Distilling Structure from Imagery:
Graph-based Models for the Interpretation of Document Images

A dissertation submitted by **Pau Riba Fiérrez** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, September 9, 2020

Directors	**Dr. Josep Lladós** Universitat Autònoma de Barcelona Dep. Ciències de la Computació & Centre de Visió per Computador **Dra. Alicia Fornés** Universitat Autònoma de Barcelona Dep. Ciències de la Computació & Centre de Visió per Computador
Thesis Committee	**Dr. Francesc Serratosa** Universitat Rovira i Virgili Tarragona, Catalunya **Dr. Dimosthenis Karatzas** Universitat Autònoma de Barcelona Dep. Ciències de la Computació & Centre de Visió per Computador **Dr. Kaspar Riesen** University of Applied Sciences and Arts Northwestern Switzerland Brugg, Switzerland
International Evaluators	**Dr. Andreas Fischer** Université de Fribourg Fribourg, Switzerland **Dr. Muhammad Muzzamil Luqman** La Rochelle Université La Rochelle, France

This document was typeset by the author using LaTeX 2$_\varepsilon$.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN 978-84-121011-6-4

Printed by Ediciones Gráficas Rey, S.L.

Hi ha un escola perduda allà al mig del Montseny
on només hi estudien els nens...
On només hi estudien els nens que somien en truites.

– Albert Pla

Yo he preferido hablar de cosas imposibles porque de lo posible se sabe demasiado.

– Silvio Rodríguez

Real stupidity beats artificial intelligence every time.

– Terry Prachett

A tots els que han sofert aquest llarg camí...

Agraïments

It is good to have an end to journey toward;
but it is the journey that matters, in the end.

– Ursula K. Le Guin

Finalment, un dijous de Corpus, és a dir de Patum, em trobo escrivint aquestes línies. Potser les més difícils d'aquesta tesi per la gran quantitat de gent que he d'agrair i no vull oblidar.

Realitzar una tesi no ha estat només un camí de formació en la recerca, sinó també de formació en la vida mateixa. Aquests anys m'han brindat la possibilitat de conèixer molta gent, tant d'aquí com de fora. Ara que arribo al final, i dono un cop d'ull a tot el que he deixat endarrere, m'adono de tota l'ajuda, coneixement i comprensió de les quals he gaudit. Voldria començar amb un agraïment sincer a totes les persones que s'han creuat amb mi al llarg d'aquest camí. Sou moltes, i espero que la majoria us trobeu interpel·lades en els següents paràgrafs però, si no és així, no és per mala fe, sinó que es fa difícil encabir a tothom en aquestes poques línies. A tots vosaltres, moltes gràcies!

Aquesta tesi no existiria si no fos pel Josep i l'Alícia. No només per ser els directors d'aquesta, sinó per haver-me mostrat, ja fa uns quants anys, que és realment el món de la visió per computador. En aquell temps, mentre encara era estudiant de la carrera, em van donar l'oportunitat d'entrar com a becari al CVC. Crec que aquest va ser el punt d'inflexió que em va fer decidir a fer el salt per realitzar el meu doctorat. Vull agrair al Josep, que hagi sigut capaç de treure temps d'on no en tenia per poder-se reunir, corregir el que escrivia i donar-me idees noves per poder seguir endavant. A l'Alícia, vull agrair-li haver tingut sempre la porta oberta per discutir com enfocar els diferents problemes que m'anaven sorgint. En definitiva, moltes gràcies als dos, sense vosaltres aquest document no hauria vist mai la llum. Així doncs, aquesta tesi, també és vostra.

I take the chance to sincerely acknowledge Andreas Fischer, Joan Bruna and Daniele Panozzo for welcome me at their respective research groups in the Université de Fribourg and the New York University. You made me feel at home at both Fribourg (Switzerland) and New York (USA). I would like to extend my

more sincere appreciation to all the friends, collegues and people I shared some time in these places. Specially, Michele Alberti, Vinaychandran Pondenkandath, Teseo Schneider, Francis Williams, Francisca Gil, Davi Colli, Zachary Ferguson and Bolun Wang. In addition I would like to thanks the colleagues from omni:us who shared a boundless number of videocalls during these years. Specially to Lutz Goldmann.

M'agradaria fer una menció especial a tots els membres del CVC. A les diferents persones de l'administració i servei d'informàtica, especialment a la Claire, l'Ana María, l'Alexandra, la Txell, la Montse, al Marc i al Joan. Als companys de dinars del divendres, en Jordi Font i la Katerine Diaz. En especial al Marçal Rossinyol i al Juan Ignacio Toledo que, a més a més, també han estat companys de grup. A la resta de companys de grup, especialment, al Dimosthenis Karatzas, a l'Ernest Valveny, l'Oriol Ramos i Joan Mas. I, finalment, a tots els amics i companys del CVC: Pau Rodríguez, Arnau Baró, Edgar Riba, Albert Berenguel, Lei Kang, Sounak Dey, Pep Gonfaus, Anjan Dutta, Asma Bensalah, Xavier Soria, Manuel Carbonell i Sanket Biswas, entre molts d'altres. Amb tots vosaltres he compartit gran part d'aquest camí, entre xerrades, cafès i cervesetes. Heu estat font de suport i inspiració durant tota aquesta recerca!

No voldria acabar sense tenir un record pels amics. En especial al Pep, el Víctor i el Joan per haver-hi estat i ser-hi, sempre. També a l'Imma, que em va obrir la porta al món dels grafs.

A l'Anna vull agrair-li haver aparegut a la meva vida amb un gran somriure i en el moment adequat. En aquests darrers mesos, sempre has sabut donar-me forces en els moments que més ho necessitava. Sense tu, això no hauria estat possible. T'estimo.

A la meva família, als meus avis, l'Alfonso, la María de los Ángeles, l'Albert i l'Elvira. També als meus tiets i cosins. A la Pilar, que ha estat al meu costat durant tot aquest temps.

I, per últim, als meus pares, l'Àngels i el Joan. Heu estat el meu suport incondicional i la meva font d'inspiració per arribar fins aquí. Gràcies per animar-me a seguir endavant i a animar-me a fer el que em feia feliç.

