used to predict the label for a new instance x. Given a source domain $\mathcal{D}_\int$ with a learning task T_S and a target domain $\mathcal{D}_\mathcal{T}$ with learning task T_T , transfer learning is defined as the process of improving the learning of the target predictive function $f_T(.)$ in $\mathcal{D}_\mathcal{T}$ using the information in $\mathcal{D}_\mathcal{S}$ and T_S, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$, or $T_S \neq T_T$. We denote $\widetilde{f}_{ST}(.)$ as the predictive model initially trained on the source domain $\mathcal{D}_\mathcal{S}$, and domain-adapted to the target domain $\mathcal{D}_\mathcal{T}$.

4.3 State of the art

Image-based insect recognition is not a newly addressed problem. A detailed study was lead in Chapter 2. This study focused successively on capture protocols, feature extraction methods and finally classification. The last two points are of first interest here while the first point is to be considered as an input constraint and depends fully on the biologists workflow. Moreover, the datasets and the classes that constitute them are very different and motivated by different biological scopes and problems.

Regarding feature extraction, the first pieces of work tackle the problem in very ad-hoc ways. [76] extract dimensions on the insects wings using venations as keypoints. Others [304] use geometrical attributes from the region of interest as features. Then, studies began to emerge using standard local and global features such as SIFT [314]. On top of these standard handcrafted features are learnt higher level features using MLPs [266], Bag of Words [159] or Sparse Coding [321]. The next step is to introduce hierarchically learnt descriptors, with many levels of abstraction. [314] uses a stack of Denoising Autoencoders to this extent. The observation that can be done is the features are no more particular to the problem itself and is about learning hierarchical representations to get satisfying feature spaces that suit the biologists goals through learning on their images. In such a frame, [266] applies MLPs on raw pixel even though the problem is a very simple one (two classes : harmful/harmless insects).

Deep neural networks are machine learning models that are now the state-of-the-art models for almost every pattern recognition tasks. Their main strength is their ability to learn suitable features for a given task, thus avoiding the need for handcrafted features design. Convolutional Neural Networks (CNN) are an instantiation of DNN dedicated to image processing. Through a mechanism of shared weights, their input layers are convolutional filters that automatically learn features from multidimensional signals [289, 87, 238, 156]. CNN now outperform most of traditional approaches (e.g. based on handcrafted features) in various image analysis and recognition domains, such as natural scene recognition [156, 284], Medical imaging [259, 19], Image segmentation [157] or handwriting recognition [225]. However, although very efficient, deep CNN models require a huge amount of labeled data to avoid overfitting, which can be compared to "learning by heart" the training database. In order to circumvent these issues, the architectures are generally trained using many tricks such as regularization, dropout or data augmentation for improving their generalization ability. But all these hints do not replace a reasonable amount of labeled data for training the architectures.

Recently, the transfer learning idea has been proposed to train huge models with small labeled datasets. It is based on the exploitation of pre-trained models on a huge datasets from another domain. The model is then fine tuned on a specific, smaller database of the domain considered. Models are often trained on imagenet, a natural scene database of more than 14M images [156]. Even if the specific domain strongly differs from the imagenet database, the transfer learning have shown very impressive results on many tasks such as handwriting recognition [56], signature identification [125], and medical imaging [17].

4.4 Proposed approach

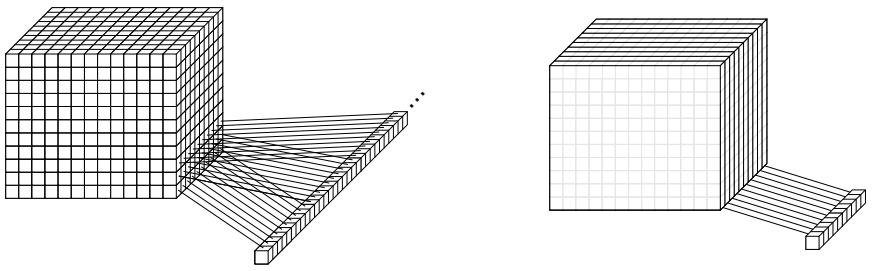
This article presents a method to easily train a deep convolutional neural network architecture using transfer learning, with application to the insect image recognition problem. The deep architecture has been adapted to be easily trained on low volume datasets and the feeding, learning and hyper-parameter optimization procedures are detailed in the next subsections.

4.4.1 Transfer learning adapted VGG-16 architecture

The fine-tuning CNN experiments involve VGG-16 [284] instances pretrained on ImageNet-1000 [53]. Although more recent models exist (for example GoogLeNet or ResNet.), preliminary experiments shown these architectures yielded similar results on the problem tackled here. Moreover, VGG-16 was chosen for its simplicity and relatively small number of layers.

VGG-16 has undergone a crucial modification in its convolutional end. The original model end (see Figure 4.2 on the left) consists of a three-layered MLP (FC1-3) which takes as inputs every coefficients from the last 2D activation map (block5-pool): The $7 \times 7 \times 512$ volume is flattened to obtain 25088 input values for the MLP. This has been replaced by a global average pooling filter which keeps only the averages of each of the 512 7×7-slices of the $7 \times 7 \times 512$ volume [176]. This reduces the input size of the MLP from 25088 to 512. A visualization of such a transformation is pictured in Figure 4.1. This modification acts as a regularization that limits the overfit as it reduces the number of units in the first part of the MLP, and takes more advantage to the structure of the last convolutional feature maps [176]. The remains of the MLP consist of a single hidden layer of size 256 and the prediction layer.

The overall transfer learning architecture is described in Figure 4.2.



(a) A flatten layer: every coefficients of the 3-D tensor become the output coefficients

(b) A global pooling layer: the average values of every slices of the 3-D tensor become the output coefficients

Figure 4.1: Layer translating volumes into 1-D vector to take as input of the fully-connected end of a CNN

4.4.2 Data preprocessing

Preparing the data to feed the neural network is critical to the learning process. It consists of two steps. The first one is about adapting the input image (size $n \times m \times 3$, n an m being respectively the height and width of the input image) to the receptive field of the neural network (size $224 \times 224 \times 3$, the size of the ImageNet input on the three color channels). The most straight-forward approach to this size adaption is to resize the image. The pros of this method lies in the fact that all the information of the image is fed to the network. The main cons is that the ratio of the image is not preserved. In the case of one dataset in use in this thesis, the images ratios are significantly various and thus resizing the images would end up in bad performance. The method used here consists in cropping the images in their center so that the image is of size $k \times k \times 3$ with $k = \min(m, n)$. Cropping instead of resizing keeps the image ratio as is. Finally, a resize operation can be performed from size $k \times k \times 3$ to size $224 \times 224 \times 3$.

Once the size normalization is done, the images must be standardized across their dataset featurewise: this is done by considering a given pixel as a random variable following a normal distribution ($X \to \mathcal{N}(M, \Sigma)$). The standardized variable is $X' = \frac{X-M}{\Sigma}$. This is done separately

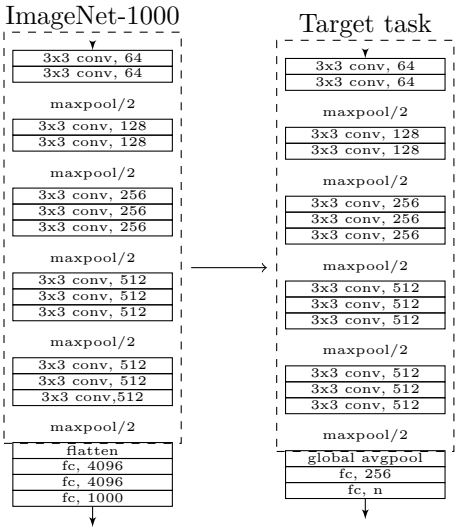


Figure 4.2: Transfer learning architecture with modified VGG-16

for each color channel. Using the image unstandardized results in inability to train the system (the loss stagnating at its initial level). This behavior can be explained in the following manner: if the image is not standardized, the differences in scales causes the gradient descend to be applied at these different scales thus differently depending on the location or pixel value.

Finally, data augmentation was used. The training images, each time they were fed to the network, were modified using a set of randomly chosen transformations (sheer, flip and zoom). It virtually augments the size of the set as many times as the number of epochs.

4.4.3 Learning rule

The data that is being dealt with is not only in small amount but also unequally distributed among classes. Neural networks are known to be very sensitive to cardinality discrepancies. There are two main solutions to address this problem. The first one consists of sampling on low-cardinality classes and the latter is to apply the gradient descent more or less depending on the cardinality of the examples classes. This latter method is the one being used here for the sake of simplicity. For each class $c \in Y$, the following weights have been applied:

$$w_c = \frac{\max_{c' \in Y} \text{Card}(c')}{\text{Card}(c)}$$

where Card(c) is the number of observations that fall into class c. w_c are used during the back-propagation step as weights to the error gradient descent term. The higher w_c, the stronger the error gradient is descended. The gradient descent method is vanilla Stochastic Gradient Descent with 0.9 Nesterov momentum [21]. On the contrary, the learning rate is divided by two if the loss hasn't been reduced in the span of 10 epochs. This intensification process enables the learning rate to be adapted depending on the scale of the gradient and therefore to avoid oscillations onto the search space. This behaviour is likely to lead to a local minimum. However in the case of multi-layer networks it is not a major issue. Choromanska et al. [55] not only states this but also that global minima are prone to lead to overfitting in practice.

4.4.4 Hyper-parameter optimization

The main limitation to the use of deep models is the hyper optimization process it requires in order to get satisfying results. Additionally, using transfer learning during the training phase adds a new hyper-parameter: the number of layers to be trained on the target task. Deep CNN are composed of low level features that are independent from the problem. These low level features are located in the first layers. An efficient learning methodology should allow not spending time optimizing these layers. In our proposal, we show that only the last convolutional block (block5) and obviously the last dense layer which has not been pretrained (fc1) are of first interest to easily train the model while achieving a good classification rate. Conceptually speaking, block5 represents the high level features that are insect-dependent, while fc1 is a decision phase to fit at best the classification task in term of number of classes and class distribution. Reducing the search space as much as possible is therefore critical when learning with small amount of data.

4.5 Experimental work

This section details the lead experiments. Experiments and their respective results are described as well as their interpretations.

4.5.1 Data

To test CNNs on insect image recognition, two datasets were generated: a lab-based one and a field-based one. These datasets are presented in Section 7.2.

4.6 Follow up

The next chapter will work around a first graph based approach to be put in competition with the CNN-based one presented in the current chapter.

Chapter 5

Graph perceptron

Contents

5.1 Introduction . . . 61
5.2 Problem statement . . . 62
5.2.1 Parameterized graph matching . . . 62
5.2.2 Graph classification problem . . . 63
5.2.3 Fixed parametrized graph matching for machine learning . . . 64
5.3 State of the art . . . 64
5.4 Proposal: a multiclass graph-based perceptron . . . 66
5.4.1 The perceptron choice . . . 66
5.4.2 2-class graph-based perceptron . . . 67
5.4.3 Multiclass graph-based perceptron . . . 68
5.4.4 Complexity analysis . . . 69
5.4.5 Discussion . . . 69
5.5 Experiments . . . 70
5.5.1 Experimental setting . . . 70
5.5.2 Results . . . 72
5.6 Conclusions and Perspectives . . . 73
5.7 Follow up . . . 74

5.1 Introduction

This chapter presents a perceptron algorithm that allows to learn to solve graph classification problems.

Our graph perceptron approach is characterized in two ways: the complexity and the *a priori* knowledge integrated in the model. First, on the complexity side, conventional kernel-based and k-NN methods cannot be applied directly on large data sets as they are of quadratic complexity. The number of calls to the dissimilarity measure grows quadratically as a function of the number of graphs in the learning and test datasets. Second, graph kernel techniques rely on the choice of *a priori* structures defining a graph similarity (i.e.: walks, paths, cliques, etc.). Also, recent studies have revealed that hand-crafted edit costs or matching functions, which are typically used in graph matching, are insufficient for capturing the inherent structure that underlies the problem at hand. As a consequence, a better optimization of the graph matching problem does not guarantee

better correspondence accuracy with respect to a user-defined ground-truth [44, 51] or a better classification rate. To tackle this issue, we introduce a set of parameters into the graph matching problem. Then, we use a learning scheme to obtain the best values for these parameters, according to the problem at hand. Such a learned matching function better model the inherent structure of the classification problem. This chapter deals with paradigms that operate directly on the graph space and can thus capture more structural distortions than embedding techniques, as well as giving more interpretable solutions (because expressed directly in the graph space).

In Section 5.2, the definitions are established and the parametrized graph matching problem is presented. In Section 5.3, the state of the art is detailed and weaknesses are identified. In Section 5.4, the methodology, complexity and algorithms of our proposed approach are presented. Section 5.5, highlights an experimental protocol along with its results. Finally, in Section 5.6, conclusions and perspectives are discussed.

5.2 Problem statement

In this section, the problem of learning discriminative graph matching is formally defined.

5.2.1 Parameterized graph matching

In the context of the objective function that is defined in Eq. 3.14, an interesting question arises: What can be learned to improve graph matching. To address this issue, we parameterize Eq. 3.14 as follows. Let $\pi(a) = i$ denote an assignment of node $v_{2,a}$ in G_2 to node $v_{1,i}$ in G_1, i.e. $y_{ia} = 1$. A joint feature map $\Phi(G_1; G_2; y)$ is defined by aligning the relevant dissimilarity values of Eq. 3.14 into a vector form as: $\Phi(G_1, G_2, y) = [\cdots, d_V(\mu_1(\pi(a)), \mu_2(a)), \cdots, d_E(\zeta_1(\pi(a)\pi(b)), \zeta_2(ab)), \cdots]$. By introducing a real-valued vector β to weight all elements of this feature map, we obtain a parametrized objective function:

$$d(G_1, G_2, y, \beta) = \beta\Phi^T(G_1, G_2, y) \tag{5.1a}$$

$$= \cdots + \beta_a \cdot d_V(L_1(\pi(a)), \mu_2(a)) + \\ + \beta_{ab} \cdot d_E(\zeta_1(\pi(a)\pi(b)), \zeta_2(ab)) + \cdots \tag{5.1b}$$

where β is a weight vector that encodes the importance of node and edge dissimilarity values. In the case of uniform weights, i.e. $\beta = \mathbf{1}$, all elements of vector β are 1, and Eq. 5.1a is reduced to the conventional graph matching score function of Eq. 3.14: $d(G_1, G_2, y) = d(G_1, G_2, y, \mathbf{1})$. An example of the parametrized objective function is given in Figure 5.1. The discriminative weight formulation is general in the sense that it can assign different parameters to individual nodes and edges. However, it does not learn a graph model that underlies the feature map, and requires a reference graph G^2 at query time, whose attributes cannot be modified in the learning phase. The new discrete optimization problem can be rewritten from Problem 10 as follows:

Problem 1. *Parametrized graph matching problem*

$$y^* = \underset{y}{\operatorname{argmin}} \quad d(G_1, G_2, y, \beta) \tag{5.2a}$$

$$\textit{subject to} \quad y \in \{0,1\}^{n_1 n_2} \tag{5.2b}$$

$$\sum_{i=1}^{n_1} y_{i,a} = 1 \quad \forall a \in [1, \cdots, n_2] \tag{5.2c}$$

$$\sum_{a=1}^{n_2} y_{i,a} = 1 \quad \forall i \in [1, \cdots, n_1] \tag{5.2d}$$

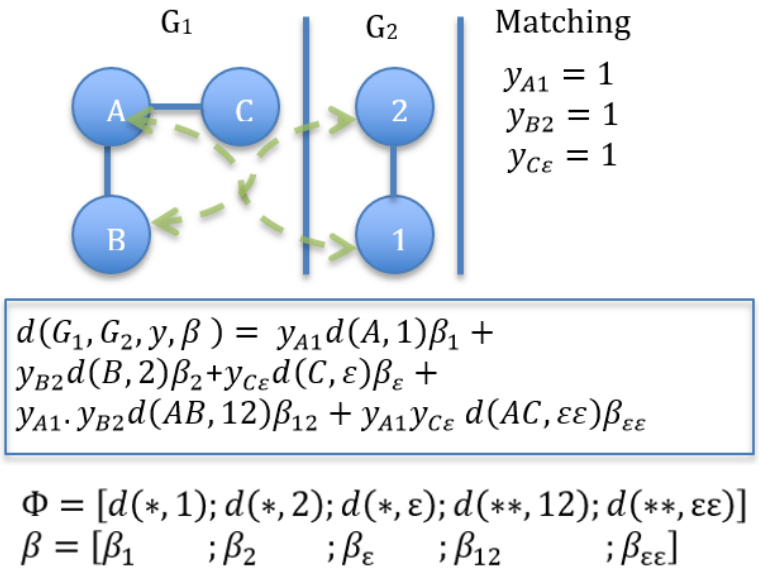


Figure 5.1: Illustration of the parametrized score function computation.

5.2.2 Graph classification problem

For clarity, the rest of the definition is focused on a 2-class problem. However the paradigm can be extended to a multi-class problem. A linear classifier is a function that maps its input $x \in \mathbb{R}^q$ (a real-valued vector) to an output value $f(x) \in \{0, 1\}$ (a single binary value):

$$f(x) = \begin{cases} 1 & \text{if } \beta \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where β is a vector of real-valued weights, and $\beta \cdot x$ is the dot product $\sum_{i=1}^{q} \beta_i x_i$, in which q is the number of inputs to the classifier and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value. The value of $f(x)$ (0 or 1) is used to classify x as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, the weighted combination of inputs must produce a positive value that is greater than $|b|$ in order to push the classifier over the 0 threshold. For clarity, b is often included in the weight vector β with $b = \beta_0 \times 1$. To extend this paradigm to graphs, let $\mathcal{D}$ be the set of graphs. Given a graph training set $TrS = \{(G_i, c_i)\}_{i=1}^{M}$, where $G_i \in \mathcal{D}$ is a graph and $c_i \in \mathcal{C}$ is the class of the graph, out of the two possible classes. The learning of a graph classifier consists of inducing from TrS a mapping function $f : \mathbb{G} \to \mathcal{C}$ that assigns a class to an unknown graph:

$$f(G) = \begin{cases} 1 & \text{if } \beta \cdot \Phi^T(G, G_m, y) + b > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{with } G_m \text{ a graph model}$$

Let $\Delta(TrS, f)$ be a function that computes the error rate obtained by a classifier f. We represent the error for the p^{th} training sample by $error_p = 1 - \delta(c_p, f(G_p))$, where c_p is the target value, $f(G^p)$ is the value that is produced by the classifier and $\delta(\cdot, \cdot)$ is the Kronecker Delta function. The error rate (Δ) is the mean of errors $error_p$ over the set TrS between the ground-truth values

and values produced by the classifier. Straightforwardly, we define $\eta = 1 - \Delta$ as the classification rate on the training set.

To address the problem of learning graph matching, we start with the discriminative weight formulation of Eq. 5.1a. We learn the weights β from labelled examples from TrS minimizing the function Δ. The objective function is the error rate function with extra weights β: $\min_{\beta} \Delta(TrS, f, \beta)$.

5.2.3 Fixed parametrized graph matching for machine learning

To perform conventional machine learning techniques such as Support Vector Machines (SVM) or Deep Neural Networks (DNN) on a data set, a fixed feature vector size is often mandatory. To obtain a fixed size, in Eq.5.1b, the sizes of the vectors β and $\Phi(G, G_m, y)$ must only depend on the size of G_m and not on the query graph $G = (V, E, L)$.

The graph elements of G and the components of graph $G_m(N_m, E_m)$ are aligned into a vectorial form by the function Φ. $\Phi(G, G_m, y)$ is a vector $\in \mathbb{R}^{|V_m|+|E_m|+2}$. Two extra components are added to accumulate node and edge deletion costs. For instance, it can occur that a node or an edge of G is mapped to a dummy element ϵ (node deletion or edge deletion). In such a case, the node deletions are mapped in the penultimate component of the vector $\Phi(\cdot)$. The edge deletions are mapped in the last component of the vector $\Phi(\cdot)$.

β is a vector $\in \mathbb{R}^{|V_m|+|E_m|+2}$. Individual weights are associated with each component of G_m. Two extra components are added to parametrize the node and edge deletion costs, respectively. A shared weight is associated with all node deletion costs. The same strategy is applied for edge deletions.

The value of the objective function remains the same. Substitutions and insertions have individual weights. Deletions share a mutual parameter. In Figure 5.1, G_1 holds 5 graph components and in Figure 5.2, G_1 holds 7 graph components. However, in both figures $\Phi(\cdot)$ and β are vectors of size 5 that only depend on model graph G_2.

5.3 State of the art

The literature on learning similarity/dissimilarity matching functions can be roughly categorized into two parts according to whether the objective is to minimize an error rate on the number of correctly matched graph components (matching level) or an error rate on a classification task (classification level).

Matching level. In this category, the purpose is to minimize the average Hamming distance between a ground-truth's correspondence and the automatically deduced correspondence. Caetano et al. [44] use a 60-dimensional node similarity function for appearance similarity and a simple binary edge similarity for edges. Leordeanu et al. [172] do not use d_V, and instead employ a multi-dimensional function d_E for dissimilarity of appearance, angle, and distance. The work of Torresani et al. [293] can be viewed as adopting 2-dimensional d_V and d_E functions for measuring appearance dissimilarity, geometric compatibility, and occlusion likelihood. In [60] a method for learning the real numbers for the insertion $d_V(\epsilon \to v)$ and deletion $d_V(v \to \epsilon)$ costs on nodes and edges is proposed. An extension to substitution costs is presented in [59]. While the optimization methods for learning these functions are different, all of them are essentially aimed at learning common weights for all the edges' and nodes' dissimilarity functions in a matching context. The discriminative weight formulation in Eq. 5.1a is more general in the sense that it can assign different parameters for individual nodes and edges. In [51], the discriminative weight formulation is also employed. The learning problem is turned into a regression problem and a structured support vector machine (SSVM) is used to minimize it. In [249], the main contribution is the prediction of

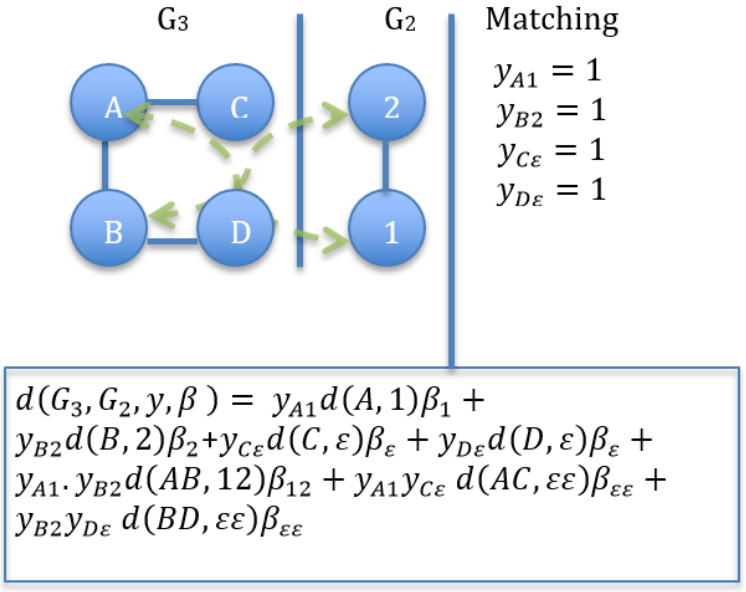


Figure 5.2: Parametrized matching function where G_1 has 7 graph components (4 nodes and 3 edges).

whether two nodes match or not. The node assignment is represented by a vector of 24 features. These numerical features are extracted from the node-to-node cost matrix **C** which was used for the original matching process. Then, using the assignments derived from exact graph edit distance computation as the ground truth, each computed node assignment is labeled correct or incorrect. This set of labeled assignments is used to train an SVM endowed with a Gaussian kernel to classify the assignments computed by the approximation as correct or incorrect. A major drawback is the optimal solution requirements for performing the learning.

Classification level. Learning graph matching in a classification context is more challenging since the ground truth is given at the class level and not at the node/edge level. In [242], a grid search on a validation set is used to determine the values of the parameters $K_n = \beta^n_{del} = \beta^n_{ins}$, which corresponds to the cost of a node deletion or insertion, and $K_e = \beta^e_{del} = \beta^e_{ins}$, which corresponds to the cost of an edge deletion or insertion. The main drawbacks of the grid search is the empirical definitions of intervals and bounds of the grid, which require expertise on the problem. The speed can also drastically decrease as the grid becomes larger. The method aimed at learning common weights for all the edges and nodes and only distinguishes weights by type of operations: deletion, insertion or substitution ($\beta_{del}, \beta_{ins}, \beta_{sub}$). Neuhaus et al. [209] address the issue of learning dissimilarity functions for numerically labeled graphs from a corpus of sample graphs. A system of self-organizing maps (SOMs) that represent the dissimilarity spaces of node and edge labels was proposed. The learning process adapts the edit costs in such a way that the similarity of graphs from the same class is increased. The matching are computed only once before learning the costs and each edit operation is considered independently from the matching it belongs. From the same authors, in [207], the graph matching process is formulated in a stochastic context. A maximum likelihood parameter estimation of the distribution of matching operations is performed. The underlying distortion model is a mixture of multivariate Gaussians. The model is learned using an Expectation Maximization algorithm. The matching costs are adapted to decrease the distances between graphs from the same class, thereby leading to compact graph clusters. The method requires the summation over all possible edit paths between two graphs which is not

tractable in practice. Two limitations can be identified: (i) attributes must be numeric vectors of a fixed dimension. The methods focus on organizing the feature space rather than learning graph matching and (ii) the methods are not discriminative, they model graphs that belong to the same class. A clear picture of the state of the art is provided in Table 5.1. Adapting methods that operate at the matching level is not trivial since node correspondences must be inferred from the class labels. The neural methods that are proposed in [209] perform well at the classification level but are limited to vector attributes and each edit operation is considered independently from the matching it belongs. The former limitation is leveraged in [207] thanks to a probabilistic framework. However, the method is not discriminative and models graphs that belong to the same class. In this paper, the contribution is to match graphs and to learn edge/node dissimilarity functions at the same time as a single minimization problem. A neural-based algorithm and the discriminative weight formulation aim at learning graph matching dissimilarity functions in a discriminative way.

This idea was published in Raveaux et al. [232] with 2-class application and Martineau et al. [194] with multiclass application. Martineau et al. [194] also introduced set median graphs as model graphs and broader experiments.

Table 5.1: Papers published about learning graph matching

Ref	Ground-Truth Level	Shared parameters	Graph type	Optimization method	Learning space	Discriminative or generative	Matching and Learning
[209]	Class	No	Numeric	SOM	Vector	Generative	Independently
[207]	Class	No	Numeric	EM	Vector	Generative	Independently
[242]	Class	Yes	Any	Grid Search	Distance	Discriminative	Simultaneously
[249]	Matching	No	Any	SVM	Distance	NA	Independently
[60, 59]	Matching	Yes	Any	Quadratic programming	Distance	NA	Independently
[44]	Matching	No	Numeric	Bundle	Vector	NA	Simultaneously
[293]	Matching	No	Any	Bundle	Distance	NA	Simultaneously
[51]	Matching	No	Any	SSVM	Distance	NA	Simultaneously
This paper	Class	No	Any	Neural network	Distance	Discriminative	Simultaneously

5.4 Proposal: a multiclass graph-based perceptron

In this section, the choice of the perceptron algorithm is explained. Then, on a 2-class problem, the key concepts of our graph-based perceptron are presented along with the changes from the original perceptron paradigm [258]. Finally, an extension to multiclass problems is detailed.

5.4.1 The perceptron choice

The choice of the perceptron may seem dated compared to recent deep neural network architectures [131]. However, four key points motivated this choice. The perceptron is deterministic, simple, capable of online learning and stackable. (i) Simple: The perceptron is a simple algorithm for binary classification in which the weights are adjusted in the direction of each misclassified example. (ii) Deterministic: Given a particular input, the algorithm will always produce the same output. In the graph domain, the parameter space is huge. Thus, having a reproducible process is appreciable. (iii) Online learning ability: The perceptron is capable of online learning (learning from samples one at a time). This is useful for larger datasets since there is no need to address the entire datasets in memory. (iv) Stackable: The simple perceptron neuron can be stacked to create a multilayer perceptron. The multilayer perceptron is a renown algorithm which is still in use in the latest deep learning architectures [131] and is especially involved in the dense layers.

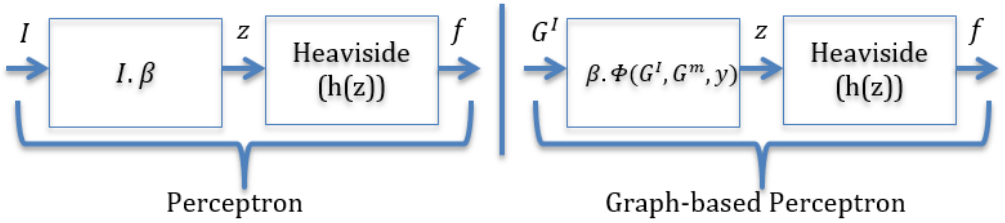


Figure 5.3: Overview of the perceptron and our proposal of a modified perceptron for graph

5.4.2 2-class graph-based perceptron

The perceptron is an algorithm [258] for learning a binary classifier $\mathcal{C} = \{0, 1\}$. In the context of neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function. A global picture of the graph-based perceptron is depicted in Figure 5.3. The conventional perceptron is adapted to graphs thanks to three main features : a) the learning rule for updating the weight vector β, b) the graph matching algorithm for finding y^*, and c) the graph model G_m.

Learning rule. The learning rule aims at modifying β. The weights should be updated in cases of wrong classifications. The correction must take into account the amount and the sign of the committed error.

$$\text{Learning rule: } \beta(t+1) = \beta(t) + \alpha(c*_k - c_k)\Phi(G_k, G_m, y^*) \tag{5.3}$$

To show the time-dependence of β, we use $\beta_i(t)$ as the weight at time t. The parameter α is the learning rate, where $0 < \alpha \leq 1$. $(c_k^* - c_k)$ is the error function. This error is positive if $(c_k^* > c_k)$ and negative if $(c_k^* < c_k)$. The learning rule is the steepest gradient descent. It attempts to reduce the error in the direction of the error descending along the gradient when considering the $\Phi(G_k, G_m, y^*)$ entries associated with weight β.

Graph matching solver. A graph matching algorithm is utilized in the computation of the function f as follows:

1. $y^* \leftarrow$ Solve Problem 10 with $d(G, G_m, y, \beta)$.
2. $f \leftarrow heaviside(\beta(t) \cdot \Phi^T(G_k, G_m, y^*))$

Many efficient approximate algorithms have been proposed to solve the graph matching problem defined in Problem 10. In [242, 31], Riesen et al. and Bougleux et al. have reformulated the Quadratic Assignment Problem of Problem 10 to a Linear Sum Assignment Problem (LSAP). Nodes of both graphs are involved in the assignment problem. A cost matrix is computed to enumerate pair-wise node distances. The LSAP can be solved in polynomial time which makes this approach very fast. To reduce the graph matching problem to a LSAP, local rather than global relationships are considered.

Graph model. The graph matching is computed between an input graph G_i and a graph model G_m. The choice of a graph model among a set of graphs is of primary interest. The graph model should represent the diversity of attributes and topologies that can be found in the graph set TrS. The graph model selection rules can be split into two categories according to whether the choice depends on a graph dissimilarity measure or not. From this perspective, two ways of choosing the graph model are proposed. First, the graph model is defined as follows : $G_m = \arg\max_{G \in TrS} |G|$. With $|G| = |V| + |E|$. Accordingly, G_L^m is the largest graph of the set. G_m may gather a large diversity of attributes along with different structures. Second, the set median

graph is defined as $G_m = \arg\min_{G \in TrS} \sum_{G_i \in TrS} d(G_i, G, y_i^*, \beta)$. However this definition relies on some predefined parameters β.

Now, let us design the learning algorithm of the graph-based perceptron. Algorithm 2 is a deterministic algorithm. $\#iter$ is the maximum number of iterations or also called *epochs* in the literature. The parametrized graph matching problem is solved in Line 5. Line 6 is the classification step while lines 8 to 11 apply the learning rule defined in Eq. 5.3 when the classification is wrong.

Algorithm 2: The learning graph-based perceptron scheme

```
   Data: TrS = {(G, c)} and G_m
   Data: #iter is the maximum number of iterations
   Data: α learning rate
   Result: Learned β. A weight vector
 1 β ← 1 and t ← 0
 2 while error > 0 and iter< #iter do
 3 |   error ← 0 and iter ← 0
 4 |   for G ∈ TrS do
 5 |   |   y* ← argmin_y  β(t) · Φ^T(G, G_m, y) // Solve problem 1
 6 |   |   c ← heaviside(β(t) · Φ^T(G, G_m, y*))
 7 |   |   c_k ← getLabel(G)
 8 |   |   if c_k − c != 0 then
 9 |   |   |   β(t + 1) ← β(t) + α(c_k − c)Φ(G, G_m, y*)
10 |   |   |   error← error +1
11 |   |   end
12 |   |   t ← t +1
13 |   end
14 |   error ← error/|TrS|
15 |   iter ← iter +1
16 end
```

5.4.3 Multiclass graph-based perceptron

The perceptron provides a natural extension of the multiclass problem. Instead of having only one neuron in the output layer, with binary output, we could have N neurons leading to multiclass classification. A set of functions $f(G, c)$ map each possible input/output pair to a real value that represents the fitness of the pair (G, c). The resulting score is used to choose among many possible outputs: $c^* = \text{argmin}_c f(G, c)$. To extend graph-based perceptron to the multiclass case, two main actions must be taken: a) the binary activation function must be replaced to gauge the fitness between the input and the class; b) the graph models must be computed for each class rather than the entire training dataset. *Activation function.* The Heaviside step function can only produce a binary output. To enable generalization, a rectified linear unit (ReLU) is employed, $\min(0, d(G, G_m, y^*, \beta))$. Well admitted, ReLU leads to faster training algorithms than standard sigmoid activation functions [167].
Graph model computation. The model graphs are specialized by class. They are computed on subsets that correspond to each class rather than the entire training set.

In Algorithm 3, the learning procedure of the multiclass graph-based perceptron is designed. In line 5, the selection of the most promising neuron is achieved by finding the minimum activation value (f_i) among all the neurons. Line 5 can be viewed as the classification step. Lines 8 and 9 are dedicated to weight updating. In the case of an incorrect classification, the misleading neuron (index i) must be pushed away from the expected neuron (index k). In line 8, the expected neuron weights (β_k) are reduced, which likely leads to the minimum activation value (f_k) in the next iteration. In the opposite way, in line 9, the misleading neuron weights (β_i) are increased to prevent the i^{th} neuron to be chosen during the next iteration.

Algorithm 3: Learning graph-based perceptron scheme in a multiclass context

Data: $TrS = \{(G, c)\}$ and $G_m^i \quad \forall i \in \{1, \cdots, |\mathcal{C}|\}$
Data: $\#iter$ is the maximum number of iterations and α learning rate
Result: Learned $\beta_i \quad \forall i \in \{1, \cdots, |\mathcal{C}|\}$. A weight vector for each class.

$\beta_i \leftarrow 1 \quad \forall i \in \{1, \cdots, |\mathcal{C}|\}$ and $t \leftarrow 0$
while *error* > 0 *and iter*$< \#iter$ **do**
  error $\leftarrow 0$ and iter $\leftarrow 0$
  for $G \in TrS$ **do**
    $c_i \leftarrow \arg \min_{o \in \{1, \cdots, |\mathcal{C}|\}} f_o(G, o)$
    $c_k \leftarrow getLabel(G)$
    if $c_k - c_i$ *!= 0* **then**
      $\beta_i(t+1) \leftarrow \beta_i(t) + \alpha\Phi(G, G_m^i, y^*)$
      $\beta_k(t+1) \leftarrow \beta_k(t) - \alpha\Phi(G, G_m^k, y^*)$
      error$\leftarrow$ error $+1$
    end
    t $\leftarrow$ t $+1$
  end
  error $\leftarrow$ error$/|TrS|$ and iter $\leftarrow$ iter $+1$
end

5.4.4 Complexity analysis

In this section, we conduct a complexity analysis for the described algorithms. Algorithm 2 is a deterministic algorithm whose complexity in terms of calls to the matching solver is $O(\#iter \cdot |TrS|)$ where $\#iter$ is the number of iterations. In addition, the entire test set (TeS) is classified by only $|TeS|$ calls to the graph matching algorithm. This linear complexity makes the decision procedure a fast graph classifier. Classical graph classifiers require generally $|TrS| \times |TeS|$ calls to the graph matching solver. Algorithm 3 is also a deterministic algorithm. The algorithm complexity in terms of calls to the matching solver is $O(\#iter \cdot |TrS| \cdot |\mathcal{C}|)$. $|\mathcal{C}|$ is the number of classes. The classification of the entire test set is performed by $|TeS| \times |\mathcal{C}|$ calls to the graph matching algorithm. Finally, the complexity of the graph matching solver used in our approach is $O((n+m)^3)$ with n and m the number of nodes in G_1 and G_2, respectively

5.4.5 Discussion

In this section, we point out the classification capability of our proposal as well as giving advantages and inconvenient of the method. First, the single layer perceptron architecture is only capable of learning linearly separable patterns. However our proposal can go beyond linearly separable features thanks to the graph space. The perceptron algorithm is operating on the graph set TrS through a change of variable from G to $\Phi(G, G_m, y^*)$ vector of minimal edit costs for each element in the model graph G_m. Then, the weights β are optimized with respect to classification loss through the perceptron algorithm. But β is also implied in finding y^* as $y^* = \underset{y}{\text{argmin}} \quad d(G, G_m, y, \beta)$. It means modifying β is not only acting on the linear projection of $\Phi(G, G_m, y^*)$ but also on the graph matching operator, which is a non-linear operation. Reinterpreting the minimization problem, it appears that both variables β and y are involved into the minimization of an empirical risk guided by the loss function l:

$$\min_{\beta, y} \sum_{(G_k, c_k) \in TrS} l(G_k, c_k, G_m, y_k, \beta) \tag{5.4}$$

$$l = \frac{1}{2}\left(c_k - heaviside(\beta \cdot \Phi^T(G^k, G_m, y_k^*))\right)^2 \tag{5.5}$$

We now describe the strengths and weaknesses of the proposed research method. The first strength of our approach is its adaptability. The graph matching solutions are adapted according to the

considered dataset because learning and matching are performed in concert. Second, our approach is discriminative. The similarity of graphs from the same class is increased, whereas the similarity of graphs from different classes is decreased. Third, According to the complexity analysis in the previous section, our approach is very fast in the classification stage because it requires one graph comparison per class. One drawback of the method is the *a priori* choice of model graph (G_m). However, this meta-parameter could be learned in a specific training phase.

5.5 Experiments

In this section, experiments are presented. First seven datasets are described along with the six methods compared in this experimental study. Finally, the protocol and the results are detailed.

5.5.1 Experimental setting

Dataset. Seven graph databases were chosen from the public IAM repository [245] and from the Tarragona graph repository for learning error-tolerant graph matching [201]. Each database consists of a set of graph instances that are divided in different classes, where each class is composed of a training set TrS and a test set TeS. The datasets are described in Table 5.2. These databases are representative of a wide range of learning problems that occur in Computer Vision. Matching functions d_v and d_e were taken from [244] and [201]. For the sake of reproducibility, the codes of the cost functions are available at https://sites.google.com/site/gperceptron along with a video demonstration.

Table 5.2: On the left side, summary of graph data set characteristics. On the right side, best parameters according to learning strategies R-1NN and C-1NN taken from Paper I [244] and from Paper II [201]

Database	size (TrS,TeS)	♯classes	node labels	edge labels	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	max $\lvert V\rvert$	max $\lvert E\rvert$	balanced
LETTER-HIGH	(750,750)	15	x,y	none	4.7	4.5	9	9	Y
LETTER-MED	(750,750)	15	x,y	none	4.7	3.2	9	9	Y
LETTER-LOW	(750,750)	15	x,y	none	4.7	3.1	9	9	Y
CMU	(71,70)	2	Shape	none	30	154.4	30	158	Y
GREC	(286,528)	22	x,y	Line types	11.5	12.2	25	30	Y
Fingerprint	(378,1533)	4	none	angle	5.42	4.42	26	24	N
COIL-DEL	(2400,1000)	100	x,y	none	21.5	54.2	77	222	Y

Table 5.3: Best parameters according to learning strategies R-1NN and C-1NN taken from Paper I [244] and from Paper II [201]

Database	R-1NN				C-1NN			
	α	K_n	K_e	From	α	K_n	K_e	From
LETTER-HIGH	0.9	1.7	0.75	Paper I	0.5	1	1	Paper II
LETTER-MED	0.7	1.9	0.75	Paper I	0.5	1	1	Paper II
LETTER-LOW	0.3	0.1	0.25	Paper I	0.5	1	1	Paper II
CMU	NA	NA	NA	NA	0.5	1000	1	Paper II
GREC	0.5	90	15	Paper I	NA	NA	NA	NA
Fingerprint	0.75	0.7	0.5	Paper I	NA	NA	NA	NA
COIL-DEL	NA	NA	NA	NA	NA	NA	NA	NA

Compared methods. A commonly used approach in pattern classification is based on nearest-neighbor classification. That is, an unknown object is assigned the class of its closest known

Table 5.4: Taxonomy of the compared methods

Classifier	Parameter learning	Model graph	Method Name
1NN	R [242]	M	R-M-1NN
		TrS	R-1NN
	C [60]	M	C-M-1NN
		TrS	C-1NN
Perceptron	Gradient descent	L	G-L-Perceptron
		M	G-M-Perceptron

element, or nearest neighbor (1NN). Two versions were utilized in the tests. R-1NN [242] and C-1NN [60] where the values of $K_e = \beta^e_{del} = \beta^e_{ins}$ and $K_n = \beta^n_{del} = \beta^n_{ins}$ were borrowed from [244] and [201], respectively. K_n corresponds to the cost of a node deletion or insertion, and K_e corresponds to the cost of an edge deletion or insertion. The aforementioned methods hold a meta parameter $\alpha \in [0, 1]$ which corresponds to the weighting parameter that controls whether the cost on the nodes or on the edges is more important.

In Table 5.2, the best values of parameters (α, K_n, K_e) are summarized. The learning method in C-1NN is based on the minimization of the distance between the graph matching solver and the human ground-truth correspondences. This approach is a generative approach while R-1NN is a discriminative one. The R-1NN learning scheme aims at finding the best parameters by performing a grid search over the parameter space. The 1NN classifier is known to be slow as the size of the training set grows. For fair competition with our algorithms, two fast classifiers were derived from R-1NN and C-1NN: R-M-1NN and C-M-1NN are 1NN classifiers for which the training is reduced to one set median graph per class. The set median graphs were computed with a graph distance function and the best learned parameters (see Table 5.3). This training set reduction may lead to a loss of accuracy but produced a large speed-up.

Our proposal is called G-Perceptron which stands for graph-based perceptron. Two versions have been derived according to whether the graph model is the largest graph or the set median graph, G-L-Percetron or G-M-Percetron, respectively. As mentioned in Section 5.4, predefined β parameters are required to compute the set median graphs of the G-M-Perceptron method. A vanilla plain solution was adopted with $\beta = \mathbf{1}$.A summary of the six compared methods is reported in Table 5.4. The G-Percetron, C-1NN and R-1NN classifiers were implemented in JAVA 7 and run on a 2.6 GHz computer with 8 GB RAM. Graph matching solvers involved in C-1NN and R-1NN are based on the Linear Sum Assignment Problem (LSAP) which is solved in our implementation by the Hungarian method. This setting makes possible a fair comparison.

Protocol. To assess the performance of our learning scheme and our new classifier, two experiments were performed. First, the impact of the learning rate α was studied on the LETTER-HIGH dataset and second, classifications were carried out on all datasets. To summarize the results of theses experiments, the classification rates (η) during the training and the test phases are reported along with the time, in **milliseconds**, for classifying all test instances. In our protocol, the retained β parameters for the test phase are those that maximize the classification rate on the training set. Finally, we define two metrics for accuracy and speed evaluation. Let $\mathcal{M}$ be the set of all methods.

$$AccDev^i = 100 * \frac{\max_{j \in \mathcal{M}} \eta^j_{TeS} - \eta^i_{TeS}}{\max_{j \in \mathcal{M}} \eta^j_{TeS}} \qquad TimeRatio^i = 100 * \frac{Time^i}{\min_{j \in \mathcal{M}} Time^j}$$

where η^i_{TeS} and $Time^i$ are the classification rate and the processing time for the method i. AccDev is the best when equal to zero. For TimeRatio, the closer to 100 it is, the better is the time performance.