Berga, 9 de setembre de 2020

Abstract

From its early stages, the community of Pattern Recognition and Computer Vision has considered the importance of leveraging the structural information when understanding images. Usually, graphs have been selected as the adequate framework to represent this kind of information due to their flexibility and representational power able to codify both, the components, objects, or entities and their pairwise relationship. Even though graphs have been successfully applied to a huge variety of tasks, as a result of their symbolic and relational nature, graphs have always suffered from some limitations compared to statistical approaches. Indeed, some trivial mathematical operations do not have an equivalence in the graph domain. For instance, in the core of many pattern recognition applications, there is a need to compare two objects. This operation, which is trivial when considering feature vectors defined in $\mathbb{R}^n$, is not properly defined for graphs.

Along this dissertation the main application domain has been on the topic of Document Image Analysis and Recognition. It is a subfield of computer vision aiming at understanding images of documents. In this context, the structure and in particular graph representations, provides a complementary dimension to the raw image contents.

In computer vision, the first challenge we face is how to build a meaningful graph representation that is able to encode the relevant characteristics of a given image. This representation should find a trade off between the simplicity of the representation and its flexibility to represent the deformations appearing on each application domain. We applied our proposal to the word spotting application where strokes are divided into graphemes which are the smaller units of a handwritten alphabet.

We have investigated different approaches to speed-up the graph comparison in order that word spotting, or more generally, a retrieval application is able to handle large collections of documents. On the one hand, a graph indexing framework combined with a votation scheme at node level is able to quickly prune unlikely results. On the other hand, making use of graph hierarchical representations, we are able to perform a coarse-to-fine matching scheme which performs most of the comparisons in a reduced graph representation. Besides, the hierarchical graph

representation demonstrated to be drivers of a more robust scheme than the original graph. This new information is able to deal with noise and deformations in an elegant fashion. Therefore, we propose to exploit this information in a hierarchical graph embedding which allows the use of classical statistical techniques.

Recently, the new advances on geometric deep learning, which has emerged as a generalization of deep learning methods to non-Euclidean domains such as graphs and manifolds, has raised again the attention to these representation schemes. Taking advantage of these new developments but considering traditional methodologies as a guideline, we proposed a graph metric learning framework able to obtain state-of-the-art results on different tasks.

Finally, the contributions of this thesis have been validated in real industrial use case scenarios. For instance, an industrial collaboration has resulted in the development of a table detection framework in annonymized administrative documents containing sensitive data. In particular, the interest of the company is the automatic information extraction from invoices. In this scenario, graph neural networks have proved to be able to detect repetitive patterns which, after an aggregation process, constitute a table.

Keywords – Computer Vision, Pattern Recognition, Graph-based Representations, Graph Indexing, Hierarchical Graphs, Graph Embeddings, Graph Neural Networks, Graph Edit Distance, Table Detection.

Resum

Des del seu inici, la comunitat investigadora sobre Reconeixement de Patrons i Visió per Computador ha reconegut la importància d'aprofitar la informació estructural de les imatges. Els grafs s'han seleccionat com el marc adequat per representar aquest tipus d'informació a causa de la seva flexibilitat i poder de representació capaç de codificar, tant els components, objectes i entitats com les seves relacions. Tot i que els grafs s'han aplicat amb èxit a una gran varietat de tasques -com a resultat de la seva naturalesa simbòlica i relacional- sempre han patit d'algunes limitacions comparats amb mètodes estadístics. Això es deu al fet que algunes operacions matemàtiques trivials no tenen una equivalència en el domini dels grafs. Per exemple, en la base de moltes aplicacions de reconeixement de patrons hi ha la necessitat de comparar objectes. No obstant això, aquesta operació trivial no està degudament definida per grafs quan considerem vectors de característiques definits en $\mathbb{R}^n$.

Al llarg d'aquesta recerca, el principal domini d'aplicació està basat en el tema de l'Anàlisi i Reconeixement d'Imatges de Documents. Aquest és un subcamp de la visió per computador que té com a objectiu compendre imatges de documents. En aquest context, l'estructura -particularment la representació en forma de graf- proporciona una dimensió complementària al contingut de la imatge.

En visió per computador la primera dificultat que ens trobem recau en construir una representació significativa de grafs capaç de codificar les característiques rellevants d'una imatge donada. Això es deu al fet que és un procés que ha de trobar un equilibri entre la simplicitat de la representació i la flexibilitat, per tal de representar les diferents deformacions que apareixen en cada domini d'aplicació. Hem estudiat aquest tema en l'aplicació de la recerca de paraules, dividint els diferents traços en grafemes –les unitats més petites d'un alfabet manuscrit–.

També, hem investigat diferents metodologies per accelerar el procés de comparació entre grafs perquè la recerca de paraules o, inclús, de forma més general, l'aplicació en la recerca de grafs, pugui incloure grans col·leccions de documents. Aquestes metodologies han estat principalment dues: (a) un sistema d'indexació de grafs combinat amb un sistema de votació en l'àmbit de nodes capaç d'eliminar resultats improbables i (b) usant representacions jeràrquiques de grafs que duen

a terme la majoria de les comparacions en una versió reduïda del graf original, mitjançant comparatives entre els nivells més abstractes i els més detallats. A més a més, la representació jeràrquica també ha demostrat obtenir una representació més robusta que el graf original, lidiant amb el soroll i les deformacions de manera elegant. Per tant, proposem explotar aquesta informació en forma de codificació jeràrquica del graf que permeti utilitzar tècniques estadístiques clàssiques.

Els nous avenços en aprenentatge profund geomètric han aparegut com una generalització de les metodologies d'aprenentatge profund aplicades a dominis no Euclidians –com grafs i varietats–, i han promogut un gran interès en la comunitat científica per aquests esquemes de representació. Així doncs, proposem una distància de grafs capaç d'obtenir resultats comparables a l'estat de l'art en diferents tasques aprofitant aquests nous desenvolupaments, però considerant les metodologies tradicionals com a base.

També hem realitzat una col·laboració industrial amb la finalitat d'extreure informació automàtica de les factures de l'empresa (amb dades anònimes). El resultat ha estat el desenvolupament d'un sistema de detecció de taules en documents administratius. D'aquesta manera les xarxes neuronals basades en grafs han demostrat ser aptes per detectar patrons repetitius, els quals, després d'un procés d'agregació, constitueixen una taula.

Paraules Clau – Visió per Computador, Reconeixement de Patrons, Representacions basades en Grafs, Indexació de Grafs, Grafs Jeràrquics, Codificació de Grafs, Xarxes Neuronals en Grafs, Distància d'Edició de Grafs, Detecció de Taules.

Resumen

Desde sus inicios la comunidad que investiga el Reconocimiento de Patrones y la Visión por Computador ha reconocido la importancia de aprovechar la información estructural de las imágenes. Los grafos se han seleccionado como el marco adecuado para representar este tipo de información, a causa de su flexibilidad y poder de representación capaz de codificar tanto los componentes, los objetos o las entidades, como sus relaciones. Aunque los grafos se han aplicado con éxito a una gran variedad de tareas –como resultado de su naturaleza simbólica y relacional–, siempre han sufrido algunas limitaciones comparados con los métodos estadísticos. Esto se debe al hecho que algunas operaciones matemáticas triviales no tienen una equivalencia en el dominio de los grafos. Por ejemplo, en la base de la mayoría de aplicaciones de reconocimiento de patrones hay la necesidad de comparar objetos. No obstante, esta operación trivial no está debidamente definida por grafos cuando consideramos vectores de características definidos en $\mathbb{R}^n$.

Durante la presente investigación, el principal dominio de aplicación se basa en la temática del Análisis y Reconocimiento de Imágenes de Documentos. Este es un subcampo de la visión por computador que tiene como objetivo comprender imágenes de documentos. En este contexto, la estructura -particularmente la representación en forma de grafo- proporciona una dimensión complementaria al contenido de la imágen.

En visión por computador la primera dificultad que nos encontramos se basa en construir una representación significativa de grafos que sea capaz de codificar las características relevantes de una imagen. Esto se debe a que es un proceso que tiene que encontrar un equilibrio entre la simplicidad de la representación y la flexibilidad, para representar las diferentes deformaciones que aparecen en cada dominio de la aplicación. Hemos estudiado este tema en la aplicación de la búsqueda de palabras, dividiendo los diferentes trazos en grafemas –las unidades más pequeñas de un alfabeto manuscrito–.

Tambien, hemos investigado diferentes metodologías para acelerar el proceso de comparación entre grafos para que la búsqueda de palabras o, incluso, de forma más general, la aplicación de búsqueda de grafos, pueda incluir grandes colecciones de documentos. Estas metodologías han estado principalmente dos: (a) un sistema

de indexación de grafos combinado con un sistema de votación en el ámbito de los nodos capaces de eliminar resultados improbables y (b) usando representaciones jerárquicas de grafos que llevan a término la mayoría de las comparaciones en una versión reducida del grafo original mediante comparativas entre los niveles más abstractos y los más detallados. Asimismo, la representación jerárquica también ha demostrado obtener una representación más robusta que el grafo original, además de lidiar con el ruido y las deformaciones de manera elegante. Así pues, proponemos explotar esta información en forma de codificación jerárquica del grafo que permita utilizar técnicas estadísticas clásicas.

Los nuevos avances en el aprendizaje profundo geométrico han aparecido como una generalización de las metodologías de aprendizaje profundo aplicadas a dominios no Euclidianos –como grafos y variedades–, y han promovido un gran interés en la comunidad científica por estos esquemas de representación. Proponemos una distancia de grafos capaz de obtener resultados comparables al estado del arte en diferentes tareas aprovechando estos nuevos desarrollos, pero considerando las metodologías tradicionales como base.

También hemos realizado una colaboración industrial con la finalidad de extraer información automática de las facturas de la empresa (con datos anónimos). El resultado ha sido el desarrollo de un sistema de detección de tablas en documentos administrativos. Así pues, las redes neuronales basadas en grafos han demostrado ser aptas para detectar patrones repetitivos, los cuales, después de un proceso de agregación, constituyen una tabla.

Palabras Clave – Visión por Computador, Reconocimiento de Patrones, Representaciones basadas en Grafos, Indexación de Grafos, Grafos Jerárquicos, Codificación de Grafos, Redes Neuronales de Grafos, Distancia de Edición de Grafos, Detección de tablas.

Contents

List of Tables

List of Figures

List of Algorithms

1 | Introduction

But those with the courage to explore the weave and structure of the Cosmos, even where it differs profoundly from their wishes and prejudices, will penetrate its deepest mysteries.

– Carl Sagan

In this chapter we introduce the context and motivation of this thesis regarding pattern recognition and computer vision. In particular, structural pattern recognition and its main challenges in document image analysis are discussed. Moreover, we summarize the main objectives and contributions of this work. Finally, we outline the structure and organization of the dissertation.

1.1 Setting the Context

Pattern Recognition is devoted to, in Cristopher M. Bishop words [21], "*the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories*". Indeed, searching for patterns in nature is not only exclusive to computer science, but has been done since the dawn of human reasoning. From astronomy to biology, observations of such patterns have fostered the advancements of human knowledge [141]. Back in the Ancient Greece, the observations of plants from Theophrastus (*c.* 372 – *c.* 287 B.C.), considered the father of botany, discovered that "*those that have flat leaves have them in a regular series*". Similarly, Pliny the Elder (23 – 79 A.D.) talked about "*regular intervals*" between leaves "*arranged circularly around the branches*". Later in the fifteenth century, Leonardo da Vinci (1452 – 1519) noted the spiral arrangement of leaf patterns with cycles of five, which was the first quantitative element added to the description. Finally, Johannes Kepler (1571 – 1630) pointed out the relation of the Fibonacci numbers to explain the pentagonal form of some flowers. Similarly to them, Pattern Recognition has been evolving during last decades finding the hidden patterns of the data.