5.5.2 Results

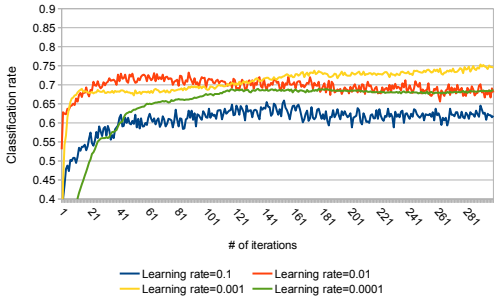
In Figure 5.4, the impact of the learning rate is depicted for the set median and large graph models. A very high learning rate ($\alpha = 0.1$) leads to poor results. The search space exploration is too fast and saddle points are missed. A high learning rate ($\alpha = 0.01$) leads to unstable results with many oscillations while a low learning rate implies a slow but smooth convergence. A trade-off can be achieved with an intermediate value ($\alpha = 0.001$). This value was the best for the set median and large graph models. For the rest of the experiments, $\alpha = 0.001$ was chosen and the number of iterations was set to 300. To continue the analysis of the learning ability of Algorithm 3, in Table 5.5, the classification rates obtained during the learning phase are tabulated (column η_{TrS}). The learning ability is demonstrated on all data sets. The classification rate is always higher on the training set than on the test set except on the CMU and Fingerprint databases where the number of classes is small. The loss between the training and test recognition rates is 3% on average. This result demonstrates the good generalization ability of our algorithms.
Now, we compare the set median and large graph models. In Table 5.6, classification rate deviations and time ratios are reported.
Median vs Large graph models. As expected, the classifier that is based on large graph models is slower than the one based on set median graphs. On average the G-M-Perceptron is approximatively two times faster than G-L-Perceptron while providing an accuracy gain of 7%. This gain in accuracy reaches 19% for LETTER-MED/HIGH. This is because the set median graph better models the intra-class distribution of a classification problem than the largest graph of a set. The set median graph is a better prototype or representative of a class. Now, we compare the 1NN based approaches to the G-M-Perceptron.
G-perceptron vs 1NN. On average, he G-M-Perceptron is 1% less accurate than R-1NN. However, the deviations from this mean are large between datasets. On Fingerprint G-M-Percetron is 24% more accurate than R-1NN whereas in GREC, G-M-Percetron is 24% less accurate than R-1NN. A plausible explanation is the lack of data on GREC dataset, with only 17 graphs by class on average, the number of weights (464) is too large and the learning algorithm fails to converge completely. Compared to the learning scheme of C-1NN, our approach is competitive with a small loss of accuracy of 2% on average on the CMU and LETTER databases. Second, in term of speed, on average, G-M-Perceptron is 66 times faster than 1NN classifiers. The time complexity of our graph-based perceptron is quadratic as a function of the test set size and the number of classes ($|TeS| \cdot |\mathcal{C}|$) whereas the complexity of any 1NN classifier grows quadratically as a function of the test and training set sizes ($|TrS| \cdot |TeS|$) with $|\mathcal{C}| << |TrS|$.
We compare the 1NN classifiers with reduced training sets to G-M-Perceptron. The training sets were reduced to one set median graph per class (M-1NN).
G-Perceptron vs M-1NN. As expected, M-1NN classifiers are fast. C-M-1NN is the fastest approach. This is because the graph matching solver is based on a fast LSAP solver. The speed gain is 8% on average compared to G-M-Perceptron. However, G-M-Perceptron is 5% more accurate on average than C-M-1NN. This result is experimental proof of the merit of learning discriminative matchings. G-M-Perceptron and R-M-1NN are based on the same graph matching solver and achieved very similar speed performances on the LETTER datasets. However on the GREC and Fingerprint databases G-M-Perceptron is 20% faster than R-M-1NN. The set median graphs are not the same for both approaches and on these difficult databases the median graph sizes do matter.

Generally, for the M-1NN classifiers, the set median graphs were selected after the learning of the best parameter values (see Table 5.3) while in the G-M-Perceptron case, the set median graphs were chosen from scratch with $\beta = \mathbf{1}$ and the parameter learning scheme had to compensate an eventual bad set median graph choice. Our experimental setting is in favor of the M-1NN classifiers. Finally, no reference parameters are available for the Coil-DEL dataset. However, this dataset is interesting for assessing the capability of our method dealing with a large number of classes (100). For instance, on this dataset, a 1NN classifier with $\beta = \mathbf{1}$ obtained a classification rate $\eta_{TeS} = 0.552$

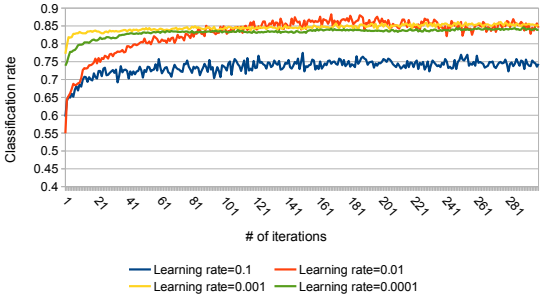
Table 5.5: Classification results. The best classification rates are marked in blue while the best processing times are in red. "Best" means an improvement over other methods statistically significant according to a z-test with a confidence level of 70%

	G-L-Perceptron			G-M-Perceptron			R-1NN		C-1NN		R-M-1NN		C-M-1NN	
Database	η_{TrS}	η_{TeS}	Time	η_{TrS}	η_{TeS}	Time	η_{TeS}	Time	η_{TeS}	Time	η_{TeS}	Time	η_{TeS}	Time
LETTER-LOW	0.97	0.95	2060	1	0.98	1436	0.99	69700	0.96	71869	0.97	1341	0.98	1341
LETTER-MED	0.66	0.64	2278	0.90	0.87	1404	0.92	74506	0.93	72150	0.86	1388	0.81	1310
LETTER-HIGH	0.75	0.70	2433	0.86	0.81	1669	0.83	86377	0.84	84911	0.82	1731	0.71	1498
CMU	1	0.99	11420	1	0.99	11029	NA	NA	0.99	427955	NA	NA	0.99	10780
GREC	0.84	0.70	23339	0.84	0.75	14370	0.98	199087	NA	NA	0.96	14726	NA	NA
Fingerprint	0.64	0.68	7768	0.74	0.76	1576	0.58	260816	NA	NA	0.73	2138	NA	NA
Coil-DEL	0.60	0.57	1091777	0.52	0.52	471575	NA	NA	NA	NA	NA	NA	NA	NA

in 13496127 ms. G-M-Perceptron can reach nearly the same accuracy ($\eta_{TeS} = 0.518$) while being 27 times faster. In contrast, a fast M-1NN classifier based on set median graphs with $\beta = \mathbf{1}$ achieved only a 1% recognition rate in 286416 ms.



(a) Large graph model



(b) Median graph model

Figure 5.4: Letter-HIGH : Impact of the learning rate on the convergence.

5.6 Conclusions and Perspectives

In this paper, a graph-based perceptron was proposed for learning discriminative graph matching in a classification context. Our proposal is supported by a formal definition and two deterministic algorithms. Graph matching was parametrized to build a weighted formulation. This weighted formulation was used to define a perceptron classifier, in which each neuron is composed of a graph model and a vector of parameters. Each weight is associated with a graph component of the graph

Table 5.6: Classification analysis. The top-2 of deviations are marked in blue while the 2 best time ratios (Time) are in red. Accuracy deviations (Acc) are in % (The lower the better). Time ratio is the best when equal to one (The lower the better).

	G-L-Perceptron		G-M-Perceptron		R-1NN		C-1NN		R-M-1NN		C-M-1NN	
Database	AccDev	TimeRatio	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
LETTER-LOW	3.91	1.54	0.81	1.07	0.00	51.98	2.70	53.59	1.62	1.00	0.94	1.00
LETTER-MED	31.38	1.74	6.30	1.07	1.58	56.87	0.00	55.08	7.74	1.06	13.04	1.00
LETTER-HIGH	16.64	1.62	3.80	1.11	1.58	57.66	0.00	56.68	3.01	1.16	16.16	1.00
CMU	0.00	1.06	0.00	1.02	NA	NA	0.00	39.70	NA	NA	0.00	1.00
GREC	29.42	1.62	23.85	1.00	0.00	13.85	NA	NA	2.69	1.02	NA	NA
Fingerprint	10.22	4.93	0.00	1.00	23.54	165.49	NA	NA	3.95	1.36	NA	NA
Coil-DEL	0.00	2.32	9.46	1.00	NA	NA	NA	NA	NA	NA	NA	NA

model. Weights are learned using the gradient descent algorithm. Two types of graph models were investigated, the set median graph and the large graph of a graph set. Classification results on 7 publicly available datasets demonstrated a large speed-up during the test phase (60 times faster in average) with a loss of accuracy of 6% on average compared to a 1-NN classifier based on an optimized graph distance. The results revealed that the set median graph was the best graph prototype for serving as graph model within a neuron.

In future work, the first objective will be to overcome the main drawback of set median graphs that require pre-defined weights to be computed. This could be achieved by looping the whole learning scheme. Learned weights could be used to compute better set median graphs. Finally, in the near future, we plan to extend our work to multiple layers and consequently to learn mid-level graph-based representations.

5.7 Follow up

Now that the perceptron paradigm has been brought to the graph-domain, let's work on the convolution paradigm. The following chapter presents an original approach to graph convolution and, more specifically, graph convolutional neural networks.

Chapter 6

Graph convolutional neural networks

Contents

6.1 Introduction . . . 75
6.2 Review of the literature . . . 76
6.3 Proposal . . . 78
6.3.1 Notation . . . 78
6.3.2 Graph matching . . . 78
6.3.3 Graph convolution based on graph matching . . . 79
6.3.4 Convolution layer . . . 80
6.3.5 About graph matching differentiation . . . 84
6.3.6 A "no edge matching" version of the graph convolution layer . . . 84
6.3.7 Graph pooling . . . 85
6.3.8 Hyperparameterization . . . 85
6.3.9 Choosing the graph matching solver . . . 86
6.4 Experimental work . . . 86
6.4.1 Baselines . . . 86
6.4.2 Data . . . 87
6.4.3 Parameterization . . . 87
6.4.4 Protocol . . . 88
6.4.5 Results . . . 89
6.5 Conclusion and perspectives . . . 90
6.6 Follow-up . . . 91

6.1 Introduction

Deep learning has been around since the beginning of this decade, enabling representation learning to a highly complex form. Convolutional neural networks (CNNs) are the most salient example when it comes to spatial deep learning. In a few decades, it has outperformed the existing state of the art methods [262].

However, in the way they were formalised, CNNs are bound to operate on euclidean spaces. Indeed, convolution is a signal operation that are defined on euclidean spaces. This has restricted deep learning main use to euclidean-defined data such as sound or image.

And yet, numerous computer application fields (among which network analysis, computational social science, chemo-informatics or computer graphics) induce non-euclideanly defined data such as graphs, networks or manifolds. Considering the aforementioned benefits on euclidean data, it goes without saying that extending CNNs to non-euclidean ones is a research direction worth considerated. To this extent, analogous operator to both convolution and pooling must be found for non-euclidean domains.

6.2 Review of the literature

There is no consensus on the convolution (operator $*$) definition on non-euclidean signals as several definitions arise from the literature [32]. This section offers a review of these existing definitions.

For the sake of simplicity, we will express all of these definitions in terms of graphs even though some are originally applied on riemannian manifolds. This choice was done not only to stay consistent with previous material presented in this thesis (for instance in Chapter 5), but also because graphs encompass most of the features that discrete riemannian geometry defines.

Convolution will be noted as $G_I * f$ with G_I the input graph and f the filter in its abstract form.

Definitions can be grouped in two different families: spectral and spatial.

Spectral definitions rely on spectral graph theory, which is the equivalent of Fourier analysis for graphs. The idea is to take advantage of the Convolution Theorem. The convolution theorem states that convolution on euclidean spaces in the spatial domain is equivalent to the Hadamard product in the frequency domain. It is then possible to multiply the two signals in the frequency domain to obtain the convolved signal. To apply convolution on a graph, the convolution theorem is in this case postulated on graph space. The structure that allows to work on the frequency domain of a graph is its Laplacian:

$$L = D - A$$

with D and A respectively being the degree and adjacency matrices.

The eigenvectors of the graph Laplacian act as the Fourier basis of the graph. Convolving over a graph from its frequency domain is as follows:

$$G_I * f = \Phi_I \text{diag}(\hat{f}) \Phi_I^\intercal M_I$$

where $\text{diag}(\hat{f})$ is the filter f in the spectral domain: the filter is here expressed as the eigenvalues of its Laplacian. M_I is the column vector of vertex attributes (size $n \times 1$ with n the number of vertices in G_I) in graph G_I and Φ_I are the Laplacian eigenvectors for graph G_I [32].

This method is used in [34]. However, getting access to the eigenbasis of the graph laplacian is time-consuming as it implies matrix inversion.

A way to avoid eigenanalysis is to consider our filter f as a polynomial of the Laplacian $f_\theta(L_I)$ (with L_I the laplacian for graph G_I and θ the polynomial parameter vector) [32, 68]:

$$G_I * f = f_\theta(L_I) M_I$$

$$f_\theta(L) = \sum_{k=0}^{r} \theta_k . L^k$$

[150, 68, 269] use this method, introducing different types of polynomials.

The major drawback of spectral approaches is sensitivity to domain changes: Similar graphs with slightly different topologies will respond very differently to a given filter. In other words, a single signal defined on anisomorph graphs will give different responses. This is due to the fact that constructing the Fourier basis depends on the graph structure.

The second type of graph convolution definition relies solely on spatial domain. The common point of these approaches is to consider the problem in a riemannian perspective: non-euclidean data can be seen as manifolds. This allows to reduce the local structure of graphs down to a compact euclidean space on which filters can be easily built: a patch operator $\mathcal{D}$ can be defined. The patch operator transforms the local space into a P-sized vector with respect to P weighting functions $(w_1, ..., w_P)$. $w_k(i)$ is the vector of weights for each node in the graph in the neighbourhood of i. The patch value at node i is defined as follows:

$$(\mathcal{D}G_I)_i = [w_k(i) M_I]_{1 \leq k \leq P}$$

These weights are defined depending on the locations of the different neighbours in a given local space.

$$w_k(i) = [e_k(\zeta_{ii'})]_{1 \leq i' \leq N}$$

where N is the number of nodes in the graph, $\zeta_{ii'}$ is the label for edge (i, i') (or the location of i' in the local space of i) and $e_k(\zeta_{ii'})$ is the weight for node i' in the neighbourhood of i.

Once a patch operator is applied, it is possible to define a convolution filter f as a vector $\mathbf{f} \in \mathbb{R}^P$. The result of the convolution at node i is as follows:

$$(G_I * f)_i = (\mathcal{D}G_I)_i . \mathbf{f}$$

Several approaches fit in this spatial framework [195, 30, 200]. The way they differ is how they define the weighting functions e_k.

[200] proposes to use parameterized gaussian kernels as weight functions:

$$e_k(\zeta_{ii'}) = \exp -\frac{1}{2}(\zeta_{ii'} - \mu_k)\Sigma_k^{-1}(\zeta_{ii'} - \mu_k)$$

where μ_k and Σ_k are the gaussian kernel parameters for the k-th weighting function. These parameters can be learned during the training of the neural network as it is included in the loss function and it is differentiable.

Nevertheless, most of the aforementioned approaches do not take full advantage of the graph topology. The graph structure is locally embedded into a vector space (i.e. the tangent space at a given point of a riemannian manifold). In a graph theory perspective, this means the notion of neighbourhood is limited to 1-hop graph neighbourhood which is bound to be a star structure.

On a conceptual level, previous approaches were rooted in manifold theory more than in graph theory [195, 30, 200]. The graphs were then considered as domains on which signals (labels) can be defined and convolved. In the frame of graph matching, graphs are considered as objects to be compared. These views are analogous as long as convolution does not modify the domain's structure, that is acting on the vertex labels.

Table 6.1: Frequently used notations

Notation	Description
G_I	An input graph
G_F	A filter graph
g_I^i	Neighbourhood subgraph rooted at vertex i in I
i, j	Vertices in graph G_I
ij	An edge in graph G_I between i and j
a	A vertex in G_F
ab	An edge in G_F between a and b
μ	Labelling function for vertices
ζ	Labelling function for edges
G_F^W	A filter graph and its associated weights
$\mu(a)$	Vertex label of a
$\mu^W(a)$	Vertex label of a parametrized by W
$\lvert\Omega_{ij}\rvert$	Cardinality of Ω_{ij}
δ_x^y	Kronecker delta of x and y

6.3 Proposal

Rather than relying on a purely euclidean definition of locality in graphs, we propose to stay in graph space by using a graph matching operator. A convolution filter (or kernel) is an attributed graph. The result of our convolution operator on an input graph is, for a given node, the matching similarity between its neighbourhood and the graph kernel. The intuition is that the local response of a convolution filter on euclidean data is analog to graph similarity between a node neighbourhood and the filter graph.

6.3.1 Notation

Frequently used notations are summarized in Table 6.1.

6.3.2 Graph matching

To define our convolution operator, we must define the graph matching function that will be pointwisely used.

6.3.2.1 Graph matching problem

Let G_1 and G_2 be attributed graphs: $G_1 = (V_1, E_1, \mu_1, \zeta_1)$ and $G_2 = (V_2, E_2, \mu_2, \zeta_2)$

$$\mathrm{GMS}(G_1, G_2) = \max_y \quad s(G_1, G_2, y), \tag{6.1a}$$

$$\text{subject to} \quad y \in \{0,1\}^{n_1 n_2} \tag{6.1b}$$

$$\sum_{i=1}^{n_1} y_{i,a} = 1 \quad \forall a \in [1, \cdots, n_2] \tag{6.1c}$$

$$\sum_{a=1}^{n_2} y_{i,a} \leq 1 \quad \forall i \in [1, \cdots, n_1] \tag{6.1d}$$

$$|V_1| \geq |V_2| \tag{6.1e}$$

The similarity function s is defined as follows:

$$s(G_1, G_2, y) = \sum_{y_{ia}=1} s_V(i, a) + \sum_{y_{ia}=1} \sum_{y_{jb}=1} s_E(ij, ab) \tag{6.2a}$$

$$s_V(i, a) = \mu_1(i).\mu_2(a) \tag{6.2b}$$

$$s_V(i, \epsilon) = s_V(\epsilon, a) = 0 \tag{6.2c}$$

$$s_E(ij, ab) = \zeta_1(ij).\zeta_2(ab) \tag{6.2d}$$

$$s_E(ij, \epsilon\epsilon) = s_E(\epsilon\epsilon, ab) = 0 \tag{6.2e}$$

Let $\pi(G_1, G_2, e)$ denote an assignment of element (edge or vertex) $e \in V_1 \cup E_1$ to some element in $V_2 \cup E_2 \cup \{\epsilon, \epsilon\epsilon\}$:

$$\pi(G_1, G_2, i) = a \iff \exists a \in V_2 : y_{ia} = 1 \tag{6.3a}$$

$$\pi(G_1, G_2, i) = \epsilon \iff \forall a \in V_2 : y_{ia} = 0 \tag{6.3b}$$

$$\pi(G_1, G_2, ij) = ab \iff \exists ab \in E_2 : y_{ia} = 1 \wedge y_{jb} = 1 \tag{6.3c}$$

$$\pi(G_1, G_2, ij) = \epsilon\epsilon \iff \forall a, b \in V_2 : y_{ia} = 0 \vee y_{jb} = 0 \tag{6.3d}$$

The similarity function can be rewritten as follows:

$$s(G_1, G_2, y) = \sum_{i \in V_1} s_V(i, \pi(G_1, G_2, i)) + \sum_{ij \in E_1} s_E(ij, \pi(G_1, G_2, ij)) \tag{6.4a}$$

6.3.3 Graph convolution based on graph matching

Now that our matching operator is formulated, we can apply it over an input graph to compute the result of a convolution.

Let G_I and G_F be attributed graphs: $G_I = (V_I, E_I, \mu_I, \zeta_I)$ and $G_F = (V_F, E_F, \mu_F, \zeta_F)$. G_I and G_F are respectively referred to as the input graph and the filter graph.

6.3.3.1 Graph convolution operator $\odot$

The graph convolution operator is a function $\mathbb{G} \times \mathbb{G} \to \mathbb{G}$ and is defined as follows:

$$G_I \odot G_F = (V_I, E_I, \mu, \zeta) \tag{6.5a}$$

$$\text{with} \quad \mu : V_I \to \mathbb{R} \quad \text{such that} \quad \mu(i) = \text{GMS}(g_I^i, G_F) \tag{6.5b}$$

$$\zeta : E_I \to \mathbb{R} \quad \text{such that} \quad \zeta(ij) = \text{score}(ij, G_I, G_F) \tag{6.5c}$$

. where g_I^i and *score* are defined as follows.

6.3.3.2 Vertex neighbourhood graph (l-hops)

g_I^i is defining the neighbourhood (which is a subgraph) for vertex i in G_I:

$$g_I^i = (N_I^l[i], E_I^i, \mu_I, \zeta_I) \tag{6.6a}$$

$$\text{with} \quad N_I^l[i] \text{ the } l\text{-hops closed neighbourhood of } i \text{ in } G_I \tag{6.6b}$$

$$\text{and} \quad E_I^i = \{kl \in E_I \quad \text{s.t.} \quad k, l \in N_I^l[i]\} \tag{6.6c}$$

6.3.3.3 Edge attribute in convolved graph

score is a function mapping an edge to its matching score in the found GMS. The problem is that it might be assigned multiple times:

$$\text{let} \quad \Omega_{ij} = \{g_I^k \quad \forall k \in V_I : ij \in g_I^k\} \quad \forall ij \in E_I \tag{6.7}$$

Ω_{ij} potentially contains more than one element. Therefore, score can be defined as follows:

$$\text{score}(ij, G_I, G_F) = \theta\left(\{s_E(ij, \pi(g_I, G_F, ij)) \quad \forall g_I \in \Omega_{ij}\}\right) \tag{6.8a}$$

$$\text{with} \quad \theta : \text{some statistical estimator (max or avg)} \tag{6.8b}$$

6.3.4 Convolution layer

Now that the convolution operator is defined, it is possible to use it as a base to build a convolution layer. This layer can be included in a graph neural network.

6.3.4.1 Graph convolution filter: the filter graph

A graph convolution filter is an attributed graph G_F^W. Its role is analogous to that of a vanilla CNN kernel: it modifies the output and gets modified through backpropagation. Every attribute function is parametrized with respect to a weight vector $W \in \mathbb{R}^{|V|+|E|}$.

$$G_F^W = (V_F, E_F, \mu_F^W, \zeta_F^W) \tag{6.9a}$$

$$\text{with} \quad \mu_F^W(a) = W_a \tag{6.9b}$$

$$\zeta_F^W(ab) = W_{ab} \tag{6.9c}$$

6.3.4.2 Graph convolution layer

A convolution layer is a set of convolution filters $\{G_F^p\}_{1 \leq p \leq n}$ applied on a same input graph G_I. The output of the layer consists of all filters results (analogous to euclidean convolution feature map) stacked up.

Let u be the output function of the layer s.t.:

$$u : \mathbb{G} \to \mathbb{G} \quad u(G_I) = \psi(\{u_p(G_I)\}_{1 \leq p \leq n}) \tag{6.10a}$$

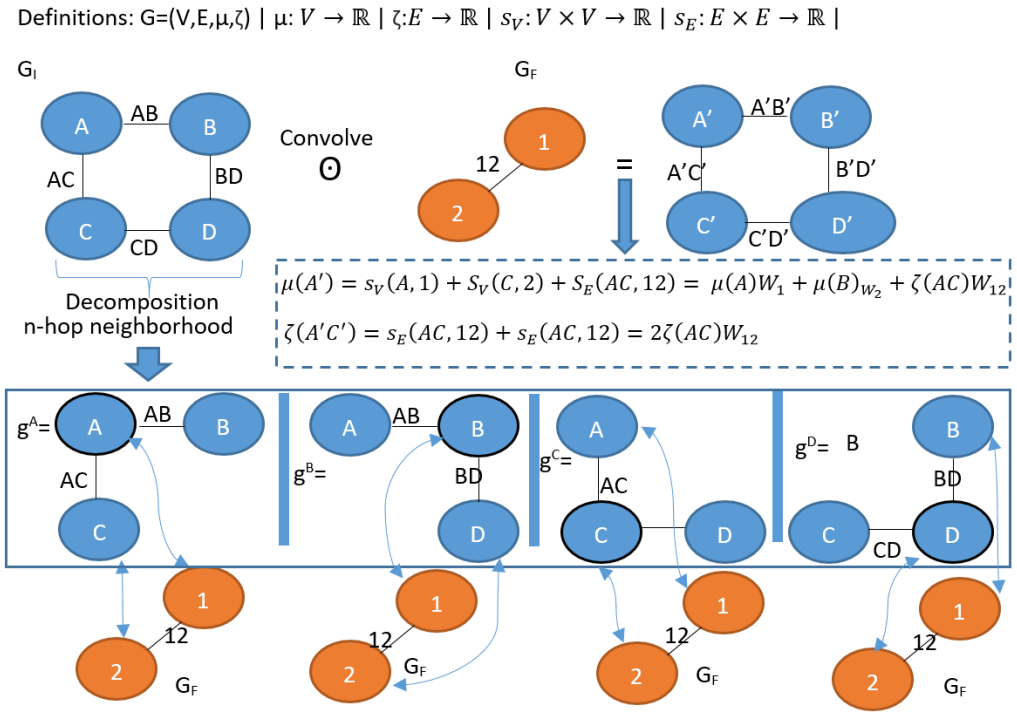


Figure 6.1: Computing graph convolution

$$\text{with} \quad \psi : \mathbb{G}^n \to \mathbb{G} \quad \psi(\{u_p(G_I)\}_{1 \leq p \leq n}) = (V_I, E_I, M, Z) \tag{6.10b}$$

$$M : V_I \to \mathbb{R}^n \quad (M(i))^p = \mu_F^p(i) \tag{6.10c}$$

$$Z : E_I \to \mathbb{R}^n \quad (Z(ij))^p = \zeta_F^p(ij) \tag{6.10d}$$

$$n \text{ the number of filters} \tag{6.10e}$$

ψ function keeps only a single graph structure and concatenates each vertex/edge attribute. The output function of the layer is a graph with same topology as G_I but with attributes as vectors composed by attributes of every filters outputs.