This thesis addresses the problem of Pattern Recognition for the specific case of document images. In the following sections, we proceed to introduce the relevant topics and methodologies of this dissertation, starting from a broad perspective and subsequently entering in depth in specific points.

1.1.1 Distilling Patterns from Images

In the particular case of detecting and understanding patterns from images, *Computer Vision* emerged as an interdisciplinary field which, as defined by Richard Szeliski [210], "*has been developing mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery*". Such an artificial visual system, requires not only to extract patterns from imaging signals, but also to interpret and understand them based on some reasoning and previous knowledge.

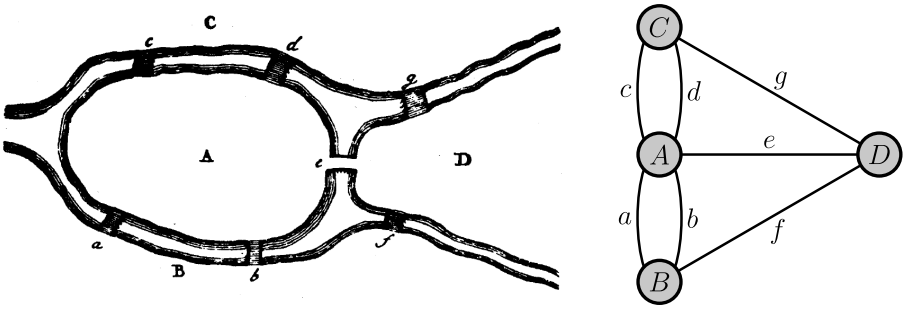
Computer vision emerged in the '60s when artificial intelligence researchers thought of solving the "visual input" problem, *i.e.* dealing with imagery as input data. In particular, Seymour Papert in 1966 proposed a Summer Vision Project[1] as a first attempt to construct a significant part of a visual system. In fact, he claimed it to be "*a real landmark in the development of pattern recognition*". Since then, a boundless number of works have been developed, demonstrating the huge complexity of this discipline in several tasks such as face detection [220], image segmentation [20], image matching [145] among others.

1.1.2 Structural Pattern Recognition

In 1741, Leonhard Euler presented a solution of the problem called *Seven bridges of Königsberg* [72]. At that time, the city of Königsberg, now Kaliningrad, had seven bridges over the Pregel River. These bridges connected the mainlands with two islands as shown in Figure 1.1(a). The problem was to find out whether or not it is possible to cross all bridges passing just once for each of them. In this seminal work, Euler established the foundations of the graph theory by modeling the above problem with the graph in Figure 1.1(b).

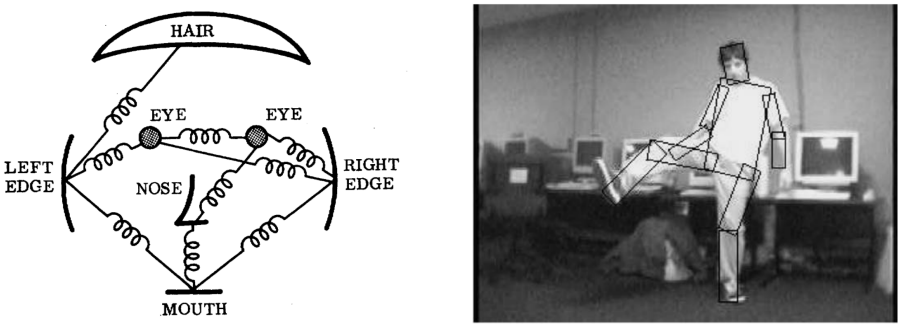
Concerning the particular case of computer vision, graph theory emerged as a tool to model the relationships between the constituent parts of the elements in the images. Taking into account these relations, the structure of the scene is incorporated into several frameworks. *Part-based models*, have been proposed as a representational framework able to model the relevant items and their pairwise relationships. In particular, the first part-based models appeared in the 70's [84] making use of a reference set of rules. Figure 1.2, reprinted from the original papers, shows on the one hand, a schematic representation of a face, indicating its components and their linkages and, on the other hand, a set of parts designed

[1] Available at `http://hdl.handle.net/1721.1/6125`



(a) (b)

Figure 1.1: Graph modeling of the problem called *Seven bridges of Königsberg*. (a) Königsberg map as in 1741. Reprinted from [72]; (b) graph representation of the problem.



(a) (b)

Figure 1.2: Part-based models structure. Reprinted from [75, 84]

to model a human body. This foundational ideas has been evolving over time not only defining a reference model, but allowing the different algorithms to learn object models from training examples [75].

Even though graphs are the appropriate tool to incorporate relationships and correspondences among relevant items, they present the important downside of being computationally complex. Several works have addressed this issue among the years [245]. However, this is still an open issue in the community.

1.1.3 Document Image Analysis and Recognition

Document Image Analysis and Recognition (DIAR) is one of the most relevant topics of Pattern Recognition. Over the years, DIAR has been the driver of huge advances on computer vision. For instance, at the early stages of Pattern Recog-

nition, document images set the paradigms that other topics have later used. An example is the Sayre's paradox, which states that, in the context of handwritten text recognition, individual characters should be previously segmented before being recognized, but to get a reliable segmentation, each character should be previously recognized [194]. In particular, lot of efforts were focused on problems such as *Page Layout Analysis* (PLA) and *Optical Character Recognition* (OCR) techniques. Moreover, DIAR is a multidisciplinary field that, in order to perform properly, it requires to make use not only of document images but also, depending on the problem, natural images [58] or NLP techniques [115].

Structural representations, and in particular graphs, have been widely used in DIAR to deal with a large variety of problems. Bunke [36] compiled the recent advances in graph-based pattern recognition with applications to DIAR. He experimentally demonstrated that these techniques have a great potential, in some cases, even to outperform traditional procedures. Written languages, from the early Summerian glyphs to modern Latin, Chinese or Arabic scripts, exhibit structural patterns that are worth to explore. In this sense, recently, graphs have been proposed to deal with handwritten language [177, 208] and mathematical formulas [143, 148].

From all the different applications of DIAR, the automation of administrative document processing has always received a lot of attention. Even though, in June 30, 1975, Bloomberg Businessweek prophesized paperless offices[2], a major concern nowadays is the automatic processing of invoice documents [175]. The main challenge of these documents is the semi-structured nature of their content. Thus, they do not have a fixed layout, but they share a common set of components, *e.g.* header, footer, recipients. However, administrative documents, suppose a big challenge due to the huge variability of these entities.

1.1.4 Deep Learning in Computer Vision

Deep Learning has transformed a wide variety of applications. It allows computers to "learn" from experience defining complicated concepts with hierarchical relations from simpler ones [99]. In the last decade, the particular case of deep neural networks have supposed a breakthrough in computer vision and artificial intelligence. In fact, the leaders of the deep learning revolution Yoshua Bengio, Geoffrey Hinton and Yann LeCun were awarded with the 2018 ACM A.M. Turing laureata[3], usually recognized as the "Nobel Prize of computing".

In 1998, LeCun *et al.* [132] presented one of the first successful deep learning models, the LeNet architecture. In particular, the paper faces the handwritten digit recognition problem which is a classical DIAR task. In addition, they provide the *Modified National Institute of Standards and Technology* (MNIST) database

[2] "The Office of the Future", Business Week (2387): 48–70, 30 June 1975

[3] `https://awards.acm.org/about/2018-turing`

consisting of images from handwritten digits. With the emergence of deep learning, MNIST became the first standard database for benchmarking CNN's classifiers. Such seminal work setting the foundations of deep learning approaches, was specifically applied in the context of DIAR.

With the great improvements obtained in the *Imagenet Large Scale Visual Recognition Competition*[4] (ILSVRC2012) [129], *Convolutional Neural Networks* (CNN) [132] have become the standard tool to solve most of the computer vision problems. Nowadays, it is difficult to think of any computer vision application that do not use or integrates CNN's. Furthermore, not only computer vision has experienced a revolution, but also *Natural Language Processing* (NLP) and *Speech Recognition*. In particular, the advances with *Recurrent Neural Networks* (RNN) [46, 107] to deal with sequential data have favored the use of neural networks in several fields. Even though the use of Neural Networks is omniscient today for both two-dimensional imaging data or sequential signals, it is still not spread out for the case of structural data.

Lately, *Geometric Deep Learning*[5] (GDL) has emerged as a generalization of deep learning methods to non-Euclidean domains such as graphs and manifolds [30]. This field has arised much attention in the recent years allowing the developed models to encode structural and relational data. Several fields have benefit from this new paradigm, for instance computer vision [239], quantum chemistry [95] and computer graphics [97] among others.

1.2 Motivation

Our research is motivated by the recent success of structural representations on the DIAR domain. In the specific case of DIAR, graphs are able to model the natural deformations of handwriting, relations of graphical entities, external knowledge or even implicit relations among different images.

Since the dawn of the deep learning era, DIAR applications have been taking advantage of the different progresses in that field to achieve outstanding performances. Nonetheless, non-Euclidean domains such as graphs and manifolds were not taking profit of that explosion. It was not until the work of Duvenaud *et al.* [67] that deep learning technique successfully extended to this type of data.

This context, at the time of start working on this thesis, defined some challenges that have guided all our research. On the one hand, following the traditional graph literature, we wanted to study whether or not, graphs are a valid alternative to fully appearance-based methodologies. On the other hand, to incorporate deep learning methodologies to traditional graph-based techniques in order to boost their performance.

[4] Available at `http://www.image-net.org/challenges/LSVRC/`
[5] `http://geometricdeeplearning.com/`

In this thesis, DIAR has been selected as the main application scenario and the topic which leads our research. However, we aimed on advancing prior arts, not only on DIAR but also on generic graph-based techniques applied to other topics such as chemistry or object classification. Therefore, the results presented in this thesis are interdisciplinary in the pattern recognition field.

Summarizing, the motivation of this dissertation is to **incorporate the structure** in computer vision pipelines for classification, retrieval and detection. Therefore, the problem of the thesis is formulated as **structurally-aware techniques in DIAR application domain**.

1.3 Objectives and Contributions of this Thesis

The main objective of this work is **to develop novel techniques for image classification, recognition and retrieval using graphs**. In particular, we propose to exploit the structural information in document collections. In this dissertation, the structure is applied in two different scenarios. First, as a direct representation of shapes from images. In this family, graphs based on strokes or skeletons are considered. Second, the structure of the whole images is taken into account by considering pairwise relationships between different elements. Thus, the document structure is not uniquely defined, but depends on the application scenario.

To this end, we propose to explore two settings. On the one hand, to extend the traditional approaches to deal with classification and retrieval problems such as graph matching and graph embeddings. On the other hand, to explore the new methodologies from geometric deep learning [30] in order to take advantage of this powerful learning-based settings directly in the graph domain.

In this thesis, the main objective has been divided in several sub-tasks detailed into the following points.

Graph-based representations

Although graphs have been widely used in the literature, there is no standard methodology to obtain its representation from a given image. Moreover, in the case of image retrieval and recognition, the practical success of statistical methodologies has faded away other schemes representationally richer but practically infeasible such as graphs.

- *Objective*: To evaluate if such representations are able to perform in a proper way for DIAR problems. Thus, to demonstrate them as a valid alternative to traditional appearance-based systems despite the complexity induced by graphs natural flexibility.
- *Contribution*: To achieve this objective, we will contribute with a graph-

based word spotting approach. Hence, handwritten words are modeled by means of graphs based on handwritten strokes. The retrieval is then formulated in terms of graph distances.

Large-scale setting

The problem of retrieval involves a large number of distance computations. Moreover, document collections are huge, therefore, using graph distances as a retrieval mechanism becomes extremely inefficient taking into account the required number of graph comparisons.

- *Objective*: To leverage the power of graph representations for large-scale scenarios. Hence, to overcome the computational time complexity of these representations.
- *Contribution*: Inspired by voting schemes, in this work we propose to deal with a large-scale graph retrieval taking advantage of a novel graph node indexation. Thus, the retrieval is formulated in terms of node voting according to its Hamming distance.