Graph convolution computation can be seen as a step-by-step process (shown in Figure 6.3.4.2). The first step is neighbourhood extraction: for each vertice i in G_I (the input graph), the neighbourhood graph g^i is extracted. It is composed of every neighbour of i in a given range (it can be 1-hop away but also n-hops away). g_i and G_F (the filter graph) are matched. The matching score $\text{GMS}(g^i, G_F)$ becomes the output of the convolution at i.

Definition 11. *Graph convolution differentiation Let $G_I = (V_I, E_I, \mu_I, \zeta_I)$ the input graph and $u(G_I)$ be the output of a convolution layer (called Conv) s.t. $u(G_I) = G_I \odot G_F^W$.*

$$u(G_I) = G_I \odot G_F^W \tag{6.11a}$$

$$= (V_I, E_I, \mu, \zeta) \tag{6.11b}$$

To simplify notations, let's consider the output of the Conv layer to be the vertex and edge labelling functions μ and ζ, as neither the vertices or edges sets change during convolution. The output will be noted as in Equation 6.12b

Let $\mathcal{J}$ be a loss function (for example mean-squared error or categorical cross-entropy). Let's suppose Conv is involved in the calculation of $\mathcal{J}$ such that:

$$\mathcal{J} = A \circ Conv \circ B$$

B and A respectively being the processing before and after Conv. In order to minimize $\mathcal{J}$, its gradient must be calculated with respect to W. This gradient will then be used to modify W itself. For calculus needs, let Conv be the output function of the Conv layer. This output function is defined w.r.t. the graph labeling functions:

$$B(W', X) = [\mu_I, \zeta_I] \quad \textit{where } X \textit{ is a training example} \tag{6.12a}$$

$$Conv(W, [\mu_I, \zeta_I]) = [\mu, \zeta] \tag{6.12b}$$

$$A([\mu, \zeta]) = \hat{Y} \quad s.t.\ \hat{Y} \in \mathbb{R} \tag{6.12c}$$

$$with\ [\mu_I, \zeta_I] = [\mu_I(i) \quad \forall i \in V_I \quad \cdots \quad \zeta_I(ij) \quad \forall ij \in E_I] \tag{6.12d}$$

$$and\ [\mu, \zeta] = [\mu(i) \quad \forall i \in V_I \quad \cdots \quad \zeta(ij) \quad \forall ij \in E_I] \tag{6.12e}$$

The error gradient for W is calculated using chain derivative:

$$\frac{\partial \mathcal{J}}{\partial W} = \frac{\partial A \circ Conv \circ B}{\partial W} \tag{6.13a}$$

$$= \frac{\partial A}{\partial Conv} \times \frac{\partial Conv}{\partial W} \qquad \left(\tfrac{\partial \mathcal{J}}{\partial W}\ \textit{doesn't depend on}\ \tfrac{\partial B}{\partial W}\right) \tag{6.13b}$$

$$= \frac{\partial \hat{Y}}{\partial [\mu, \zeta]} \times \frac{\partial [\mu, \zeta]}{\partial W} \tag{6.13c}$$

Let's assume $\frac{\partial \hat{Y}}{\partial [\mu,\zeta]}$ exists and is known. Therefore, only $\frac{\partial [\mu,\zeta]}{\partial W}$ is to be calculated.
First of all, we need to expand $[\mu, \zeta]$. Let's expand μ first for a given vertex $i \in G_I$:

$$\mu(i) = GMS(g_I^i, G_F^W) \tag{6.14a}$$

$$= \left(\max_y \quad s(g_I^i, G_F^W, y) \right) \tag{6.14b}$$

$$= s(g_I^i, G_F^W, \hat{y}) \quad with\ \hat{y} = \operatorname*{argmax}_y \quad s(g_I^i, G_F^W, y) \tag{6.14c}$$

$$= \sum_{k \in V_I^i} s_V(k, \pi(g_I^i, G_F^W, k)) + \sum_{kl \in E_I^i} s_E(kl, \pi(G_I^i, G_F^W, kl)) \tag{6.14d}$$

$$\mu(i) = \sum_{k \in V_I^i} \mu_I(k).W_{\pi(g_I^i, G_F, k)} + \sum_{kl \in E_I^i} \zeta_I(kl).W_{\pi(g_I^i, G_F, kl)} \tag{6.14e}$$

Then let's expand ζ:

$$\zeta(ij) = score(ij, G_I, G_F^W) \tag{6.14f}$$

$$\zeta(ij) = \theta\left(\{s_E(ij, \pi(g_I, G_F^W, ij)) \quad \forall g_I \in \Omega_{ij}\}\right) \tag{6.14g}$$

If θ is max*, the same rewriting as in Equation 6.14c applies:*

$$\zeta(ij) = s_E(ij, \pi(g_I^*, G_F^W, ij)) \tag{6.14h}$$

$$\text{with } g_I^* = \underset{g_I \in \Omega_{ij}}{\text{argmax}} \quad s_E(ij, \pi(g_I, G_F^W, ij)) \tag{6.14i}$$

$$\zeta(ij) = \zeta_I(ij).W_{\pi(g_I^*, G_F^W, ij)} \tag{6.14j}$$

If θ is avg:

$$\zeta(ij) = \frac{1}{|\Omega_{ij}|} \left(\sum_{g_I \in \Omega_{ij}} \zeta_I(ij).W_{\pi(g_I, G_F^W, ij)} \right) \tag{6.14k}$$

Let's differentiate μ with respect to W. $\forall i \in V_I$:

$$\frac{\partial \mu(i)}{\partial W_a} = \mu_I(\pi(G_F^W, g_I^i, a)) \qquad \forall a \in V_F \tag{6.15a}$$

$$\frac{\partial \mu(i)}{\partial W_{ab}} = \zeta_I(\pi(G_F^W, g_I^i, ab)) \qquad \forall ab \in E_F \tag{6.15b}$$

Now let's differentiate ζ. $\forall ij \in E_I$ and $\forall ab \in E_F$:
If θ is max*:*

$$\frac{\partial \zeta(ij)}{\partial W_{ab}} = \zeta_I(\pi(g_I^*, G_F^W, ab)) \tag{6.16a}$$

If θ is avg:

$$\frac{\partial \zeta(ij)}{\partial W_{ab}} = \frac{1}{|\Omega_{ij}|} \sum_{g_I \in \Omega_{ij}} \zeta_I(\pi(g_I, G_F^W, ab)) \tag{6.17a}$$

In any case:

$$\frac{\partial \zeta(ij)}{\partial W_a} = 0 \quad \forall a \in V_I \tag{6.18a}$$

Now, $\frac{\partial \mathcal{J}}{\partial W}$ can be calculated:

$$\frac{\partial \mathcal{J}}{\partial W_a} = \frac{\partial \hat{Y}}{\partial [\mu, \zeta]} \times \frac{\partial [\mu, \zeta]}{\partial W_a} \qquad \forall a \in V_F \tag{6.19a}$$

$$= \sum_{i \in V_I} \frac{\partial \hat{Y}}{\partial \mu(i)} \frac{\partial \mu(i)}{\partial W_a} + \sum_{ij \in E_I} \frac{\partial \hat{Y}}{\partial \zeta(ij)} \frac{\partial \zeta(ij)}{\partial W_a} \tag{6.19b}$$

$$= \sum_{i \in V_I} \frac{\partial \hat{Y}}{\partial \mu(i)} \frac{\partial \mu(i)}{\partial W_a} \qquad \textit{w.r.t. Eq 6.18a} \tag{6.19c}$$

$$\frac{\partial \mathcal{J}}{\partial W_{ab}} = \frac{\partial \hat{Y}}{\partial [\mu, \zeta]} \times \frac{\partial [\mu, \zeta]}{\partial W_{ab}} \qquad \forall ab \in E_F \tag{6.20a}$$

$$= \sum_{i \in V_I} \frac{\partial \mathcal{J}}{\partial \mu(i)} \frac{\partial \zeta(i)}{\partial W_{ab}} + \sum_{ij \in E_I} \frac{\partial \mathcal{J}}{\partial \zeta(ij)} \frac{\partial \zeta(ij)}{\partial W_{ab}} \tag{6.20b}$$

Finally, let's suppose B is parameterized with vector W'. In this case, $\frac{\partial \mathcal{J}}{\partial W'}$ is to be calculated:

$$\frac{\partial \mathcal{J}}{\partial W'} = \frac{\partial B \circ Conv \circ A}{\partial W'} \tag{6.21a}$$

$$= \frac{\partial A}{\partial Conv} \times \frac{\partial Conv}{\partial B} \times \frac{\partial B}{\partial W'} \tag{6.21b}$$

$$= \frac{\partial \hat{Y}}{\partial [\mu, \zeta]} \times \frac{\partial [\mu, \zeta]}{\partial [\mu_I, \zeta_I]} \times \frac{\partial [\mu_I, \zeta_I]}{\partial W'} \tag{6.21c}$$

Let's assume $\frac{\partial [\mu_I, \zeta_I]}{\partial W'}$ exists and is known. $\frac{\partial \hat{Y}}{\partial [\mu, \zeta]}$ has already been evaluated. Therefore, only $\frac{\partial [\mu, \zeta]}{\partial [\mu_I, \zeta_I]}$ is to be calculated. $\forall i \in V_I$:

$$\frac{\partial \mu(i)}{\partial \mu_I(k)} = \begin{cases} W_{\pi(g_I^i, G_F^W, k)} & \text{if } k \in V_I^i \\ 0 & \text{else} \end{cases} \qquad \forall k \in V_I \tag{6.22a}$$

$$\frac{\partial \mu(i)}{\partial \zeta_I(kl)} = \begin{cases} W_{\pi(g_I^i, G_F^W, kl)} & \text{if } k \in V_I^i \\ 0 & \text{else} \end{cases} \qquad \forall kl \in E_I \tag{6.22b}$$

If θ is max, $\forall kl \in E_I, \forall ij \in E_I$*:*

$$\frac{\partial \zeta(ij)}{\partial \zeta_I(kl)} = \delta_k^i \delta_j^l W_{\pi(g_I^*, G_F^W, kl)} \tag{6.23a}$$

If θ is avg, $\forall kl \in E_I, \forall ij \in E_I$:

$$\frac{\partial \zeta(ij)}{\partial \zeta_I(kl)} = \delta_k^i \delta_j^l \frac{1}{|\Omega_{ij}|} \sum_{g_I \in \Omega_{ij}} W_{\pi(g_I, G_F^W, ij)} \tag{6.24a}$$

In any cases:

$$\frac{\partial \zeta(ij)}{\partial \mu_I(k)} = 0 \quad \forall k \in V_I \tag{6.25}$$

6.3.5 About graph matching differentiation

In Definition 11, a differentiation of the convolution operator is proposed. This differentiation does not take into account the dependencies between the optimal graph matching $\hat{y}$ and the variables $\{\mu_I(k)\}_{k \in E_I}$ and $\{\zeta_I(kl)\}_{kl \in V_I}$. As these variables are used to calculate the possible matchings, it is trivial to conclude such dependencies exist. Nevertheless, the matching solver in use (see Subsection 6.3.9) is not differentiable, at least a priori. We therefore assumed $\hat{y}$ as a constant in the gradient calculus with respect to these variables by means of change of variable in Eq. 6.14c.

6.3.6 A "no edge matching" version of the graph convolution layer

This section presents a degraded model. It ignores topology at a local level by not matching edges. It therefore reduces the graph matching problem to a node assignment problem inside a given neighborhood. One concern on this simplification could be that we do not take advantage of the graphs topology. However, topology information is used when computing vertices neighbourhoods. Additionally, this model has lower time complexity as edge information is not taken into account (see details in Subsection 6.3.9.)

Used graphs are 3-uplets (V, E, μ) and the similarity function is simplified as follows:

$$s(G_1, G_2, y) = \sum_{y_{ia}=1} s_V(i, a) \tag{6.26a}$$

$$s_V(i, a) = \mu_1(i).\mu_2(a) \tag{6.26b}$$

As a consequence of the edge attributes deletion in the filter graph, its parameter becomes vector $W \in \mathbb{R}^{|V|}$ (as many parameters as vertex). The filter is defined as follows:

$$G_F^W = (V_F, E_F, \mu_F^W) \tag{6.27a}$$

$$\text{with} \quad \mu_F^W(a) = W_a \tag{6.27b}$$

The output function of the filter $u : \mathbb{G} \to \mathbb{G}$ is defined as follows:

$$u(G_I) = G_I \odot G_F^W \tag{6.28a}$$
$$= (V_I, E_I, \mu) \tag{6.28b}$$

6.3.7 Graph pooling

As in euclidean convolutional neural nets, we want to implement not only convolutional layers but also pooling/downsampling layers. In the existing literature, downsampling is view as graph coarsening [32]. A recurrent graph coarsening algorithm choice seems to be Graclus [73] (used in [200, 68]).

We propose to use a community detection algorithm (Louvain method [27]) as the base of our graph pooling layer. Louvain method deals with weighted graphs. In our case, edge weights are computed by scalar products of involved vertices. This choice is brought by the following intuition: the higher nodes attributes scalar product get, the more these vertices probabilities to fall in the same cluster increases (because a higher scalar product implies vector similarity).

6.3.8 Hyperparameterization

As in any neural network, graph neural networks have parameters that won't be optimized from gradient descent.

The first one is the graph filter (its number of nodes and adjacency matrix). The number of nodes in the graph filter is analogous to the size of a classic convolution kernel. A 3×3 kernel filter is equivalent to a 9 nodes filter graph with grid-like adjacency. The second hyperparameter is the size of extracted neighbourhoods graphs which is the maximum node distance in a given node neighbourhood. A 2-hop-sized neighbourhoods will include nodes that can be reached from the origin node in two hops or less.

These hyperparameters could be optimized through grid or random search. However, to restrain our study, we will consider the following postulate: a graph filter should be congruent with extracted neighbourhoods. In other words, the two should have equal sizes and identical topologies as much as possible. This postulate comes from classic graph conv where each kernel coefficient is matched with one and only one image coefficient.

6.3.9 Choosing the graph matching solver

The algorithm for solving the graph matching problem is a critical element for the model. The first reason is that it is potentially the highest in complexity since graph matching problems are up to NP-hard. Additionally, graph matching is solved as many times as there are vertices in the input graph (the size of every problem to solve being that of every vertex neighbourhood).

We opted for a bipartite graph matching algorithm [242]. Complexity of such an algorithm is among the lowest (polynomial time) for solving error-tolerant graph matching problems suboptimally.

Bipartite graph matching algorithm reduces graph matching to vertex matching by embedding an estimation for edge costs in the vertex costs. This edge cost estimation is computed by solving an edge-assignment problem for every node-matching possibility. Therefore, BP has to solve as many matching problems as there are edge-costs.

We used a variant of BP called Square Fast BP [275] where the cost matrix for vertex matching is of size $\max(|g_I|, |G_F|) \times \max(|g_I|, |G_F|)$ with $|G_F|$ and $|g_I|$ being number of vertices in filter graph G_F and neighbourhood graph g_I. Assuming both neighbourhood and filter graphs are complete, a matching problem complexity is $O(\max(|g_I|, |G_F|)^3)$.

As a consequence, worst case complexity with fast bipartite matching is the following:

$$O(\max(|g_I|, |G_F|)^5)$$

Some preliminary experiments showed impracticable computation time of the full model. As a first workaround, the experimental part of this chapter will focus on "no edge matching" model. This workaround allowed to keep processing to an acceptable level (that is suitable for small classification experiments). Edge cost estimation by edge matching is no longer required. The simplified model has the following pointwise complexity:

$$O(\max(|g_I|, |G_F|)^3)$$

6.4 Experimental work

In this section, we test the model according to several parameters. We want to test our model with a simple classification task on MNIST digit images.

6.4.1 Baselines

Our approach was compared with two other approaches:

- Vanilla CNN layer
- Monti et al. [200] mixture model graph CNN.

Same network topology was used for all approaches. It consists of classical ConvPool blocks linearly connected. Figure 6.2 shows the exact network structure in use. In case of graph convolution, $n \times n$ convolution filters equivalents are n^2 nodes filters and 2×2 pooling becomes 4 nodes pooling. n is set depending on average graph connectivity in a given dataset: if the average number of neighbours in a given dataset is 9, $n = 9$.

The last layer is a global pooling one. As in the euclidean case, it consists in aggregating each filter feature map in one scalar value. In our case, feature maps are aggregated by taking its average value.

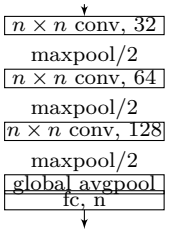


Figure 6.2: Network structure used for graph convolution experiments

6.4.2 Data

Quantitative experiments in this section are operated on digit images of MNIST dataset [168]. We chose this dataset as this was in use in the graph convolution literature. MNIST is a good "hello world" machine learning (ML) dataset. MNIST helps at quickly iterating on the learning model. Performance information gathered from experiments on MNIST can be great for judging how the model might perform on much harder and larger datasets like ImageNet.

In addition to the original MNIST dataset, a rotated version was used [160]. To compare results with MNIST-rotated, MNIST-original has to be modified as follows. MNIST-reduced proportions are unusual: 10000, 2000 and 50000 images respectively for train, valid and test whereas MNIST-original has 60000 and 10000 images respectively for train/valid and test. We used MNIST-reduced, a resampled version of MNIST-original to fit MNIST-rotated ratio between subsets cardinalities: MNIST-reduced and MNIST-rotated have both 10000, 2000 and 50000 images respectively for train, valid and test. All the set cardinalities are summed up in Table 6.2. Note that the test set of MNIST-reduced is larger than the training set by a factor 5 consequently, the generalization ability is better assessed.

Table 6.2: Different MNIST-based graph datasets

Dataset	Training set	Validation set	Testing set
MNIST-original	48 000	12 000	10 000
MNIST-rotated MNIST-reduced MNIST-mixed	10 000	2 000	50 000

Lastly, to test rotation invariance, a third MNIST-based dataset was added: MNIST-mixed. It was generated by combining MNIST-reduced train and valid sets and MNIST-rotated test set. It is design so that the models are trained on rotation-free images but tested on rotated images.

As MNIST is an image dataset, a graph-based representation of images has to be chosen. Representations used in Monti et al. [200] are superpixels graphs and grid graphs. We used $\frac{1}{4}$ grids (28×28 images resized to 14×14) and generated 75 superpixels Region Adjacency Graphs (RAG) using SLIC algorithm [1] with superpixel adjacency (see Table 6.3). Sample graphs are depicted in Figure 6.3.

6.4.3 Parameterization

Following hyperparameters were set after preliminary tests were conducted: Models are trained during 50 epochs using Adaptive Moment (Adam) gradient descent (learning rate 10^{-3}). Neighbourhood reach in use is 1-hop and filter size was set in accordance with average neighbourhood size (9 nodes).

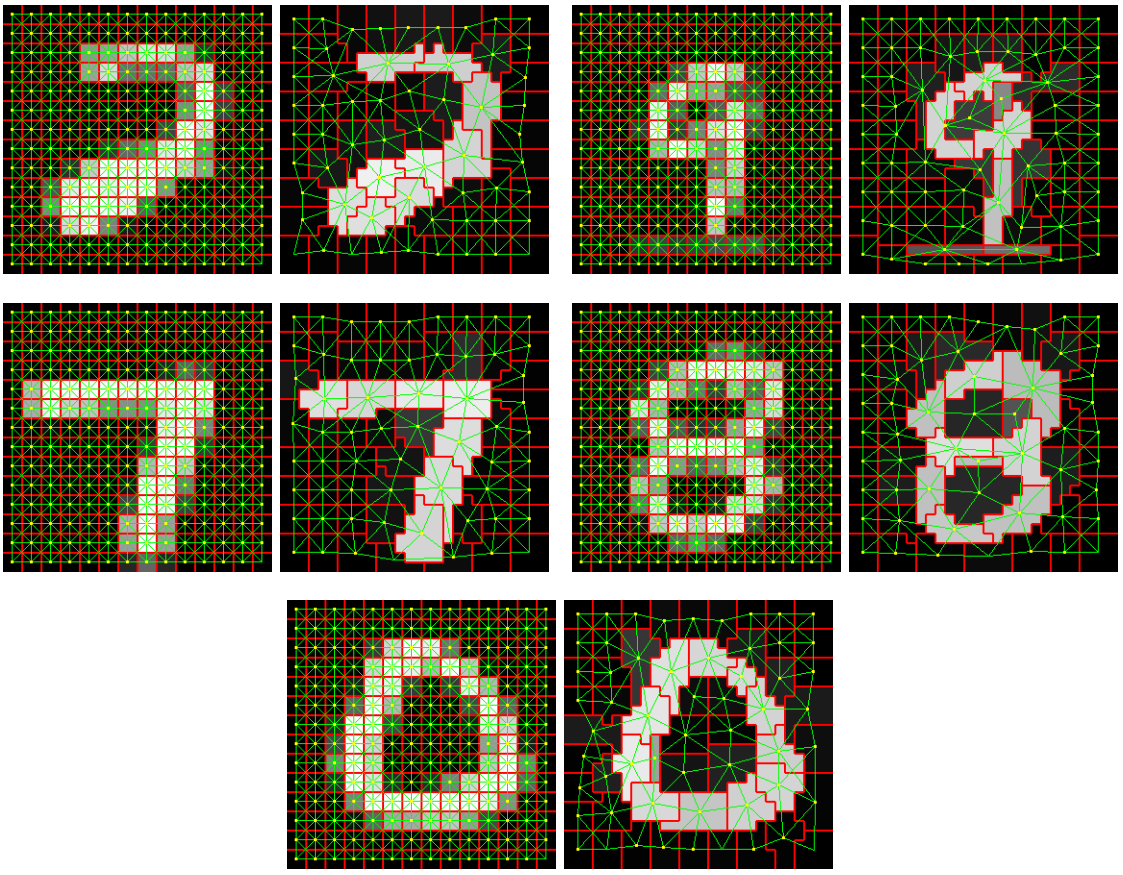


Figure 6.3: MNIST graphs. Left is $\frac{1}{4}$ grid, right is 75 superpixels RAG. Red symbolizes vertex frontiers and green shows edges.

6.4.4 Protocol

Following experiments were conducted:

Experiment 1 Models are tested on MNIST digit images classification task

Experiment 2 Several neighbourhood connectivities are tested on our model (1 and 2 hops)

Experiment 3 Rotation invariance is investigated. Spatial information for our datasets is conveyed by edge attributes. In such a frame, as our "no edges" model ignores edge attributes, it is theoretically rotation-invariant. Experiment 3 aims at experimentally validating this claim. This is done by training models on unrotated images and testing on rotated ones. MNIST-mixed set is used to this end.

Experiment 4 A sample filter is visualized on some MNIST example images

Experiment 5 Graph based methods are tested on regular grids and on irregular graphs (75 superpixels RAG) for testing sensitivity to domain changes

As stated before and because of technical limitations, experiments involving MNIST datasets will focus on the two first MNIST classes (referred to as MNIST-2class)

Table 6.3: MNIST representations

Representation	**Nb nodes**	**Vertex attributes**	**Edge attributes**	**Connectivity**
$\frac{1}{4}$ grid	14^2	Pixel intensities	Relative polar coordinates	8-connectivity
75 superpixels	75 (average)	Average superpixel intensities		1-hop

Table 6.4: Recognition rates on MNIST 2class

Representation	**Dataset**	**CNN**		**MoNet**		**Ours**	
		Valid	Test	Valid	Test	Valid	Test
$\frac{1}{4}$ grid	MNIST reduced	100 %	99.88 %	97.56 %	99.40 %	99.51 %	97.76 %
	MNIST mixed	100 %	89.87 %	97.76 %	88.90 %	99.27 %	95.63 %
75 superpixels	MNIST reduced			94.13 %	92.70 %	94.13 %	89.53 %
	MNIST mixed			94.13 %	92.90 %	94.62 %	94.17 %

6.4.5 Results

Results on MNIST-2class are listed on Table 6.4. Results include classification from both $\frac{1}{4}$ grid graphs and SLIC 75-superpixels graphs. This table shows results for each dataset using classic CNN, MoNet [200] and our method.

6.4.5.1 Experiment 1: MNIST

On MNIST-2class, our model competes in a 3% margin with used baselines.

6.4.5.2 Experiment 2: Neighbourhood size

Extending the neighbourhood size did not have any significant effect on performance (see Table 6.5)

6.4.5.3 Experiment 3: Rotation invariance

On MNIST-mixed, no performance loss was observed on testing for our method. This is especially visible on grid graphs results where only classic CNN and MoNet show a 10 percent loss. A trivial explanation of how is this invariance obtained is that our graphconv filters are non-oriented because edge attributes are ignored.

6.4.5.4 Experiment 4: Visualizing graph conv on images

As an additional experimental material, we tried to visualize the result of a handcrafted filter on images. As for euclidean convolution, the most straightforward filter operation is edge detection.

Table 6.5: Recognition rates for different neighbourhood sizes on MNIST reduced 2 class

Representation	1 hop		2 hops	
	Valid	Test	Valid	Test
$\frac{1}{4}$ grid	99.02%	97.55%	98.04%	96.47%
75 superpixels	97.55%	93.74%	96.82%	93.62%

This is usually done by using Sobel operator that calculates intensity gradient at each spatial point of the image.

A potential equivalent graph conv filter is $(-1 \;\; 1)$ (the filter is a 2-nodes graph with respective attributes -1 and 1.) The intuition behind this filter is that the nodes will be matched respectively to the lowest (for the attributed -1 node) and highest (for the attributed 1 node) intensities. As a consequence, this filter will find the highest node attribute difference in every node neighbourhood, making it a sort of eager edge detection filter.

We applied this filter on grid graphs to visualize the output graph as an image (as the graph-to-image transformation is trivial). Figure 6.4 shows example applications of this filter on both original and rotated examples. This last figure suggests rotation invariance.



Figure 6.4: MNIST graph conv examples (respectively original, convoluted and rotated convoluted versions)

6.4.5.5 Experiment 5: Testing graphconv across domain

A particular concern on graph convolution operators is sensitivity to domain changes, i.e. capacity to identify similarities on irregular graphs. Both graph convolution tested show little performance loss between regular (grids) and irregular (75 superpixels RAG) results.

6.4.5.6 Training duration

As mentioned in Subsection 6.3.9, complexity of the model makes experiment tedious to lead. Epoch durations are given in Table 6.6.

6.5 Conclusion and perspectives

In this chapter, a graph convolutional neural network layer is proposed and tested in a simplified form.

Table 6.6: Epoch durations on MNIST 2class (Models use different implementations/hardware: CNN is Keras on GPU, MoNet is Theano on GPU and Ours is Keras on CPU)

Representation	**CNN**	**MoNet**	**Ours**
$\frac{1}{4}$ grid	1s	1s	17min 29s
75 superpixels	NA	1s	2min 42s

Our model performance is at state of the art level on simple tasks. It shows robustness with respect to graph domain changes.

Following improvements could highly benefit to performances and computational costs. The bipartite solver is not the most suitable choice for our use. Complexity seems to be too high for an efficient application. Using a less complex solver would allow the full model to be used in practice and applied to larger graphs. Using the edge information would probably enhance performances significantly. Moreover, it will probably help with solving more complex problems.

Another point of improvement is regarding differentiation: the solver operator is not differentiable. The gradient must then be approximated by neglecting contribution of the solver intermediary states. Finding a differentiable solver would enhance trainability of the model.

Finally, investigating our downsampling layer would justify a whole study for itself. It would be interesting to study the quality of the downsampled graphs but also to study the effect of weighting edges regarding vertex similarity.

6.6 Follow-up

All previously presented approaches are to be experimented on insect image recognition. The next chapter will present final data and results using these approaches.

Chapter 7

Experimental results on insect recognition

Contents

7.1 Introduction . . . 93
7.2 Datasets . . . 93
7.2.1 IRBI . . . 94
7.2.2 ImageNet-arthropods . . . 95
7.3 Models . . . 95
7.4 Model parameterization . . . 96
7.5 Protocol . . . 96
7.6 Results . . . 97
7.6.1 Experiment 1: Euclidean classification . . . 97
7.6.2 Experiment 2: Advanced study on transfer learning . . . 97
7.6.3 Experiment 3: Graph classification . . . 99
7.6.4 Experiment 4: Misclassification analysis . . . 99
7.7 Conclusion . . . 100

7.1 Introduction

This section focuses on showing results on insect image recognition. In a first time, datasets in use are presented. Results obtained with models involved in the thesis are presented and analysed. Misclassified elements are investigated in order to get insights on how the models dealt with this problem.

7.2 Datasets

The target domain is that of insect images and the task is to recognize the class they have been labeled in. Two sets have been used to that extent:

- IRBI, a lab-based dataset built by CETU Innophyt biologists according to their needs and taxonomy
- ImageNet-arthropods, a subset of ImageNet

7.2.1 IRBI

IRBI is a lab-based insect images set. It has been designed in CETU Innophyt[1] according to a special taxonomy. This taxonomy is aimed at biodiversity and pest management. To these ends, it has two main criteria: categories must facilitate human recognition (so that the recognition can be done with reduced learning) and they gather insects with similar dietary behaviours (so that categories give information whether a given individual is a pest or not).

Insects are captured into soapy liquid traps put at soil level. The captured specimens are then conserved in an alcohol solution before being identified. These same specimens are laid down on a plain background under a constant and controlled lighting. Multiple shots are taken for a single individual : they are taken along 3 different orientation of the individual and with 7 different smartphones which makes up 21 shots for a single insect.

The set in its first version is made of 3156 images among 37 different classes (based on an applied taxonomy). As mentioned before, the cardinalities are strongly unbalanced (from 7 to 370 images per class with an average of 85 images ± 71). The dataset split into train/valid/test sets has been performed at the insect level, in order to prevent the presence of an insect in two (or more) sets with different orientations, which would make the evaluation unfair. Besides, cardinalities discrepancies call for a stratified sampling at the class level. Stratified sampling consists in sampling (or splitting) a population not on a global scale but on its subpopulation scales (on classes scales in our case). As applied here, it guarantees class distribution equivalence between train/valid/test.

Example images are shown in Figure 7.1.



Figure 7.1: IRBI image examples

[1] https://innophyt.univ-tours.fr/ (in French)

7.2.2 ImageNet-arthropods

To have an analogous dataset with pictures taken in field-based settings, a subset of ImageNet was extracted: each leaf-synsets under the synset "arthropods" was used which make up 501 classes. In order to emulate the same constraints as in a real entomology set, the average class cardinality was lowered down to the average class cardinality of the IRBI set (see Table 7.1). The statistical validation method in use in this study is a 5-fold stratified cross-validation. To be able to apply such a split while keeping individuals from every class in the two sub-sets, classes with less than 5 individuals were removed.

Example images are shown in Figure 7.2.



Figure 7.2: ImageNet-arthropods image examples

Table 7.1: The image datasets

Dataset	Nb of classes	μ(Card(c))	σ(Card(c))	min(Card(c))	max(Card(c))
IRBI	30	85	71	33	370
ImageNet-arthropods	443	96	78	6	392

7.3 Models

The following models were used to solve insect image classification on the aforementioned datasets:

- SIFTBoW : a SIFT codebook representation combined with a SVM classifier.
- VGG16 as of Chapter 4
- Our graph convolutional neural network model as of Chapter 6.

7.4 Model parameterization

Following hyperparameters were set:

For SIFTBoW, the SVM was optimized with a grid-search approach on the validation dataset. RBF and linear kernels were tested with penalty parameter values $C \in \{10^0, \ldots, 10^6\}$. RBF kernel was tested with gamma parameter values $\gamma \in \{10^{-6}, \ldots, 10^0\}$.

Several VGG16-based models are presented:

frsc: learnt from scratch with initial random weights (during 200 epochs);

fitu: pre-trained network on ImageNet-1000 (during 50 epochs), and fine tuned on IRBI/Arthropods;

fitu/w: same as **fitu**, with example weighting based on each class cardinality (see Section 4.4.3)

Graph convolutional neural network architecture in use is shown in Figure 7.3. Filter and hop sizes (respectively 9 nodes and 1-hop) are fixed according to section 6.3.8. Numbers of filters were set empirically, one can notice only last conv layer was increased compared to architecture in use in section 6.4.

Images in insect image datasets are too big to be transformed into grid graphs. For this reason, 75 superpixels RAG is the only representation used.

For CNN and graph CNN, images were cropped to their center squared parts. As CNN requires fixed size input, images were downscaled to 224×224.

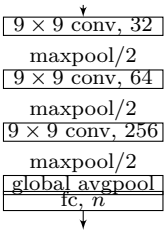


Figure 7.3: Network structure used for graph convolution network

7.5 Protocol

Following experiments were conducted:

Experiment 1 Images are classified in the euclidean image with SIFTBoW and VGG16 models. Results are obtained through a 5-fold cross-validation process. This method ensures statistical significance of presented results (because less depending on train/test data split).

Experiment 2 Advanced study on transfer learning is lead: influence of number of finetuned layers on performance is studied

Experiment 3 Images are classified through graph convolution network.

Experiment 4 Best model errors are investigated through visualizations.

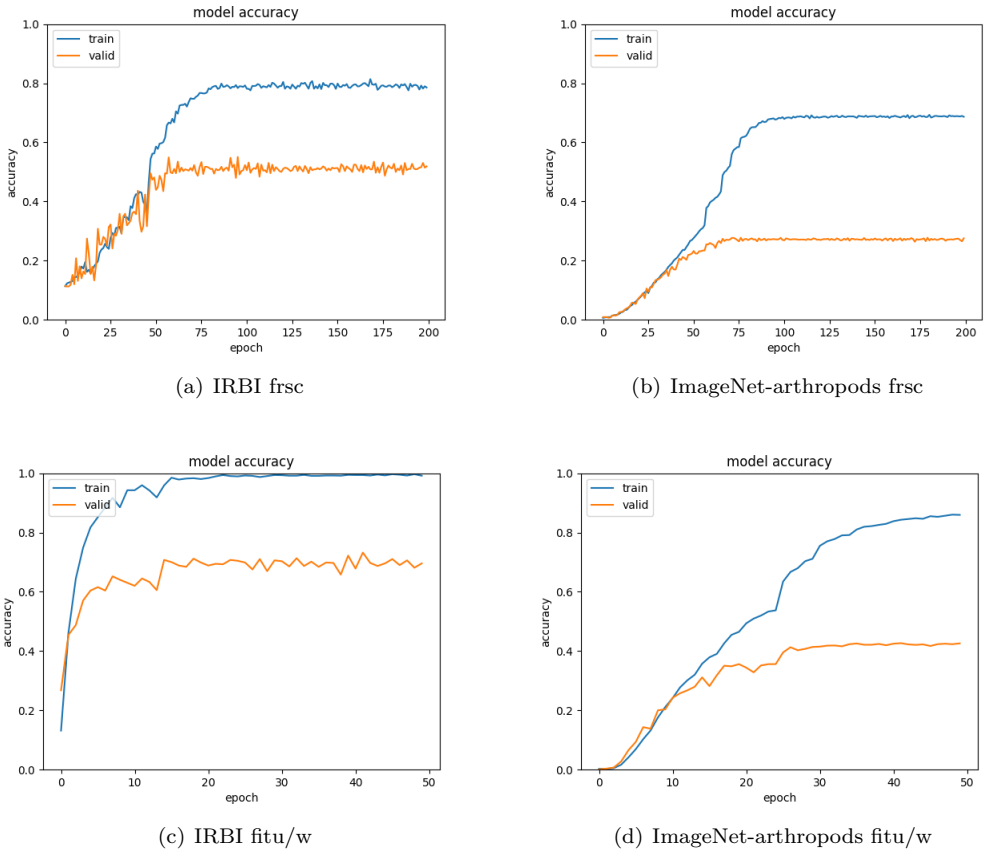
For experiments 1, 2 and 3, Top-k refers to the percentage of correct answers within the top k class probabilities returned by every model.

For experiments 2 and 3, simple train/valid/test split was used (respectively 80%, 10% and 10%).

7.6 Results

7.6.1 Experiment 1: Euclidean classification

Euclidean classification results are presented in Table 7.2. Additionally, the training curves (representing the recognition rate on validation set at each epoch) for VGG-16 both frsc and fitu/w are shown on Figure 7.4.



(a) IRBI frsc

(b) ImageNet-arthropods frsc

(c) IRBI fitu/w

(d) ImageNet-arthropods fitu/w

Figure 7.4: Learning curves for VGG-16 (blue for train accuracy, orange for valid accuracy).

Several observations can be made on these experiments results. First, the traditional approach based on handcrafted features (SIFTBow) performs moderately. Second, the VGG-16 architecture trained from scratch (VGG16-frsc) on the target dataset slightly outperforms the traditional BoW approach. On the other hand, the VGG-16 fine-tuned with transfer learning (VGG-16-fitu) gives significantly higher results compared to those of the traditional approach Additionally, weighting training examples based on class cardinalities (VGG16-fitu/w) made the model better on average regarding the top-1 figures.

7.6.2 Experiment 2: Advanced study on transfer learning

In a second time, an advanced study was conducted on the behavior and efficiency of transfer learning and CNN. In this study two factors are investigated: the number of learning examples

Table 7.2: Recognition rates on 5-fold experiments

Model	**IRBI**		**ImageNet-arthropods**	
	Top-1	Top-5	Top-1	Top-5
SIFTBoW	52.3 % ± 3.7	82.7 % ± 3.3	11.7 % ± 0.2	25.9 % ± 0.4
VGG16-frsc	54.0 % ± 5.0	84.9 % ± 3.0	26.9 % ± 0.7	50.1 % ± 0.7
VGG16-fitu	72.0 % ± 3.2	92.1 % ± 1.1	42.7 % ± 0.9	69.4 % ± 0.6
VGG16-fitu/w	73.6 % ± 1.8	92.4 % ± 2.2	43.5 % ± 1.1	71.3 % ± 0.8

and the number of layer to be learnt. The results for one fold are shown in Figure 7.5. The dataset used in this frame is IRBI. While the effect of changing the number of training example is not so surprising, it is interesting to note that learning only on the last 7 layers seems enough to reach a 70 % accuracy score. This observation can be interpreted as follows: the ImageNet-1000 features are generic enough (up to the $22 - 7 = 15$th layer) to learn on the lab-based insect identification problem. Learning only the 7 last layers is in fact learning the last VGG16 block (block5) and the two dense layers. Also, it is equivalent to learn about half of the network weights. This experiment corroborates the statement made in Section 1 that the first layers can be considered as a generic feature extractor while the last convolutional block and the dense layer represents the task-dependent features.

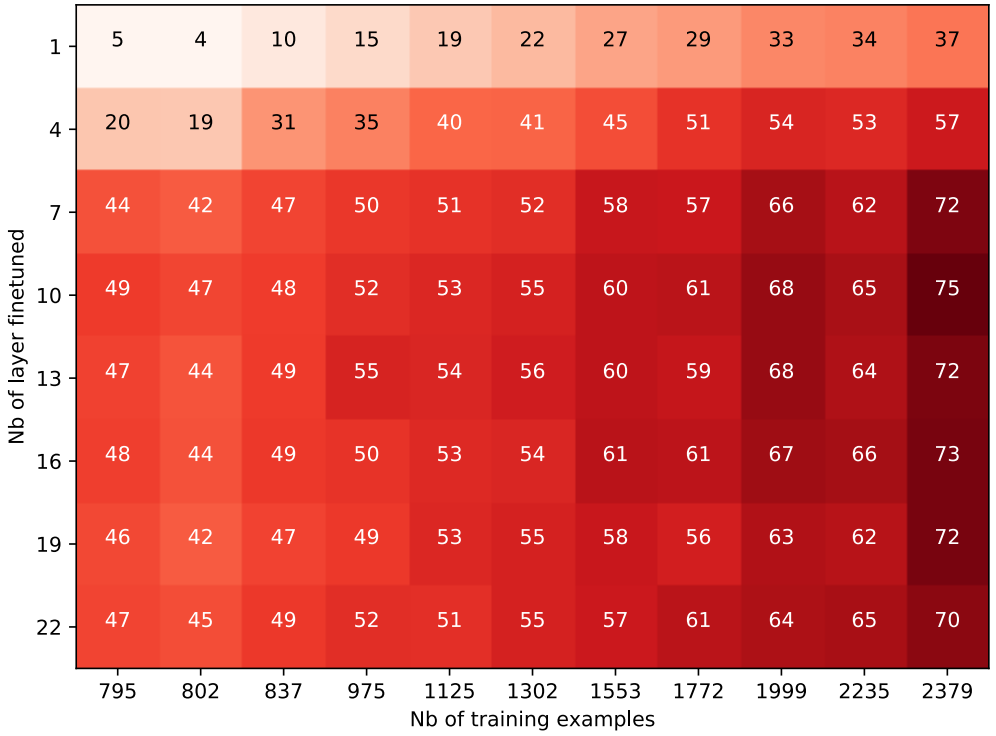


Figure 7.5: Recognition rates on the testing set depending on the number of training examples and the number of layer finetuned

Table 7.3 shows detailed results fine-tuning VGG16 model for both IRBI and ImageNet. These results show that the training phase is sped up by a factor of two by learning only the 7 last layers instead of the whole network.

Table 7.3: Recognition rates on 5-fold VGG16 transfer learning and mean epoch times

Learnt layers	**IRBI**			**ImageNet-arthropods**		
	Top-1	Top-5	mean time (s)	Top-1	Top-5	mean time (s)
All layers	73.6 % ± 1.8	92.4 % ± 2.2	102.6	43.5 % ± 1.1	71.3 % ± 0.8	1473.2
7 last layers	72.4 % ± 2.8	92.6 % ± 2.1	52.4	43.3 % ± 0.6	71.8 % ± 0.4	721.6

Table 7.4: Graph representations in use for insect recognition

Representation	**Nb nodes**	**Vertex attributes**	**Edge attributes**	**Connectivity**
75 superpixels	75 (average)	Average superpixel intensities	None	1 hop

7.6.3 Experiment 3: Graph classification

In this experiment, insect images were classified using a graph representation (detailed in Table 7.4)Graph-based classification results are listed in Table 7.5. Our approach shows reduced performance. A possible reason is that edge attributes are not taken into account (because using the "no edges" model), resulting in less discriminative filters.

7.6.4 Experiment 4: Misclassification analysis

Finally, some observations were made at the query level to have an insight on what are the possible causes of misclassification. This analysis was performed on model which yielded best results: VGG16-fitu/w.

The problem is that it is very tedious to get an idea of how CNNs discriminate between classes. In [328] two main methods classes are listed : 1) to display the convolution filters, 2) to use the filter responses as a tool to generate images. The drawback of such a method is that the produced images are often abstract and therefore hard to interpret. A possible solution to the interpretation issue is to rely on the database images rather than trying to make up images. In this extent, we adopted the following method: for each misclassified image, we retrieve the output vector of the CNN and try to find the images whose output vectors are nearest in the sense of vector space distance. This enables to work in a Content-Based Image Retrieval-like framework and listing the k-nearest neighbors images for a given query. Examples of query results produced by using this method are shown on Figure 7.6. Figure 7.6(a) shows a misclassification occurrence which is likely to be caused by the small apparent size of the insect. Each of the nearest images were from different classes yet quite resembling due to the low resolution of the capture. Figure 7.6(b) shows an example of a specimen mimicking another insect: syrphid flies colors enable them to be confused with bees or wasps suggesting they could sting a predator.

Table 7.5: Recognition rates on graph cnn experiments

Dataset	**IRBI**	
Model	Top-1	Top-5
Our graphcnn	30.40 %	71.03 %



(a) Small specimen confused with other small specimens

(b) Resembling classes

Figure 7.6: Misclassified samples. Left big image is the query image and right small images are the 5-nearest neighbors.

7.7 Conclusion

Several models were applied to insect image indentification datasets. CNNs show good performance although these can certainly be enhanced further by means of architecture modification for instance. Transfer learning definitely helps to compensate the low amount of data as performances clearly outperform traditional approach when using the transfer learning trick. Also, we have found that training only the last convolutional block of VGG16 is sufficient to find an almost optimal solution. This can lead to the conclusion that ImageNet-1000 features are generic enough up to a high level of abstraction. Finally, weighting the examples to tackle the unbalanced learning problem did influence positively the results but only marginally.

Our Graph CNN approach does not seem to match classical CNN performances on insect image identification task. However, the graph convolutional network model in use is way smaller (3 conv layers) than VGG16. This suggests good results when complexity reduction is obtained.

Datasets were created during the thesis. Although IRBI dataset is not aimed at public use, ImageNet-arthropods is a field-based insect image dataset easily generated and can be used as a general benchmark set publicly usable.

Chapter 8

Conclusions and perspectives

Contents

8.1 Conclusion 101
8.2 Perspectives 102
8.2.1 To go further 102
8.2.2 Other potential research directions 103

This chapter sets conclusion over the work achieved during the thesis and tries to envision perspectives that are relevant to this thesis but also to insect image identification in general.

8.1 Conclusion

In this thesis, several contributions are presented:

The first of them is a thorough review of literature. It shows various families of methods that have been used to tackle the insect image recognition problem. While discussing the nature of this problem, scientific locks are formalised and blind spots of the existing literature are pointed out. The most salient and global blind spot in the literature is performance evaluation: Each study have their own datasets and it is therefore tedious to compare studies in terms of accuracy. Finally this review sets the research direction that shaped the rest of the thesis.