Noisy graph representations

Obtaining a graph-based representation from a given image is a difficult task, usually leading to noisy representations. Thus, the obtained graphs tend to have spurious nodes and incorrect connections.

- *Objective*: To obtain abstract graph representations which enhance the original graph with high-level information, as well as, hierarchical connections.
- *Contribution*: Inspired by the works on *Maximally Stable Extremal Regions* (MSER) [153], we contribute to structural approaches with a coarse-to-fine representation for graphs. Therefore, we propose a novel hierarchical construction as well as as new coarse-to-fine graph matching algorithm.

Graph embedding

Until the new advances on geometric deep learning, machine learning tools were not designed to work directly on graph structures. In the literature, graph embeddings were proposed in order to obtain a vectorial representation for a given structure.

- *Objective*: To define a graph embedding approach able to encode the structural information at several abstraction levels.
- *Contribution*: We contribute to the literature by devising a graph embedding approach making use of the hierarchical information designed in the previous stage. Hence, graph abstractions and hierarchical relationships are also taken into account.

Graph metric learning

With the preeminence of deep learning in machine learning approaches, our objective is to make use of these techniques to process graph data.

- *Objective*: To combine the use of deep metric learning techniques with the emerging field of geometric deep learning.
- *Contribution*: Inspired by well-known methodologies such as the Hausdorff edit distance [82], the main contribution of this section is to exploit the learning framework called geometric deep learning [30] in order to allow our system to adapt our graph representations to a predefined task.

Structure in invoice documents

Automatic processing of administrative documents have become a major concern on DIAR. Leading technological companies are very interested in this research field because of their need to process a significant amount of administrative documents. However, due to the confidential nature of these archives, public datasets are not available making research problematic.

- *Objective*: To detect tables in administrative documents. In particular, to invoice documents where tables do not appear in a regular grid.
- *Contribution*: For this objective, two contributions are proposed. Firstly, we release a novel dataset where these documents have been previously anonymized. Therefore, they do not contain sensitive data or any type of content information rather than the structure. Finally, we contribute with an invoice table detection approach based only on the structure of the document. The proposed approach is the result of an industrial collaboration. For confidentiality reasons, they require algorithms able to extract layout information disregarding the textual content itself.

1.4 Organization

The rest of the dissertation is organized in ten chapters and one appendix structured in two parts and two global chapters. This division follows the classical paradigm on graph theory (Part I) and the emerging field of geometric deep learning (Part II). Figure 1.3 presents the thesis outline with the main relations between chapters.

PART I: Graph Representations

The first part of this thesis explores the graph theory that has been traditionally used for Pattern Recognition. In that sense, this part explores the problems of graph retrieval and classification.

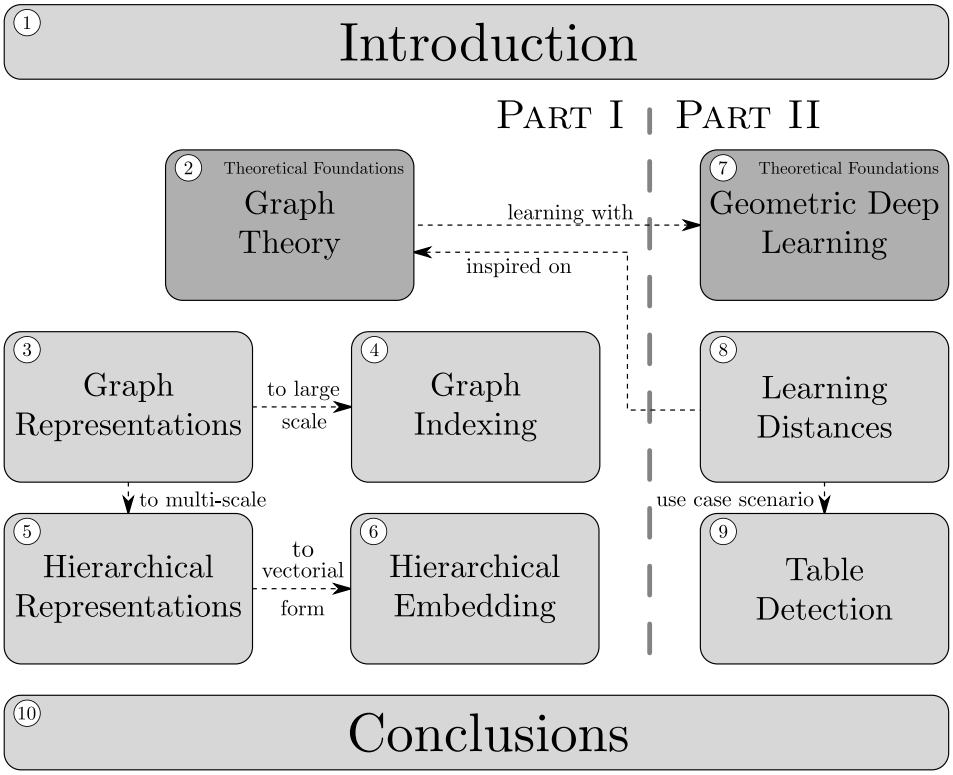


Figure 1.3: Overview of the present dissertation.

- Chapter 2, presents the methodological basis where this thesis is built upon. First, the notations and definitions are introduced. Later, the main techniques used for pattern recognition are reviewed.
- In Chapter 3 we present a graph representation specially suitable for handwritten text. In particular, we face the problem of word spotting in terms of graph retrieval.
- In Chapter 4 we extensively research on the large-scale nature of graph retrieval. Specially, in the context of document analysis, a huge number of comparisons should be performed while the user will expect an answer in real time.
- Chapter 5 focuses on enhancing the graph information by describing the graphs at different abstract levels. These abstractions provide representations that are able to capture the high-level information of the graph. In this sense two problems are tackled, firstly, the noisy data produced by the graph generation and secondly, the large-scale retrieval by performing comparisons on the abstract graphs which are much smaller than the original ones.

- Finally, Chapter 6 takes advantage of the defined hierarchy to generate a graph embedding. The proposed embedding is constructed by graphlet sampling at different hierarchical levels. Thus, the structural information is preserved at low and high level.