Two distinct directions are investigated.

The first one is applying a standard state-of-the-art image classification pipeline to insect image recognition: Convolutional Neural Networks with transfer learning.

The second one calls for embedding images onto the graph space. Graph perceptron and graph convolutional neural networks approaches are proposed.

Finally, CNN-based and graph-convolution-based methods are tested on insect image recognition. Datasets in use are presented. One of them is a dataset (ImageNet-based) that is proposed so that any future study on insect image recognition can be compared on equivalent data.

Table 8.1 lists proposed methods in this thesis, along with some of their characteristics: Time complexities, differentiations, stackabilities and performance on IRBI set.

Time complexity is a crucial factor because it not only gives insights on answering time but also training handiness. The longer the training phase lasts, the more tedious empirical and automated hyper-optimization become.

Using fully-differentiable operators ensures quality of the loss gradient, resulting in efficient

Table 8.1: Comparisons of contributions proposed in this PhD thesis

Contribution	**Complexity** ($\|I\|$ and $\|W\|$ resp. being sizes of input and weight set)	**Fully differentiated**	**Endomorphic**	**Performance on IRBI**
CNN w/transfer learning (Chapter 4)	$O(\|I\|.\|W\|)$	Yes	Yes	74%
Graph-perceptron (Chapter 5)	$O(\|I\|.\|W\|^5)$	No	No	-
GCNN (Chapter 6)	$O(\max(\|I\|, \|W\|)^5)$	No	Yes	30%

convergence to optimal weights. Graph-based approaches proposed in this thesis are only partially differentiated. Although convergence could be observed, it is likely that better quality gradient could lead to better weights.

Endomorphism means the defined operator outputs data in the same domain as the input it takes. Endomorphism allows to stack layers, as data type remains constant. As for CNN methods, this opens possibility to learn representation hierarchically.

Finally, results on IRBI for GCNN are not satifying, although model used for this method is way smaller than that used for CNN and no transfer learning was used. Lower time complexity, apart from already mentioned advantages, could enable using bigger models and apply transfer learning on GCNN.

8.2 Perspectives

This section tries to state possible research paths to continue the already done work but also tackling the insect image recognition problem in general.

8.2.1 To go further

Contributions presented in this thesis are to be improved. Here are some possible follow-up.

8.2.1.1 CNN-based recognition

Although giving best results, our CNN pipeline surely can be improved. One possible improvement is to include a segmentation phase. Even further, insect detection might be an interesting feature, especially if users want to batch-process insects with the help of multi-individuals images. This could be formalised through generic object detection. Object detection consists of finding multiple objects in images. Found objects are encoded by their predicted classes and bounding boxes.

Recent deep-based methods can be categorized into two groups [92]: region proposal based methods [118, 117, 237] and single shot methods [235, 177]. Region proposal based methods use region detection algorithms as part of their workflows: regions of interest are detected through some adhoc algorithm or some CNN. These regions are then classified. Single shot methods decompose the image along a grid. This grid is used to compose several regions. A CNN not only classifies these template-based regions but also adjusts the region of interest position by regressing a bounding

box offset. While single shot methods are faster, region proposal based are more accurate in localization.

Such a switch from plain image classification to object detection would imply a new multi-individuals dataset with bounding box and class labeling. This would therefore be more tedious as a data collection campaign even though the workflow could be separated into a) drawing bounding boxes b) identifying the enclosed objects. While the second one would have to be done by biologists or experienced workers, the first one would necessitate no particular qualification.

8.2.1.2 Graph CNN

The second axis of possible follow-up is about graph convolution: Graph CNN model as of Chapter 6 is not implemented in its full form. Making the full model practicable would be an obvious follow-up.

A first element that is to be studied is the graph matching solver, as discussed in conclusion of Chapter 6. Its limitations are two-fold: a) its complexity leads to prohibitory computation time and b) its derivative is tedious to compute. An alternative solver is to be found that is both a) of near-linear complexity and b) easily differentiable. Additionally, it would be possible to relax the graph matching problems to work on a continuous domain. This would imply a given vertex/edge could be matched to many elements.

Another possible improvement is around working on full-GPU implementation of the model. Our current solver is indeed CPU-based. GPU-based implementation would certainly improve computation time significantly.

8.2.2 Other potential research directions

A question that we should mention at this point is « Is the path taken relevant, or is it worth exploring other directions? » Results don't suggest that insect image recognition as a classification problem is any different from generic image classification. This is the case because generic image classification methods such as CNNs seem to take over and give good results. If this hypothesis is confirmed, trying to come up with special classification methods is not relevant.

An other way to work around insect image recognition would be to focus on other of these characteristics. Here are some alternative research directions that seem relevant to insect image recognition.

8.2.2.1 Working against high intraclass variability

As discussed in Chapter 2, image insect recognition as a classification problem shows high intra class variability. In the context of an insect recognition application, it might be interesting to the user not only having the identified class but also very similar known individuals in the database. This would increase user trust in the system because it would allow them to check for answer correctness likelihood. The way to select resembling individuals is to define a distance measure between image pairs.

Although a distance metric could be extracted from the CNN used in Chapter 4 (for instance from last fully-connected feature map), insect image recognition could be rephrased as a content-based image retrieval problem. The model wouldn't be in charge of discriminating between classes but rather learning an insect images distance metric. Deep metric learning is achieved through siamese method [54]. A siamese method consists of a neural net that maps a given image to a vector. To get the distance between two images, one must obtain the images vectors v_1 and v_2 computed by this neural net. The value of the metric is the L2 norm of $v_1 - v_2$. This L2 norm is also the loss used during training.

8.2.2.2 Handling classes with no known individuals

Insect image recognition problems can become very complex. Depending on the taxonomical scale in use, classes can get highly numerous. Even further, it is likely that some classes were never encountered during learning phase.

This aspect of the problem could be solved by means of incremental learning algorithm. Literature on incremental deep learning is to our knowledge very recent and rare. There are nevertheless some attempts to formalise incremental frameworks that can be used in deep learning context [332, 107, 303, 48, 85]. These attempts propose new weights update rules or backpropagation that suit incremental learning.

Alternatively, be seen as zero-shot learning, where some classes don't have individuals in the learning set. Such a task is fulfilled by using a priori knowledge. This a priori knowledge usually represents semantics between all classes [306]. Among recent work on this topic, [306] combines CNN with Graph Convolution Networks [150]: the CNN part handles the image and the GCN processes a priori knowledge under the form of a graph.

In our context, a priori knowledge could emerge from biologist semantical knowledge. The way to both formalise and encode such knowledge data remains unsure.

List of publications

Journal articles

[191] Chloé Martineau et al. “A survey on image-based insect classification”. In: *Pattern Recognition* 65 (2017), pp. 273–284. ISSN: 0031-3203.

[194] Chloé Martineau et al. “Learning error-correcting graph matching with a multiclass neural network”. In: *Pattern Recognition Letters* (2018). ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.03.031. URL: http://www.sciencedirect.com/science/article/pii/S0167865518301107.

In proceedings

[192] Chloé Martineau et al. “Approches connexionnistes pour la reconnaissance d’images d’insectes”. In: *Journées ORASIS’17*. Colleville-sur-Mer, France, 2017.

[193] Chloé Martineau et al. “Effective Training of Convolutional Neural Networks for Insect Image Recognition”. In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2018, pp. 426–437.

[232] Romain Raveaux et al. “Learning Graph Matching with a Graph-Based Perceptron in a Classification Context”. In: *Graph-Based Representations in Pattern Recognition - 11th IAPR-TC-15 International Workshop, GbRPR 2017, Proceedings*. 2017, pp. 49–58.

Bibliography

[1] Radhakrishna Achanta et al. “SLIC superpixels compared to state-of-the-art superpixel methods”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282.

[2] H. A. Almohamad and S. O. Duffuaa. “A Linear Programming Approach for the Weighted Graph Matching Problem”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 15.5 (1993), pp. 522–525.

[3] A. Alpher. “Frobnication”. In: *Journal of Foo* 12.1 (2002), pp. 234–778.

[4] A. Alpher and J. P. N. Fotheringham-Smythe. “Frobnication revisited”. In: *Journal of Foo* 13.1 (2003), pp. 234–778.

[5] A. Alpher, J. P. N. Fotheringham-Smythe, and G. Gamow. “Can a machine frobnicate?” In: *Journal of Foo* 14.1 (2004), pp. 234–778.

[6] Alan N Andersen. “The use of ant communities to evaluate change in Australian terrestrial ecosystems: a review and a recipe”. In: *Proceedings of the Ecological Society of Australia*. Vol. 16. 1990, pp. 347–357.

[7] Kurt M. Anstreicher and Nathan W. Brixius. “A new bound for the quadratic assignment problem based on convex quadratic programming”. In: *Mathematical Programming* 89 (2001), pp. 341–357.

[8] Tom Arbuckle et al. “Biodiversity informatics in action: identification and monitoring of bee species using ABIS”. In: *Proc. 15th Int. Symp. Informatics for Environmental Protection*. Vol. 1. 2001, pp. 425–430.

[9] I. Arnault et al. *Method for classification and identification of morphological species*. 2015.

[10] Siti Noorul Asiah Hassan et al. “Vision Based Entomology : A Survey”. In: *International Journal of Computer Science & Engineering Survey* 5.1 (Feb. 2014), pp. 19–32. ISSN: 09763252, 09762760. (Visited on 02/04/2016).

[11] Authors. *Frobnication tutorial*. Supplied as additional material `tr.pdf`. 2006.

[12] Authors. *The frobnicatable foo filter*. ECCV06 submission ID 324. Supplied as additional material `eccv06.pdf`. 2006.

[13] László Babai. “Graph Isomorphism in Quasipolynomial Time”. In: *CoRR* abs/1512.03547 (2015).

[14] Gökhan H. Bakır, Alexander Zien, and Koji Tsuda. “Learning to Find Graph Pre-images”. In: *Pattern Recognition*. Ed. by Carl Edward Rasmussen et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 253–261. ISBN: 978-3-540-28649-3.

[15] Amit Kumar Banerjee et al. “Classification and identification of mosquito species using artificial neural networks”. In: *Computational Biology and Chemistry* 32.6 (2008), pp. 442–447. ISSN: 1476-9271.

[16] P. Baptiste et al. “Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances”. In: *J. Scheduling* 13.1 (2010), pp. 39–47.

[17] Yaniv Bar et al. “Deep learning with non-medical training used for chest pathology identification”. In: *Proc. SPIE, Medical Imaging: Computer-Aided Diagnosis* 9414 (2015), pp. 94140V–7.

[18] Mokhtar S. Bazaraa and Hanif D. Sherali. “On the Use of Exact and Heuristic Cutting Plane Methods for the Quadratic Assignment Problem”. In: *Journal of the Operational Research Society* 33.11 (Nov. 1982), pp. 991–1003. ISSN: 1476-9360. DOI: 10.1057/jors.1982.210. URL: https://doi.org/10.1057/jors.1982.210.

[19] S. Belharbi et al. “Spotting L3 slice in CT scans using deep convolutional network and transfer learning”. In: *Computers in Biology and Medicine* 87 (2017), pp. 95–103.

[20] Yoshua Bengio. “Learning deep architectures for AI”. In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[21] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. “Advances in Optimizing Recurrent Networks”. In: *CoRR* abs/1212.0901 (2012).

[22] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[23] Yosua Bengio. *Response to Gary Marcus - December 26th 2019*. 2019. URL: https://docs.google.com/document/d/1P9YyZ4xEDO98qPTa1Al4RnTqxj3mMiCvQE4i3hZkthY (visited on 01/14/2020).

[24] S. Berretti, A. Del Bimbo, and E. Vicario. “Efficient matching and indexing of graph models in content-based retrieval”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23.10 (2001), pp. 1089–1105.

[25] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[26] Mark Blaxter. “Molecular systematics: counting angels with DNA”. In: *Nature* 421.6919 (2003), pp. 122–124.

[27] Vincent D. Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), p. 10008. DOI: 10.1088/1742-5468/2008/10/P10008. arXiv: 0803.0476 [physics.soc-ph].

[28] Eric Bonabeau. “Graph multidimensional scaling with self-organizing maps”. In: *Inf. Sci.* 143.1-4 (2002), pp. 159–180.

[29] P. Boniecki et al. “Neural identification of selected apple pests”. en. In: *Computers and Electronics in Agriculture* 110 (Jan. 2015), pp. 9–16. (Visited on 02/04/2016).

[30] Davide Boscaini et al. “Learning shape correspondence with anisotropic convolutional neural networks”. In: *CoRR* abs/1605.06437 (2016). arXiv: 1605.06437. URL: http://arxiv.org/abs/1605.06437.

[31] Sébastien Bougleux et al. “Graph edit distance as a quadratic assignment problem”. In: *Pattern Recognition Letters* 87 (2017), pp. 38–46.

[32] Michael M. Bronstein et al. “Geometric deep learning: going beyond Euclidean data”. In: *CoRR* abs/1611.08097 (2016). arXiv: 1611.08097. URL: http://arxiv.org/abs/1611.08097.

[33] D. C. Brown. “Close-range camera calibration”. In: *Photogrammetric Eng.* 37.8 (1971), pp. 855–866.

[34] Joan Bruna et al. “Spectral Networks and Locally Connected Networks on Graphs”. In: *CoRR* abs/1312.6203 (2013). arXiv: 1312.6203. URL: http://arxiv.org/abs/1312.6203.

[35] Wolfgang Büchs. “Biodiversity and agri-environmental indicators—general scopes and skills with special reference to the habitat level”. In: *Agriculture, Ecosystems & Environment* 98.1 (2003), pp. 35–78.

[36] H Bunke and G Allermann. “Inexact Graph Matching for Structural Pattern Recognition”. In: *Pattern Recogn. Lett.* 1.4 (1983), pp. 245–253.

[37] Horst Bunke. “On a relation between graph edit distance and maximum common subgraph”. In: *Pattern Recogn. Lett.* 18.8 (1997), pp. 689–694.

[38] Horst Bunke and Kaspar Riesen. “Graph Classification on Dissimilarity Space Embedding”. In: *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*. Ed. by Niels da Vitoria Lobo et al. Vol. 5342. Lecture Notes in Computer Science. Springer, 2008, p. 2. ISBN: 978-3-540-89688-3.

[39] Horst Bunke and Kaspar Riesen. “Towards the unification of structural and statistical pattern recognition”. In: *Pattern Recogn. Lett.* 33.7 (2012), pp. 811–825.

[40] Horst Bunke and Kim Shearer. “A Graph Distance Metric Based on the Maximal Common Subgraph”. In: *Pattern Recogn. Lett.* 19.3-4 (1998), pp. 255–259.

[41] Horst Bunke et al. “Matching of Hypergraphs — Algorithms, Applications, and Experiments”. English. In: *Applied Pattern Recognition*. Ed. by Horst Bunke, Abraham Kandel, and Mark Last. Vol. 91. Studies in Computational Intelligence. Springer, 2008, pp. 131–154. ISBN: 978-3-540-76830-2.

[42] Hans-Jürgen Busse, Ewald BM Denner, and Werner Lubitz. “Classification and identification of bacteria: current approaches to an old problem. Overview of methods used in bacterial systematics”. In: *Journal of biotechnology* 47.1 (1996), pp. 3–38.

[43] T. S. Caetano et al. “Learning Graph Matching”. In: *IEEE Trans. on Pattern Anal. Mach. Intell.* 31.6 (2009), pp. 1048–1058.

[44] Tibério S. Caetano et al. “Learning Graph Matching”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 31.6 (2009), pp. 1048–1058.

[45] Bilson Jake Libres Campana and Eamonn J. Keogh. “A compression-based distance measure for texture”. In: *Statistical Analysis and Data Mining* 3.6 (2010), pp. 381–398.

[46] Gail A Carpenter, Stephen Grossberg, and John H Reynolds. “ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network”. In: *Neural networks* 4.5 (1991), pp. 565–588.

[47] Chad Carson et al. “Blobworld: A system for region-based image indexing and retrieval”. In: *International Conference on Advances in Visual Information Systems*. Springer. 1999, pp. 509–517.

[48] Kai Chen and Qiang Huo. “Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering”. In: *2016 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2016, pp. 5880–5884.

[49] Yanping Chen et al. “Flying insect classification with inexpensive sensors”. In: *Journal of insect behavior* 27.5 (2014), pp. 657–677.

[50] Yudong Chen, S. Sanghavi, and Huan Xu. “Improved Graph Clustering”. In: *Information Theory, IEEE Transactions on* 60.10 (2014), pp. 6440–6455.

[51] Minsu Cho, Karteek Alahari, and Jean Ponce. “Learning Graphs to Match”. In: *IEEE International Conference on Computer Vision, ICCV 2013*. 2013, pp. 25–32.

[52] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. "Reweighted Random Walks for Graph Matching". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 492–505. ISBN: 978-3-642-15555-0.

[53] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[54] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. "Learning a similarity metric discriminatively, with application to face verification". In: *CVPR (1)*. 2005, pp. 539–546.

[55] Anna Choromanska et al. "The loss surfaces of multilayer networks". In: *Artificial Intelligence and Statistics*. 2015, pp. 192–204.

[56] Dan C Cireşan, Ueli Meier, and Jürgen Schmidhuber. "Transfer learning for Latin and Chinese characters with deep neural networks". In: *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE. 2012, pp. 1–6.

[57] D. Claus and A. W. Fitzgibbon. "A Rational Function Lens Distortion Model for General Cameras". In: *Proc. CVPR*. 2005, pp. 213–219.

[58] D. Conte et al. "Thirty Years Of Graph Matching In Pattern Recognition". In: *International Journal of Pattern Recogn. and Artif. Intell.* 18.3 (2004).

[59] Xavier Cortés and Francesc Serratosa. "Learning Graph Matching Substitution Weights Based on the Ground Truth Node Correspondence". In: *IJPRAI* 30.2 (2016).

[60] Xavier Cortés and Francesc Serratosa. "Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences". In: *Pattern Recogn. Lett.* 56 (2015), pp. 22–29.

[61] Belén Cotes et al. "Coccinellid morphospecies as an alternative method for differentiating management regimes in olive orchards". In: *ecological indicators* 9.3 (2009), pp. 548–555.

[62] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. "Balanced Graph Matching". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 313–320. URL: http://papers.nips.cc/paper/2960-balanced-graph-matching.pdf.

[63] Peter Cranston and Terry Hillman. "Rapid assessment of biodiversity using'biological diversity technicians'." In: *Australian Biologist* 5 (1992), pp. 144–154.

[64] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[65] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[66] Mostafa Darwiche et al. "Graph edit distance: Accuracy of local branching from an application point of view". In: *Pattern Recognition Letters* (2018, in press). ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.03.033. URL: http://www.sciencedirect.com/science/article/pii/S0167865518301119.

[67] Ingrid Daubechies. "The wavelet transform, time-frequency localization and signal analysis". In: *IEEE Trans. Information Theory* 36.5 (1990), pp. 961–1005.

[68] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *CoRR* abs/1606.09375 (2016). arXiv: 1606.09375. URL: http://arxiv.org/abs/1606.09375.

[69] Natalia Larios Delgado. "Local-Feature Generic Object Recognition with Application to Insect-Species Identification". PhD thesis. University of Washington, 2010. (Visited on 02/04/2016).

[70] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255. (Visited on 02/04/2016).

[71] José GB Derraik et al. “Arthropod morphospecies versus taxonomic species: a case study with Araneae, Coleoptera, and Lepidoptera”. In: *Conservation Biology* 16.4 (2002), pp. 1015–1023.

[72] F. Devernay and O. Faugeras. “Straight lines have to be straight”. In: *MVA* 13 (2001), pp. 14–24.

[73] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. “Weighted graph cuts without eigenvectors a multilevel approach”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.11 (2007), pp. 1944–1957.

[74] Peter J. Dickinson et al. “Matching graphs with unique node labels”. In: *Pattern Anal. Appl.* 7.3 (2004), pp. 243–254.

[75] Christopher H Dietrich, Ted H Emigh, and Lewis L Deitz. “Morphometric discrimination among females of sibling species of Aconophorini (Homoptera: Membracidae)”. In: *Systematic entomology* 16.3 (1991), pp. 311–318.

[76] Christopher H Dietrich and Christopher D Pooley. “Automated identification of leafhoppers (Homoptera: Cicadellidae: Draeculacephala Ball)”. In: *Annals of the Entomological Society of America* 87.4 (1994), pp. 412–423.

[77] Hristo N. Djidjev and Melih Onus. “Scalable and Accurate Graph Clustering and Community Structure Detection”. In: *Parallel and Distributed Systems, IEEE Trans. on* 24.5 (2013), pp. 1022–1029.

[78] MT Do, JM Harp, and KC Norris. “A test of a pattern recognition system for identification of spiders”. In: *Bulletin of Entomological Research* 89.03 (1999), pp. 217–224.

[79] Philippe Dosch and Ernest Valveny. “Report on the Second Symbol Recognition Contest”. In: *Graphics Recognition. Ten Years Review and Future Perspectives*. Vol. 3926. Lecture Notes in Computer Science. 2006, pp. 381–397.

[80] Philippe Dosch and Ernest Valveny. “Report on the Second Symbol Recognition Contest”. In: *Graphics Recognition. Ten Years Review and Future Perspectives*. Vol. 3926. Lecture Notes in Computer Science. 2006, pp. 381–397.

[81] Ji-Xiang Du, Xiao-Feng Wang, and Guo-Jun Zhang. “Leaf shape based plant species recognition”. en. In: *Applied Mathematics and Computation* 185.2 (Feb. 2007), pp. 883–893. ISSN: 00963003. DOI: 10.1016/j.amc.2006.07.072. (Visited on 02/04/2016).

[82] Peter Duelli and Martin K Obrist. “Biodiversity indicators: the choice of values and measures”. In: *Agriculture, ecosystems & environment* 98.1 (2003), pp. 87–98.

[83] V. Dumoulin and F. Visin. “A guide to convolution arithmetic for deep learning”. In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.07285 [stat.ML].

[84] Scott A. Elias. *Quaternary insects and and their environments*. Smithsonian Institution Press, 1994.

[85] Ryan Elwell and Robi Polikar. “Incremental learning in nonstationary environments with controlled forgetting”. In: *2009 International Joint Conference on Neural Networks*. Citeseer. 2009, pp. 771–778.

[86] P. Erdős and A Rényi. “On the Evolution of Random Graphs”. In: *Publication of the mathematical institute of the Hungarian academy of sciences*. 1960, pp. 17–61.

[87] D. Erhan et al. “Scalable Object Detection Using Deep Neural Networks”. In: *CVPR*. 2014, pp. 2155–2162.

[88] Francisco Escolano and Mario Vento, eds. *Graph-Based Representations in Pattern Recognition, 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007, Proceedings*. Vol. 4538. Lecture Notes in Computer Science. Springer, 2007.

[89] Fina Faithpraise et al. "Automatic plant pest detection and recognition using k-means clustering algorithm and correspondence filters". In: *International Journal of Advanced Biotechnology and Research* 4.2 (2013), pp. 189–199. (Visited on 02/04/2016).

[90] Stefan Fankhauser et al. "Suboptimal Graph Isomorphism using bipartite Matching". In: *IJPRAI* 26.6 (2012).

[91] Colin Favret and Jeffrey M. Sieracki. "Machine vision automated species identification scaled towards production levels: Machine vision ID". en. In: *Systematic Entomology* 41.1 (Jan. 2016), pp. 133–143. ISSN: 03076970. DOI: 10.1111/syen.12146. (Visited on 02/04/2016).

[92] Li Fei Fei, Justin Johnson, and Serena Yeung. *CS231n: Convolutional Neural Networks for Visual Recognition*. http://cs231n.stanford.edu/. 2018.

[93] Linan Feng, Bir Bhanu, and John Heraty. "A software system for automated identification and retrieval of moth images based on wing attributes". en. In: *Pattern Recognition* 51 (Mar. 2016), pp. 225–241. ISSN: 00313203. (Visited on 02/04/2016).

[94] Mirtha-Lina Fernández and Gabriel Valiente. "A graph distance metric combining maximum common subgraph and minimum common supergraph". In: *Pattern Recogn. Lett.* 22.6–7 (2001), pp. 753–758.

[95] AT Finnamore. "The advantages of using arthropods in ecosystem management". In: *Biological Survey of Canada (Terrestrial Arthropods) for Canadian Museum of Nature and Entomological Society of Canada, Ottawa* (1996).

[96] A. Fischer and H. Bunke. "Character prototype selection for handwriting recognition in historical documents". In: *Signal Processing Conference, 2011 19th European*. 2011, pp. 1435–1439.

[97] Andreas Fischer et al. "A Hausdorff Heuristic for Efficient Computation of Graph Edit Distance". In: *Joint IAPR Int. Workshop, S+SSPR 2014 Proceedings*. Ed. by Pasi Fränti et al. Vol. 8621. Lecture Notes in Computer Science. Springer, 2014, pp. 83–92. ISBN: 978-3-662-44414-6.

[98] Andreas Fischer et al. "Approximation of graph edit distance based on Hausdorff matching". In: *Pattern Recogn.* 48.2 (2015), pp. 331–343.

[99] Joern Fischer and David B Lindenmayer. "Landscape modification and habitat fragmentation: a synthesis". In: *Global Ecology and Biogeography* 16.3 (2007), pp. 265–280.

[100] A. W. Fitzgibbon. "Simultaneous Linear Estimation of Multiple View Geometry and Lens Distortion". In: *Proc. CVPR*. 2001.

[101] Pasquale Foggia and Mario Vento. "Graph Embedding for Pattern Recognition". In: *Recognizing Patterns in Signals, Speech, Images and Videos*. Vol. 6388. Lecture Notes in Computer Science. 2010, pp. 75–82.

[102] David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, eds. *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II*. Vol. 5303. Lecture Notes in Computer Science. Springer, 2008. ISBN: 978-3-540-88685-3.

[103] N.A. Fox et al. "Robust Biometric Person Identification Using Automatic Classifier Fusion of Speech, Mouth, and Face Experts". In: *Multimedia, IEEE Transactions on* 9.4 (2007), pp. 701–714.

[104] Tiago Mauricio Francoy et al. “Identification of Africanized honey bees through wing morphometrics: two fast and efficient procedures”. In: *Apidologie* 39.5 (Sept. 2008), pp. 488–494. ISSN: 0044-8435, 1297-9678. (Visited on 02/04/2016).

[105] Pasi Fränti et al., eds. *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*. Vol. 8621. Lecture Notes in Computer Science. Springer, 2014. ISBN: 978-3-662-44414-6.

[106] Paolo Frasconi, Kristian Kersting, and Koji Tsuda, eds. *Mining and Learning with Graphs, MLG 2007, Firence, Italy, August 1-3, 2007, Proceedings*. 2007.

[107] LiMin Fu, Hui-Huang Hsu, and Jose C Principe. “Incremental backpropagation learning networks”. In: *IEEE Transactions on Neural Networks* 7.3 (1996), pp. 757–761.

[108] Xinbo Gao et al. “A Survey of Graph Edit Distance”. In: *Pattern Anal. Appl.* 13.1 (2010), pp. 113–129.

[109] Xinbo Gao et al. “Image categorization: Graph edit distance+edge direction histogram”. In: *Pattern Recogn.* 41.10 (2008), pp. 3179–3191.

[110] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. NY, USA: W. H. Freeman & Co., 1979. ISBN: 0716710447.

[111] Thomas Gärtner, JohnW. Lloyd, and PeterA. Flach. “Kernels for Structured Data”. English. In: *Inductive Logic Programming*. Ed. by Stan Matwin and Claude Sammut. Vol. 2583. Lecture Notes in Computer Science. Springer, 2003, pp. 66–83. ISBN: 978-3-540-00567-4.

[112] Habib Gassoumi, Nadipuram R. Prasad, and John J. Ellington. “Neural network-based approach for insect classification in cotton ecosystems”. In: *International Conference on Intelligent Technologies (InTech 2000), Bangkok, Thailand*. Vol. 7. 2000. (Visited on 02/04/2016).

[113] K. J. Gaston and M. A. O’Neill. “Automated species identification: why not?” en. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 359.1444 (Apr. 2004), pp. 655–667. ISSN: 0962-8436, 1471-2970. (Visited on 02/04/2016).

[114] Benoit Gaüzere, Luc Brun, and Didier Villemin. “Two new graphs kernels in chemoinformatics”. In: *Pattern Recogn. Lett.* 33.15 (2012), pp. 2038–2047.

[115] PW Geier and LR Clark. “An ecological approach to pest control”. In: *Proceedings of the eighth technical meeting. International Union of Conservation of Nature and Natural Resources, Warsaw*. 1960, pp. 10–18.

[116] Heloise Gibb and Dieter F Hochuli. “Habitat fragmentation in an urban environment: large and small fragments support different arthropod assemblages”. In: *Biological conservation* 106.1 (2002), pp. 91–100.

[117] Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[118] Ross B. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[119] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323.

[120] S. Gold and A. Rangarajan. “A graduated assignment algorithm for graph matching”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.4 (1996), pp. 377–388.

[121] Steven Gold and Anand Rangarajan. “A Graduated Assignment Algorithm for Graph Matching”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 18.4 (1996), pp. 377–388.

[122] Lee Gregory and Josef Kittler. "Using Graph Search Techniques for Contextual Colour Retrieval". English. In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Terry Caelli et al. Vol. 2396. Lecture Notes in Computer Science. Springer, 2002, pp. 186–194.

[123] Ehud Gudes, Solomon Eyal Shimony, and Natalia Vanetik. "Discovering Frequent Graph Patterns Using Disjoint Paths". In: *IEEE Trans. on Knowl. and Data Eng.* 18.11 (2006), pp. 1441–1456.

[124] Lin Guo, De-Shuang Huang, and Wenbo Zhao. "The optimization of radial basis probabilistic neural networks based on genetic algorithms". In: *International Joint Conference on Neural Networks*. Vol. 4. IEEE. 2003, pp. 3213–3217.

[125] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira. "Writer-independent Feature Learning for Offline Signature Verification using Deep Convolutional Neural Networks". In: *CoRR* abs/1604.00974 (2016).

[126] Mark Hall et al. "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.

[127] P. M. Hammond. "Species inventory". In: *Global biodiversity*. Ed. by Groombridge B. London: Chapman and Hall, 1992, pp. 17–39.

[128] P.E. Hart, N.J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *Systems Science and Cybernetics, IEEE Trans. on* 4.2 (1968), pp. 100–107.

[129] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[130] M. Havaei et al. "Brain Tumor Segmentation with Deep Neural Networks". In: *CoRR* abs/1505.03540 (2015).

[131] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[132] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep Convolutional Networks on Graph-Structured Data". In: *CoRR* abs/1506.05163 (2015). arXiv: 1506.05163. URL: http://arxiv.org/abs/1506.05163.

[133] Pierre Héroux, Pierre Le Bodic, and Sébastien Adam. "Graphics Recognition. Current Trends and Challenges: 10th Int. Workshop, GREC 2013, Revised Selected Papers". In: ed. by Bart Lamiroy and Jean-Marc Ogier. 2014. Chap. Datasets for the Evaluation of Substitution-Tolerant Subgraph Isomorphism, pp. 240–251.

[134] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*. Vol. 49. NBS, 1952.

[135] Colin de la Higuera et al. "Polynomial algorithms for open plane graph and subgraph isomorphisms". In: *Theor. Comput. Sci.* 498 (2013), pp. 76–99.

[136] Isidore Isaac Hirschman and David V Widder. *The convolution transform*. Courier Corporation, 2012.

[137] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: http://www.sciencedirect.com/science/article/pii/089360809190009T.

[138] Gary B. Huang and Viren Jain. "Deep and Wide Multiscale Recursive Networks for Robust Image Labeling." In: *CoRR* abs/1310.0354 (2013).

[139] *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. IEEE Computer Society, 2013. ISBN: 978-1-4799-2839-2. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6750807.

[140] Chuntao Jiang, Frans Coenen, and Michele Zito. "A survey of frequent subgraph mining algorithms". In: *The Knowledge Engineering Review* 28 (01 2013), pp. 75–105.

[141] Xiaoyi Jiang and Horst Bunke. "Optimal quadratic-time isomorphism of ordered graphs". In: *Pattern Recogn.* 32.7 (1999), pp. 1273–1283.

[142] Xiaoyi Jiang, Miquel Ferrer, and Andrea Torsello, eds. *Graph-Based Representations in Pattern Recognition - 8th IAPR-TC-15 International Workshop, GbRPR 2011, Münster, Germany, May 18-20, 2011. Proceedings*. Vol. 6658. Lecture Notes in Computer Science. Springer, 2011.

[143] Utsugi Jinbo, Toshihide Kato, and Motomi Ito. "Current progress in DNA barcoding and future implications for entomology". In: *Entomological Science* 14.2 (2011), pp. 107–124.

[144] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. "Cutting-plane Training of Structural SVMs". In: *Mach. Learn.* 77.1 (Oct. 2009), pp. 27–59. ISSN: 0885-6125.

[145] Alexis Joly et al. "Interactive plant identification based on social image data". en. In: *Ecological Informatics* 23 (Sept. 2014), pp. 22–34. ISSN: 15749541. DOI: 10.1016/j.ecoinf.2013.07.006. (Visited on 02/04/2016).

[146] Richard Jonker et al. "Automated identification and characterisation of microbial populations using flow cytometry: the AIMS project". In: *Scientia Marina* 64.2 (2000), pp. 225–234.

[147] Derek Justice and Alfred Hero. "A Binary Linear Programming Formulation of the Graph Edit Distance". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.8 (2006), pp. 1200–1214.

[148] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *CoRR* abs/1609.04836 (2016). arXiv: 1609.04836. URL: http://arxiv.org/abs/1609.04836.

[149] Diana N Kimberling, James R Karr, and Leska S Fore. "Measuring human disturbance using terrestrial invertebrates in the shrub–steppe of eastern Washington (USA)". In: *Ecological indicators* 1.2 (2001), pp. 63–81.

[150] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[151] RL Kitching et al. "Moth assemblages as indicators of environmental quality in remnants of upland Australian rain forest". In: *Journal of Applied Ecology* 37.2 (2000), pp. 284–297.

[152] Orestis Kostakis. "Classy: fast clustering streams of call-graphs". In: *Data Mining and Knowledge Discovery* 28.5 (2014), pp. 1554–1585.

[153] Stefan Kramer and Luc De Raedt. "Feature Construction with Version Spaces for Biochemical Applications". In: *Proc. of the 18th Int. Conf. on Machine Learning*. 2001, pp. 258–265.

[154] Claire Kremen. "Assessing the indicator properties of species assemblages for natural areas monitoring". In: *Ecological applications* 2.2 (1992), pp. 203–217.

[155] Colwell Kremen et al. "Terrestrial arthropod assemblages: their use in conservation planning". In: *Conservation biology* 7.4 (1993), pp. 796–808.

[156] A. Krizhevsky, I. Sutskever, and G.E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS 25*. Ed. by F. Pereira et al. 2012, pp. 1097–1105.

[157] Matthew Lai. "Deep Learning for Medical Image Segmentation". In: *CoRR* abs/1505.02000 (2015).

[158] Kevin Lambeets et al. "Integrating environmental conditions and functional life-history traits for riparian arthropod conservation planning". In: *Biological Conservation* 142.3 (2009), pp. 625–637.

[159] Natalia Larios et al. "Automated insect identification through concatenated histograms of local appearance features: feature vector generation and region detection for deformable objects". en. In: *Machine Vision and Applications* 19.2 (Mar. 2008), pp. 105–123. ISSN: 0932-8092, 1432-1769. DOI: 10.1007/s00138-007-0086-y. (Visited on 02/04/2016).

[160] Hugo Larochelle et al. "An empirical evaluation of deep architectures on problems with many factors of variation". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 473–480.

[161] John H Lawton et al. "Biodiversity inventories, indicator taxa and effects of habitat modification in tropical forest". In: *Nature* 391.6662 (1998), pp. 72–76.

[162] Pierre Le Bodic et al. "An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings". In: *Pattern Recogn.* 45.12 (2012), pp. 4214–4224.

[163] Yann Le Cun. *Some folks still seem confused about what deep learning is*. 2019. URL: https://www.facebook.com/722677142/posts/10156463919392143/ (visited on 01/14/2020).

[164] Yann LeCun. "What's Wrong With Deep Learning?" Slide #3 — Available at: http://www.pamitc.org/cvpr15/files/lecun-20150610-cvpr-keynote.pdf. June 2015.

[165] Yann A LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[166] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[167] Yann LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade - Second Edition*. Ed. by Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller. Vol. 7700. Lecture Notes in Computer Science. Springer, 2012, pp. 9–48. ISBN: 978-3-642-35288-1.

[168] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[169] J. Lee, M. Cho, and K. M. Lee. "A Graph Matching Algorithm Using Data-Driven Markov Chain Monte Carlo Sampling". In: *2010 20th International Conference on Pattern Recognition*. Aug. 2010, pp. 2816–2819. DOI: 10.1109/ICPR.2010.690.

[170] M. Leordeanu and M. Hebert. "A spectral technique for correspondence problems using pairwise constraints". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. Oct. 2005, 1482–1489 Vol. 2. DOI: 10.1109/ICCV.2005.20.

[171] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. "An Integer Projected Fixed Point Method for Graph Matching and MAP Inference". In: *Proceedings Neural Information Processing Systems*. 2009, pp. 1114–1122.

[172] Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. "Unsupervised Learning for Graph Matching". In: *International Journal of Computer Vision* 96.1 (2012), pp. 28–45.

[173] Lee Kien Leow et al. "Automated identification of copepods using digital image processing and artificial neural network". In: *BMC bioinformatics* 16.Suppl 18 (2015), S4. (Visited on 02/04/2016).

[174] Julien Lerouge et al. "New binary linear programming formulation to compute the graph edit distance". In: *Pattern Recognition* 72 (2017), pp. 254–265. DOI: 10.1016/j.patcog.2017.07.029. URL: https://doi.org/10.1016/j.patcog.2017.07.029.

[175] Jih-Pai Lin et al. “A Parvancorina-like arthropod from the Cambrian of South China”. In: *Historical Biology* 18.1 (2006), pp. 33–45.

[176] Min Lin, Qiang Chen, and Shuicheng Yan. “Network In Network”. In: *CoRR* abs/1312.4400 (2013).

[177] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *CoRR* abs/1512.02325 (2015). arXiv: 1512.02325. URL: http://arxiv.org/abs/1512.02325.

[178] Zhiyong Liu and Hong Qiao. “GNCCP - Graduated NonConvexityand Concavity Procedure”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 36.6 (2014), pp. 1258–1267.

[179] Zhiyong Liu and Hong Qiao. “GNCCP - Graduated NonConvexityand Concavity Procedure”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 36 (2014), pp. 1258–1267.

[180] Lorenzo Livi and Antonello Rizzi. “The graph matching problem”. In: *Pattern Analysis and Applications* 16.3 (2013), pp. 253–283.

[181] Daniel P. Lopresti and Gordon T. Wilfong. “A fast technique for comparing graph representations with applications to performance evaluation.” In: *IJDAR* 6.4 (2003), pp. 219–229.

[182] David G Lowe. “Object recognition from local scale-invariant features”. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. IEEE. 1999, pp. 1150–1157.

[183] Miguel Angel Lozano and Francisco Escolano. “Graph matching and clustering using kernel attributes”. In: *Neurocomputing* 113 (2013), pp. 177–194.

[184] An Lu et al. “Insect species recognition using discriminative local soft coding”. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 1221–1224. (Visited on 02/04/2016).

[185] An Lu et al. “Insect Species Recognition using Sparse Representation”. en. In: *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2010, pp. 108.1–108.10. (Visited on 02/04/2016).

[186] Eugene M. Luks. “Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time”. In: *J. Comput. Syst. Sci.* 25.1 (1982), pp. 42–65.

[187] B Luo, R C Wilson, and E R Hancock. “Spectral embedding of graphs”. In: *Pattern Recogn.* 36.10 (2003), pp. 2213–2230.

[188] Muhammad Muzzamil Luqman et al. “Fuzzy multilevel graph embedding”. In: *Pattern Recognition* 46.2 (2013), pp. 551–565.

[189] David A. Lytle et al. “Automated processing and identification of benthic invertebrate samples”. en. In: *Journal of the North American Benthological Society* 29.3 (Sept. 2010), pp. 867–874. ISSN: 0887-3593, 1937-237X. DOI: 10.1899/09-080.1. (Visited on 02/04/2016).

[190] Norman MacLeod and Systematics Association, eds. *Automated taxon identification in systematics: theory, approaches and applications*. The Systematics Association special volume series 74. Boca Raton: CRC Press, 2007. ISBN: 978-0-8493-8205-5.

[195] Jonathan Masci et al. “Geodesic convolutional neural networks on riemannian manifolds”. In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45.

[196] Michael Mayo and Anna T. Watson. “Automatic species identification of live moths”. en. In: *Knowledge-Based Systems* 20.2 (Mar. 2007), pp. 195–202. (Visited on 02/04/2016).

[197] Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[198] N. David Mermin. "What's wrong with these equations?" In: *Physics Today* (Oct. 1989).

[199] Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, eds. *Neural Networks: Tricks of the Trade - Second Edition*. Vol. 7700. Lecture Notes in Computer Science. Springer, 2012. ISBN: 978-3-642-35288-1.

[200] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *CoRR* abs/1611.08402 (2016). arXiv: 1611.08402. URL: http://arxiv.org/abs/1611.08402.

[201] Carlos Francisco Moreno-García, Xavier Cortés, and Francesc Serratosa. "A Graph Repository for Learning Error-Tolerant Graph Matching". In: *Structural, Syntactic, and Statistical Pattern Recognition S+SSPR 2016*. 2016, pp. 519–529.

[202] Eric N. Mortensen, Hongli Deng, and Linda Shapiro. "A SIFT descriptor with global context". In: *Proceedings of CVPR 2005*. Vol. 1. 2005, pp. 184–190. (Visited on 02/04/2016).

[203] Eric N. Mortensen et al. "Pattern recognition for ecological science and environmental monitoring: An initial report". In: *Automated Taxon Identification in Systematics* (2007), pp. 189–206. (Visited on 02/04/2016).

[204] J. Munkres. "Algorithms for the Assignment and Transportation Problems". In: *Journal of the Society of Industrial and Applied Mathematics* 5.1 (1957), pp. 32–38.

[205] James R. Munkres. "Algorithms for the Assignment and Transportation Problems". In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957), pp. 32–38.

[206] Richard Myers, Richard C. Wilson, and Edwin R. Hancock. "Bayesian Graph Edit Distance". In: *IEEE Trans. on Pattern Anal. and Mach. Intell.* 22.6 (2000), pp. 628–635.

[207] Michel Neuhaus and Horst Bunke. "Automatic learning of cost functions for graph edit distance". In: *Inf. Sci.* 177.1 (2007), pp. 239–247.

[208] Michel Neuhaus and Horst Bunke. *Bridging the Gap between Graph Edit Distance and Kernel Machines*. Vol. 68. Series in Machine Perception and Artificial Intelligence. WorldScientific, 2007. ISBN: 978-981-270-817-5.

[209] Michel Neuhaus and Horst Bunke. "Self-organizing maps for learning the edit costs in graph matching". In: *IEEE Trans. Systems, Man, and Cybernetics, Part B* 35.3 (2005), pp. 503–514.

[210] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. "Fast Suboptimal Algorithms for the Computation of Graph Edit Distance". In: *SSPR/SPR*. 2006, pp. 163–172.

[211] Jari Niemelä, David Langor, and John R Spence. "Effects of Clear-Cut Harvesting on Boreal Ground-Beetle Assemblages (Coleoptera: Carabidae) in Western Canada". In: *Conservation biology* 7.3 (1993), pp. 551–561.

[212] MK Obrist and P Duelli. "Rapid biodiversity assessment of arthropods for monitoring average local species richness and related ecosystem services". In: *Biodiversity and conservation* 19.8 (2010), pp. 2201–2220.

[213] Frode. Ødegaard. "How many species of arthropods? Erwin's estimate revised". In: *Biological Journal of the Linnean Society* 71.4 (2000), pp. 583–597.

[214] Ian Oliver and Andrew J Beattie. "Invertebrate morphospecies as surrogates for species: a case study". In: *Conservation Biology* 10.1 (1996), pp. 99–109.

[215] M. A. O'Neill et al. "Daisy: an automated invertebrate identification system using holistic vision techniques". In: *Proceedings of the Inaugural Meeting BioNET-INTERNATIONAL Group for Computer-Aided Taxonomy (BIGCAT)*. 1997, pp. 13–22. (Visited on 02/04/2016).

[216] Mark A. O'Neill. "DAISY: a practical tool for semi-automated species identification". In: *Automated Taxon Identification in Systematics: Theory, Approaches and Applications*. Ed. by N. MacLeod. Boca Raton, FL: CRC Press, 2008. Chap. 9, pp. 131–152.

[217] Deepti Pachauri, Maxwell D. Collins, and Vikas Singh. "Incorporating Domain Knowledge in Matching Problems via Harmonic Analysis". In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL: http://icml.cc/2012/papers/639.pdf.

[218] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347.

[219] Naveen Pandey, Satyanarayan Krishna, and Shanu Sharma. "Automatic Seed Classification by Shape and Color Features using Machine Vision Technology". In: *International Journal of Computer Applications Technology and Research* 2.2 (2013), pp. 208–213. (Visited on 02/04/2016).

[220] Panos M. Pardalos, Kowta A. Murthy, and Terry P. Harrison. "A computational comparison of local search heuristics for solving quadratic assignment problems". In: 1993.

[221] Jörg Perner and Steffen Malt. "Assessment of changing agricultural land use: response of vegetation, ground-dwelling spiders and beetles to the conversion of arable land into grassland". In: *Agriculture, Ecosystems & Environment* 98.1 (2003), pp. 169–181.

[222] D. Pisani et al. "The colonization of land by animals: molecular phylogeny and divergence times among arthropods". In: *BMC Biology* 2 (2004), p. 1.

[223] Marc Pollet and Patrick Grootaert. "An estimation of the natural value of dune habitats using Empidoidea (Diptera)". In: *Biodiversity & Conservation* 5.7 (1996), pp. 859–880.

[224] David I Poole, Randy G Goebel, and Alan K Mackworth. *Computational intelligence*. Oxford University Press New York, 1998.

[225] Arik Poznanski and Lior Wolf. "CNN-N-Gram for Handwriting Word Recognition". In: *CVPR*. 2016, pp. 2305–2314.