PART II: Geometric Deep Learning

In the last decade, deep learning has been able to solve a huge variety of problems. In particular, the research has been mainly focused on data defined in the Euclidean domain. Thus, working with the raw image data or vectorial representations has been the main trend in computer vision. Therefore, non-Euclidean data such as graphs or manifolds was not usually considered. However, the recent advances in the emerging topic of geometric deep learning [30] expanded the frontiers of graph methodologies.

- In Chapter 7 the theoretical foundations on geometric deep learning as well as its current trends and challenges are reviewed. Moreover, some works using these novel methodologies are introduced.
- Chapter 8 extends the idea of graph distances into a learning-based scenario. Thus, we design a new methodology based on the Hausdorff edit distance [82] to learn a novel metric space for graphs.
- In Chapter 9, the industrial needs on processing administrative documents has been taken into consideration. In particular, a industrial contribution with the German company *omni:us*[6] demonstrate that table detection in invoice documents is a problem far from being solved. Moreover, requirements from their clients demonstrate the need to provide a whole anonymized framework able to spot tables based on the structure of the documents.

Finally, Chapter 10 draws the thesis conclusions. Moreover, among the contributions, we present a discussion about some future research lines that have emerged during the development of this thesis.

Together with this work, a compilation of recurrent information along the chapters is provided in form of an appendix. In Appendix A the main datasets used to evaluate the performance of the proposed methodologies are carefully described.

[6] `https://omnius.com/company/`

Part I

Graph Representations

Structural representations have been widely studied in the literature. In particular, graph-based representations are drivers of more powerful approaches for visual recognition and retrieval than statistical approaches. Despite their representational power in front of classical appearance based models, graphs have faced some limitations. The main disadvantages of structural approaches are the time complexity and scalability. Moreover, the lack of mathematical operations defined in graph domain often makes graph-based methods unusable.

2 | Graph Theory for Pattern Recognition

If I have seen further
it is by standing on the shoulders of Giants.

– Isaac Newton

This chapter overviews the relevant literature on graph theory and the main approaches that are relevant in the context of this thesis. Firstly, the main definitions and notations pertinent to this dissertation are introduced. The notation has been inspired in the previous works of Riesen et al. *[183] and Dutta [62]. Secondly, the key task, called graph matching, required for using graphs in a pattern recognition scenario, is introduced. Finally, some approaches to deal with structural representations are presented* viz. *graph indexing; graph kernels and embeddings; and hierarchical graph representations.*

2.1 Definitions and Notations

A graph is roughly defined as a symbolic data structure describing relations (*edges*) between a finite set of objects (*nodes*). Several formal definitions have been proposed in the literature, the following is a flexible definition which stands for a large number of tasks including the ones tackled in this thesis.

Definition 2.1.1 (Graph)**.** Let L_V and L_E be a finite or infinite label sets for nodes and edges, respectively. A *graph* g is a $4-$tuple $g = (V, E, \mu, \nu)$ where,

- V is the finite set of *nodes*, also known as *vertices*,
- $E \subseteq V \times V$ is the set of *edges*,
- $\mu \colon V \to L_V$ is the node labeling function, and
- $\nu \colon E \to L_E$ is the edge labeling function.

$|V|$ denotes the number of nodes of a graph g, namely, the *order* of the graph and $|E|$ the number of edges, in other words, the *size* of the graph. We denote $\mathcal{G}$ as the space of graphs.

Based on the definitions of the labeling sets L_V and L_E, some useful definitions are provided. On the one hand, *Unlabeled* or *Unattributed graphs* are defined as those graphs obtained by assigning the same label ε to all nodes and edges, *i.e.* $L_V = L_E = \{\varepsilon\}$. On the other hand, *Labeled graphs*, also known as, *Attributed graphs* are those graphs whose labels are assigned either by a discret set, *e.g.* a set of integers $L = \{1, 2, 3, \ldots\}$ or symbolic labels $L = \{\alpha, \beta, \gamma, \ldots\}$; or a continuous space, *e.g.* $L = \mathbb{R}^n$, namely, *Discrete* and *Continuous attributed graphs*.

Given a graph $g = (V, E, \mu, \nu) \in \mathcal{G}$, we denote an edge $e \in E$ according to its source u and target v nodes, $e = (u, v)$ where $u, v \in V$. Then, u and v are referred as *adjacent* nodes. Moreover, the *degree* of a node $u \in V$, denoted as $\deg(u)$, is the number of incident edges of u. According to the edges, a graph is classified as directed or undirected graphs.

Definition 2.1.2 (Directed Graph)**.** Let $g = (V, E, \mu, \nu)$ be a graph, it is a *directed graph* if given an edge $e = (u, v) \in E$, the existence of an edge $e' = (v, u) \in E$ is not assured.

Definition 2.1.3 (Undirected Graph)**.** Let $g = (V, E, \mu, \nu)$ be a graph, it is an *undirected graph* if given an edge $e = (u, v) \in E$, it exists an edge $e' = (v, u) \in E$ and $\nu(e) = \nu(e')$.

Inclusion relationships in graphs are defined in a similar way from the set theory.

Definition 2.1.4 (Subgraph)**.** Given a graph $g = (V, E, \mu, \nu)$, another graph $g' = (V', E', \mu', \nu')$ is said to be a *subgraph* of g and is denoted by $g' \subseteq g$ if and only if,

- $V' \subseteq V$
- $E' \subseteq E$
- $\mu'(u) = \mu(u), \forall u \in V'$
- $\nu'(e) = \nu(e), \forall e \in E'$

Moreover, if $E' = E \cap V' \times V'$, g' is known as *induced subgraph* of g.

In order to study a graph g, we need to mathematically describe its structure. In the literature, a common way to describe it is by means of the *degree matrix*, the *adjacency matrix* and the *Laplacian matrix* of g.

Definition 2.1.5 (Degree Matrix)**.** Given a graph $g = (V, E, \mu, \nu)$, with $|V| = n$ nodes, the *degree matrix* $D = (d_{ij})_{n \times n}$ of the graph g is defined by

$$d_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where $v_i \in V$.

Definition 2.1.6 (Adjacency Matrix)**.** Given a graph $g = (V, E, \mu, \nu)$, with $|V| = n$ nodes, the *adjacency matrix* $A = (a_{ij})_{n \times n}$ of the graph g is defined by

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

where $v_i, v_j \in V$.

Definition 2.1.7 (Laplacian Matrix)**.** Given a graph $g = (V, E, \mu, \nu)$, with $|V| = n$ nodes, the *Laplacian matrix* $L = (l_{ij})_{n \times n}$ of the graph g is defined by

$$l_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

where $v_i, v_j \in V$.

Observe that the edges on a graph define the node connectivity. In that sense, the graph is navigated through its edges. Thus, some definitions arise depending on the way the graph is navigated.

Definition 2.1.8 (Walk)**.** A *walk* is a sequence of edges which joins a sequence of vertices.

Definition 2.1.9 (Trail)**.** A *trail* is a walk in which all edges are distinct.

Definition 2.1.10 (Path)**.** A *path* is a trail in which all vertices are distinct.

2.2 Graph Matching

In the core of many pattern recognition applications there is the need to compare two objects. This operation, which is trivial when considering feature vectors, is not properly defined in graphs [50, 85]. In addition, taking into account feature vectors defined in the $\mathbb{R}^n$ space, each vector position i usually encodes the same information and is efficiently compared. However, due to the inherent graph flexibility, we are forced to adopt some definitions ad hoc to particular purposes.

Definition 2.2.1 (Graph Comparison Problem [22])**.** Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be two graphs from $\mathcal{G}$, the graph comparison problem is to find a function

$$d \colon \mathcal{G} \times \mathcal{G} \to \mathbb{R}$$

such that $d(g_1, g_2)$ quantifies the dissimilarity (similarity) of g_1 and g_2.

In general, the terms *graph matching* and *graph comparison* do not refer to the same problem. The first one is defined as the task of finding a mapping operation

between nodes and edges of two graphs, whereas the second one aims to quantify a (dis)similarity score between them. However, in the literature, *graph matching* and *graph comparison* sometimes are interchangeable used as the former one is just used to compute a proximity measure.

The notion of identity is the first that emerges when dealing with comparison problems. Thus, the goal of *exact graph matching* is to decide whether two graphs, or at least a (sub)graph of them, are identical in terms of structure and labels. Then, the identity between two graphs g_1 and g_2 is established by a mapping function called *graph isomorphism*.

Definition 2.2.2 (Graph Isomorphism)**.** Given two graphs $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$, a *graph isomorphism* is a bijective function $f\colon V_1 \to V_2$ satisfying

- $\mu_1(u) = \mu_2(f(u)),\ \forall u \in V_1$,
- $\forall e_1 = (u, v) \in E_1,\ \exists e_2 = (f(u), f(v)) \in E_2$ such that $\nu_1(e_1) = \nu_2(e_2)$,
- $\forall e_2 = (u, v) \in E_2,\ \exists e_1 = (f^{-1}(u), f^{-1}(v)) \in E_2$ such that $\nu_1(e_1) = \nu_2(e_2)$.

The two graphs g_1 and g_2 are called to be isomorphic, and denoted by $g_1 \cong g_2$, if and only if there exists a graph isomorphism between them.

In 1979, graph isomorphism was included in a list of three decision problems for which it was not yet known whether they are P or NP-complete. The other two classical problems are *linear programming* (LP) and *prime number testing* (PRIMES). Nowadays, these two problems have been proved to belong to P in [10] for LP and in [2] for PRIMES. Moreover, researchers have defined a new class called GI as the set of problems with a polynomial-time Turing reduction to the graph isomorphism problem. Then, graph isomorphism is a GI-complete problem [154]. Babai [13] demonstrated that computing graph isomorphisms in quasi-polynomial time is possible.

Equivalently, the graph isomorphism definition is extended to subgraphs. Now, the idea is just to find a part of a graph which is identical to a another graph.

Definition 2.2.3 (Subgraph Isomorphism)**.** Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be graphs. An injective function $f\colon V_1 \to V_2$ is a *subgraph isomorphism* if there exists a subgraph $g \subseteq g_2$ such that f is a graph isomorphism between g_1 and g.

Subgraph Isomorphism is a well-known NP-complete problem.

Even though these definitions define relations between two graphs in a similar fashion as equality for sets, in pattern recognition we are interested in the possibility of comparing two graphs in a more flexible way.

2.2.1 Error-tolerant Graph Matching

In pattern recognition, the constraints imposed by exact graph matching are unrealistic for real applications. Usually, real data is noisy and the extracted representation is influenced by this noise. Error-tolerant or inexact graph matching aims to establish a (sub)graph isomorphism that may include some distortions. The type of distortions that are considered are application dependent. For instance, Zhou and de la Torre [245] proposed to factorize the large pairwise affinity matrix into smaller matrices that encode, on the one hand, the local node structure of each graph and on the other hand, the pairwise affinity between nodes and edges.

One of the most popular error-tolerant graph matching methods is based on the *Graph Edit Distance* (GED) [34, 88, 192]. Roughly speaking, the problem consists in finding the minimum transformation cost of one of the graphs such that an isomorphism exists between the transformed graph and the second one. Formally, GED is defined as,

Definition 2.2.4 (Graph Edit Distance)**.** Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be the source and the target graphs respectively. The *graph edit distance* between g_1 and g_2 is defined by

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_2)\in\Upsilon(g_1,g_2)} \sum_{i=1}^{k} c(e_i),$$

where $\Upsilon(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, and $c(e)$ denotes the cost function measuring the strength of the edit operation e.

A typical distortion model is inspired by string matching. It includes the insertion, deletion and substitution of both vertices and edges. However, other operations such as node split or merge can be considered. These distortions are called edit operations and have an associated cost. Figure 2.1 illustrates an edit path between two graphs. Usually, *branch and bound* techniques are used to compute the minimum cost edit sequence from one graph to another. However, its complexity is exponential in the number of nodes of the involved graphs. Thus, approximate or suboptimal variations of graph edit distance have been proposed to overcome this difficulty.