[226] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[227] *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[228] Yao Qing et al. "An insect imaging system to automate rice light-trap pest identification". In: *Journal of Integrative Agriculture* 11.6 (2012), pp. 978–985.

[229] Zhu Le-Qing. "Insect sound recognition based on mfcc and pnn". In: *Multimedia and Signal Processing (CMSP), 2011 International Conference on*. Vol. 2. IEEE. 2011, pp. 42–46.

[230] Romain Raveaux, Jean-Christophe Burie, and Jean-Marc Ogier. "A graph matching method and a graph matching distance based on subgraph assignments". In: *Pattern Recogn. Lett.* 31.5 (2010), pp. 394–406.

[231] Romain Raveaux, Jean-Christophe Burie, and Jean-Marc Ogier. "Structured representations in a content based image retrieval context". In: *J. Visual Communication and Image Representation* 24.8 (2013), pp. 1252–1268.

[233] Romain Raveaux et al. "Learning graph prototypes for shape recognition". In: *Computer Vision and Image Understanding* 115.7 (2011), pp. 905–918.

[234] Dan Raviv, Ron Kimmel, and Alfred M. Bruckstein. "Graph Isomorphisms and Automorphisms via Spectral Signatures". In: *IEEE Trans. on Pattern Anal. and Mach. Intell.* 35.8 (2013), pp. 1985–1993.

[235] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[236] Peng Ren, R.C. Wilson, and E.R. Hancock. “Graph Characterization via Ihara Coefficients”. In: *Neural Networks, IEEE Trans. on* 22.2 (2011), pp. 233–245.

[237] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[238] S. Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *NIPS 28* (2015). Ed. by C. Cortes et al., pp. 91–99.

[239] K. Riesen and H. Bunke. “Graph Classification by Means of Lipschitz Embedding”. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39.6 (2009), pp. 1472–1483.

[240] Kaspar Riesen. *Structural Pattern Recognition with Graph Edit Distance - Approximation Algorithms and Applications*. Advances in Comp. Vis. and Pattern Recogn. Springer, 2015. ISBN: 978-3-319-27251-1.

[241] Kaspar Riesen. *Structural Pattern Recognition with Graph Edit Distance - Approximation Algorithms and Applications*. Advances in Computer Vision and Pattern Recognition. Springer, 2015.

[242] Kaspar Riesen and Horst Bunke. “Approximate graph edit distance computation by means of bipartite graph matching”. In: *Image Vision Comput.* 27.7 (2009), pp. 950–959.

[243] Kaspar Riesen and Horst Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2010. ISBN: 9789814304719, 9814304719.

[244] Kaspar Riesen and Horst Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*. Vol. 77. Series in Machine Perception and Artificial Intelligence. WorldScientific, 2010. ISBN: 978-981-4304-71-9.

[245] Kaspar Riesen and Horst Bunke. “IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning”. In: *SSPR*. Vol. 5342. Lecture Notes in Computer Science. 2008, pp. 287–297.

[246] Kaspar Riesen and Horst Bunke. “IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning”. In: *SSPR, Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*. Ed. by Niels da Vitoria Lobo et al. Vol. 5342. Lecture Notes in Computer Science. Springer, 2008, pp. 287–297. ISBN: 978-3-540-89688-3.

[247] Kaspar Riesen and Horst Bunke. “Improving bipartite graph edit distance approximation using various search strategies”. In: *Pattern Recognition* 48.4 (2015), pp. 1349–1363.

[248] Kaspar Riesen, Stefan Fankhauser, and Horst Bunke. “Speeding Up Graph Edit Distance Computation with a Bipartite Heuristic”. In: *Mining and Learning with Graphs, MLG 2007 Proceedings*. Ed. by Paolo Frasconi, Kristian Kersting, and Koji Tsuda. 2007.

[249] Kaspar Riesen and Miquel Ferrer. “Predicting the correctness of node assignments in bipartite graph matching”. In: *Pattern Recognition Letters* 69 (2016), pp. 8–14.

[250] Kaspar Riesen, Andreas Fischer, and Horst Bunke. “Improving Graph Edit Distance Approximation by Centrality Measures”. In: *Proceeding of the 22nd Int. Conf. on Pattern Recogn.* 2014, pp. 3910–3914.

[251] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. "Bipartite Graph Matching for Computing the Edit Distance of Graphs". In: *Proc. of the 6th IAPR Int. Workshop on Graph-Based Repres. in Pattern Recogn.* 2007, pp. 1–12.

[252] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. "Graph Embedding in Vector Spaces by Means of Prototype Selection". In: *Proc. of the 6th IAPR International Workshop on Graph-based Repr. in Pattern Recogn.* 2007, pp. 383–393.

[253] Antonio Robles-Kelly and Edwin R. Hancock. "Graph Edit Distance from Spectral Seriation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.3 (2005), pp. 365–378.

[254] Antonio Robles-Kelly et al., eds. *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2016, Mérida, Mexico, November 29 - December 2, 2016, Proceedings*. Vol. 10029. Lecture Notes in Computer Science. 2016. ISBN: 978-3-319-49054-0.

[255] Lior Rokach. "Ensemble-based classifiers". In: *Artificial Intelligence Review* 33.1-2 (2010), pp. 1–39.

[256] David M. Rosenberg, H. V. Danks, and Dennis M. Lehmkuhl. "Importance of insects in environmental impact assessment". In: *Environmental Management* 10.6 (1986), pp. 773–783.

[257] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[258] Frank Rosenblatt. "The Perceptron – a perceiving and recognizing automaton." In: *Cornell Aeronautical Laboratory.* (1957),

[259] Holger R. Roth et al. "Detection of Sclerotic Spine Metastases via Random Aggregation of Deep Convolutional Neural Network Classifications". In: *CoRR* abs/1407.5976 (2014).

[260] Volker Roth et al. "Optimal Cluster Preserving Embedding of Nonmetric Proximity Data". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.12 (2003), pp. 1540–1551.

[261] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0.

[262] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[263] K. N. Russell et al. "Introducing SPIDA-Web: Wavelets, Neural Networks and Internet Accessibility in an Image-Based Automated Identification System". In: *Automated Taxon Identification in Systematics: Theory, Approaches and Applications*. Ed. by N. MacLeod. Boca Raton, FL: CRC Press, 2008. Chap. 9, pp. 131–152.

[264] Jessica J Rykken, David E Capen, and Sean P Mahabir. "Ground beetles as indicators of land type diversity in the Green Mountains of Vermont". In: *Conservation Biology* 11.2 (1997), pp. 522–530.

[265] R. Saigal. *Linear Programming: A Modern Integrated Analysis*. Kluwer Academic Publishers, 1995.

[266] Saleh Mufleh Al-Saqer and Ghulam Mubashar Hassan. "Artificial neural networks based red palm weevil (Rynchophorus Ferrugineous, Olivier) recognition system". In: *Am. J. Agric. Biol. Sci* 6 (2011), pp. 356–364.

[267] Matt J. Sarpola et al. "An aquatic insect imaging system to automate insect classification". In: *Transactions of the ASABE* 51.6 (2008), pp. 2217–2225. (Visited on 02/04/2016).

[268] T Sattler et al. "Response of arthropod species richness and functional groups to urban habitat structure and management". In: *Landscape Ecology* 25.6 (2010), pp. 941–954.

[269] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605. URL: https://doi.org/10.1109/TNN.2008.2005605.

[270] Christian Schellewald and Christoph Schnörr. “Probabilistic Subgraph Matching Based on Convex Relaxation”. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Ed. by Anand Rangarajan, Baba Vemuri, and Alan L. Yuille. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 171–186. ISBN: 978-3-540-32098-2.

[271] I. Schomburg et al. “BRENDA, the enzyme database: updates and major new developments”. In: *Nucleic Acids Research* 32.suppl 1 (2004), pp. D431–D433.

[272] S. I. Sergeev. “Improved Lower Bounds for the Quadratic Assignment Problem”. In: *Automation and Remote Control* 65.11 (Nov. 2004), pp. 1733–1746. ISSN: 1608-3032. DOI: 10.1023/B:AURC.0000047888.76717.7a. URL: https://doi.org/10.1023/B:AURC.0000047888.76717.7a.

[273] Francesc Serratosa. “Computation of graph edit distance: Reasoning about optimality and speed-up”. In: *Image Vision Comp.* 40 (2015), pp. 38–48.

[274] Francesc Serratosa. “Fast computation of Bipartite graph matching”. In: *Pattern Recogn. Lett.* 45 (2014), pp. 244–250.

[275] Francesc Serratosa. “Speeding up fast bipartite graph matching through a new cost matrix”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 29.02 (2015), p. 1550010.

[276] L. G. Shapiro and R. M. Haralick. “Structural Descriptions and Inexact Matching”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3.5 (Sept. 1981), pp. 504–519. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1981.4767144.

[277] Jianbo Shi and Jitendra Malik. “Normalized Cuts and Image Segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8 (2000), pp. 888–905.

[278] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[279] Huang Shiguo et al. “Ontology-based insect recognition”. In: *Image Analysis and Signal Processing, 2009. IASP 2009. International Conference on*. IEEE, 2009, pp. 176–178.

[280] N. Sidère, Pierre Héroux, and J.-Y. Ramel. “Vector Representation of Graphs: Application to the Classification of Symbols and Letters”. In: *Proc. of the Int. Conf. on Doc. Anal. and Recogn.* 2009, pp. 681–685.

[281] Felipe Leno da Silva et al. “Evaluating classification and feature selection techniques for honeybee subspecies identification using wing images”. en. In: *Computers and Electronics in Agriculture* 114 (June 2015), pp. 68–77. ISSN: 01681699. DOI: 10.1016/j.compag.2015.03.012. (Visited on 02/04/2016).

[282] Ivan Nunes da Silva et al. “The ADALINE Network and Delta Rule”. In: *Artificial Neural Networks : A Practical Course*. Cham: Springer International Publishing, 2017, pp. 41–54. ISBN: 978-3-319-43162-8. DOI: 10.1007/978-3-319-43162-8_4. URL: https://doi.org/10.1007/978-3-319-43162-8_4.

[283] Nadja K Simons, Wolfgang W Weisser, and Martin M Gossner. “Multi-taxa approach shows consistent shifts in arthropod functional traits along grassland land-use intensity gradient”. In: *Ecology* 97.3 (2016), pp. 754–764.

[284] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.1556.

[285] AlexanderJ. Smola and Risi Kondor. "Kernels and Regularization on Graphs". English. In: *Learning Theory and Kernel Machines*. Ed. by Bernhard Schölkopf and ManfredK. Warmuth. Vol. 2777. Lecture Notes in Computer Science. Springer, 2003, pp. 144–158. ISBN: 978-3-540-40720-1.

[286] LO Solis-Sánchez et al. "Machine vision algorithm for whiteflies (Bemisia tabaci Genn.) scouting under greenhouse environment". In: *Journal of Applied Entomology* 133.7 (2009), pp. 546–552.

[287] Luis O. Solis-Sánchez et al. "Scale invariant feature approach for insect monitoring". en. In: *Computers and Electronics in Agriculture* 75.1 (Jan. 2011), pp. 92–99. (Visited on 02/04/2016).

[288] R. Swaminathan and S. Nayar. "Nonmetric calibration of wide-angle lenses and polycameras". In: *IEEE T-PAMI* 22.10 (2000), pp. 1172–1178.

[289] C. Szegedy, A. Toshev, and D. Erhan. "Deep Neural Networks for Object Detection". In: *NIPS 26*. Ed. by C. J. C. Burges et al. 2013, pp. 2553–2561.

[290] Ville Tirronen et al. "Multiple order gradient feature for macro-invertebrate identification using support vector machines". In: *Adaptive and Natural Computing Algorithms*. Springer, 2009, pp. 489–497. (Visited on 02/04/2016).

[291] Adam Tofilski. "DrawWing, a program for numerical description of insect wings". In: *Journal of Insect Science* 4.1 (2004), p. 17. (Visited on 02/04/2016).

[292] L. Torresani, V. Kolmogorov, and C. Rother. "A Dual Decomposition Approach to Feature Correspondence". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.2 (Feb. 2013), pp. 259–271. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.105.

[293] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. "Feature Correspondence Via Graph Matching: Models and Global Optimization". In: *Computer Vision - ECCV*. Ed. by David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman. Vol. 5303. Lecture Notes in Computer Science. Springer, 2008, pp. 596–609. ISBN: 978-3-540-88685-3.

[294] Wen-hsiang Tsai, Student Member, and King-sun Fu. "Pattern Deformational Model and Bayes Error-Correcting Recognition System". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1979), pp. 745–756.

[295] Y. R. Tsai. "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision". In: *Proc. CVPR*. 1986.

[296] S. Umeyama. "An eigendecomposition approach to weighted graph matching problems". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 10.5 (1988), pp. 695–703.

[297] Shinji Umeyama. "An Eigendecomposition Approach to Weighted Graph Matching Problems". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 10.5 (1988), pp. 695–703.

[298] G. Urban et al. "Multi-modal Brain Tumor Segmentation using Deep Convolutional Neural Networks". In: *MICCAI BraTS Challenge Proceedings*. 2014, pp. 31–35.

[299] M Van Helden et al. "Arthropod biodiversity monitoring using RBA techniques in viticulture." In: *IOBC/WPRS Bulletin* 56 (2010), pp. 117–120.

[300] Nico M Van Straalen. "Evaluation of bioindicator systems derived from soil arthropod communities". In: *Applied Soil Ecology* 9.1 (1998), pp. 429–437.

[301] Niels da Vitoria Lobo et al., eds. *SSPR, Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*. Vol. 5342. Lecture Notes in Computer Science. Springer, 2008. ISBN: 978-3-540-89688-3.

[302] W. D. Wallis et al. "Graph Distances Using Graph Union". In: *Pattern Recogn. Lett.* 22.6-7 (2001), pp. 701–704.

[303] Sheng Wan and Larry E Banta. "Parameter incremental learning algorithm for neural networks". In: *IEEE Transactions on Neural Networks* 17.6 (2006), pp. 1424–1438.

[304] Jiangning Wang et al. "A new automatic identification system of insect images at the order level". en. In: *Knowledge-Based Systems* 33 (Sept. 2012), pp. 102–110. ISSN: 09507051. DOI: 10.1016/j.knosys.2012.03.014. (Visited on 02/04/2016).

[305] Jiangning Wang et al. "The identification of butterfly families using content-based image retrieval". en. In: *Biosystems Engineering* 111.1 (Jan. 2012), pp. 24–32. ISSN: 15375110. DOI: 10.1016/j.biosystemseng.2011.10.003. (Visited on 02/04/2016).

[306] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. "Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs". In: *CoRR* abs/1803.08035 (2018). arXiv: 1803.08035. URL: http://arxiv.org/abs/1803.08035.

[307] Anna T. Watson, Mark A. O'Neill, and Ian J. Kitching. "Automated identification of live moths (Macrolepidoptera) using digital automated identification System (DAISY)". en. In: *Systematics and Biodiversity* 1.3 (Feb. 2004), pp. 287–300. (Visited on 02/04/2016).

[308] P. J. D. Weeks et al. "Automating insect identification: exploring the limitations of a prototype system". In: *Journal of Applied Entomology* 123.1 (1999), pp. 1–8. (Visited on 02/04/2016).

[309] P. J. D. Weeks et al. "Automating the identification of insects: a new solution to an old problem". In: *Bulletin of Entomological Research* 87.02 (1997), pp. 203–211.

[310] P. J. D. Weeks et al. "Species–identification of wasps using principal component associative memories". In: *Image and Vision Computing* 17.12 (1999), pp. 861–866. (Visited on 02/04/2016).

[311] Chenglu Wen and Daniel Guyer. "Image-based orchard insect automated identification and classification method". en. In: *Computers and Electronics in Agriculture* 89 (Nov. 2012), pp. 110–115. ISSN: 01681699. DOI: 10.1016/j.compag.2012.08.008. (Visited on 02/04/2016).

[312] Chenglu Wen, Daniel E. Guyer, and Wei Li. "Local feature-based identification and classification for orchard insects". en. In: *Biosystems Engineering* 104.3 (Nov. 2009), pp. 299–307. (Visited on 02/04/2016).

[313] Chenglu Wen and Qingyuan Zhu. "Dimension reduction analysis in image-based species classification". In: *Intelligent Computing and Intelligent Systems (ICIS)*. Vol. 3. 2010, pp. 731–734. (Visited on 02/04/2016).

[314] Chenglu Wen et al. "Pose estimation-dependent identification method for field moth images using deep learning architecture". en. In: *Biosystems Engineering* 136 (Aug. 2015), pp. 117–128. (Visited on 02/04/2016).

[315] Bernard Widrow and Marcian E Hoff. *Adaptive switching circuits*. Tech. rep. Stanford Univ Ca Stanford Electronics Labs, 1960.

[316] Richard C. Wilson and Edwin R. Hancock. "Structural Matching by Discrete Relaxation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 19.6 (1997), pp. 634–648. ISSN: 0162-8828.

[317] L. A. Wolsey. *Integer programming*. NY, USA: Wiley-Interscience, 1998. ISBN: 0-471-28366-5.

[318] J. Wu et al. "Boosting for Multi-Graph Classification". In: *Cybernetics, IEEE Transactions on* 45.3 (2014), pp. 430–443.

[319] Barend J. van Wyk and Michael A. van Wyk. "A POCS-Based Graph Matching Algorithm". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.11 (2004), pp. 1526–1530. ISSN: 0162-8828.

[320] Bing Xiao et al. “HMM-based graph edit distance for image indexing”. In: *Int. J. Imaging Systems and Technology* 18.2-3 (2008), pp. 209–218.

[321] Chengjun Xie et al. “Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning”. en. In: *Computers and Electronics in Agriculture* 119 (Nov. 2015), pp. 123–132. ISSN: 01681699. DOI: 10.1016/j.compag.2015.10.015. (Visited on 02/04/2016).

[322] Shahrul Nizam Yaakob and Lakhmi Jain. “An insect classification analysis based on shape features using quality threshold ARTMAP and moment invariant”. en. In: *Applied Intelligence* 37.1 (July 2012), pp. 12–30. ISSN: 0924-669X, 1573-7497. (Visited on 02/04/2016).

[323] Hulya Yalcin. “Vision based automatic inspection of insects in pheromone traps”. In: *Agro-Geoinformatics (Agro-geoinformatics), 2015 Fourth International Conference on*. IEEE, 2015, pp. 333–338. (Visited on 02/04/2016).

[324] He-Ping Yang et al. “A tool for developing an automatic insect identification system based on wing outlines”. In: *Scientific Reports* 5 (Aug. 2015), p. 12786. ISSN: 2045-2322. (Visited on 02/04/2016).

[325] Shulin Yang et al. “Unsupervised template learning for fine-grained object recognition”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 3122–3130. (Visited on 02/04/2016).

[326] Dit-Yan Yeung et al., eds. *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006, Proceedings*. Vol. 4109. Lecture Notes in Computer Science. Springer, 2006.

[327] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[328] Jason Yosinski et al. “Understanding Neural Networks Through Deep Visualization”. In: *CoRR* abs/1506.06579 (2015).

[329] Ehsan Zare Borzeshi et al. “Discriminative Prototype Selection Methods for Graph Embedding”. In: *Pattern Recogn.* 46.6 (2013), pp. 1648–1657. ISSN: 0031-3203.

[330] I. Y. Zayas and P. W. Flinn. “Detection of Insects in Bulk Wheat Samples With Machine Vision”. In: *Transactions of the ASAE* 41.3 (1998), pp. 883–888. (Visited on 02/04/2016).

[331] Zhiping Zeng et al. “Comparing Stars: On Approximating Graph Edit Distance”. In: *Proc. of the VLDB Endowment*. Vol. 2. 2009, pp. 25–36.

[332] Qingchen Zhang et al. “A survey on deep learning for big data”. In: *Information Fusion* 42 (2018), pp. 146–157.

[333] Z. Zhang. “On the Epipolar Geometry Between Two Images With Lens Distortion”. In: *Proc. ICPR*. 1996, pp. 407–411.

[334] Le-Qing Zhu and Zhen Zhang. “Auto-classification of insect images based on color histogram and GLCM”. In: *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*. Vol. 6. IEEE, 2010, pp. 2589–2593. (Visited on 02/04/2016).

[335] Le-qing Zhu and Zhen Zhang. “Insect recognition based on integrated region matching and dual tree complex wavelet transform”. en. In: *Journal of Zhejiang University SCIENCE C* 12.1 (Jan. 2011), pp. 44–53. ISSN: 1869-1951, 1869-196X. (Visited on 02/04/2016).

[336] Zhaonian Zou et al. “Mining Frequent Subgraph Patterns from Uncertain Graph Data”. In: *Knowledge and Data Engineering, IEEE Transactions on* 22.9 (2010), pp. 1203–1218.

Chloé Martineau

Apprentissage profond dans l'espace des graphes : application à la reconnaissance d'images d'insectes

Résumé :
Le but de cette thèse est d'étudier la reconnaissance d'insectes comme un problème de reconnaissance des formes basé images. Bien que ce problème ait été étudié en profondeur au long des trois dernières décennies, un aspect reste selon nous toujours à expérimenter à ce jour : les approches profondes (deep learning). À cet effet, la première contribution de cette thèse consiste à déterminer la faisabilité de l'application des réseaux de neurones convolutifs profonds (CNN) au problème de reconnaissance d'images d'insectes. Les limitations majeures ont les suivantes: les images sont très rares et les cardinalités de classes sont hautement déséquilibrées. Pour atténuer ces limitations, le *transfer learning* et la pondération de la fonction de coûts ont été employés. Des méthodes basées graphes sont également proposées et testées. La première consiste en la conception d'un classificateur de graphes de type *perceptron*. Le second travail basé sur les graphes de cette thèse est la définition d'un opérateur de convolution pour construire un modèle de réseaux de neurones convolutifs s'appliquant sur les graphes (GCNN.) Le dernier chapitre de la thèse s'applique à utiliser les méthodes mentionnées précédemment à des problèmes de reconnaissance d'images d'insectes. Deux bases d'images sont ici proposées. Là première est constituée d'images prises en laboratoire sur arrière-plan constant. La seconde base est issue de la base ImageNet. Cette base est composée d'images prises en contexte naturel. Les CNN entrainés avec transfer learning sont les plus performants sur ces bases d'images.

Mots clés : apprentissage profond, graphe, insectes, arthropodes, réseaux de neurones, convolution

Deep learning onto graph space : application to image-based insect recognition

Abstract :
The goal of this thesis is to investigate insect recognition as an image-based pattern recognition problem. Although this problem has been extensively studied along the previous three decades, an element is to the best of our knowledge still to be experimented as of 2017: deep approaches. Therefore, a contribution is about determining to what extent deep convolutional neural networks (CNNs) can be applied to image-based insect recognition. Graph-based representations and methods have also been tested. Two attempts are presented: The former consists in designing a graph-perceptron classifier and the latter graph-based work in this thesis is on defining convolution on graphs to build graph convolutional neural networks. The last chapter of the thesis deals with applying most of the aforementioned methods to insect image recognition problems. Two datasets are proposed. The first one consists of lab-based images with constant background. The second one is generated by taking a ImageNet subset. This set is composed of field-based images. CNNs with transfer learning are the most successful method applied on these datasets.

Keywords : deep learning, graph, insects, arthropods, neural networks, convolution

université de TOURS

LIFAT
64 avenue Jean Portalis
37200 Tours

LIFAT EA 6300
LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS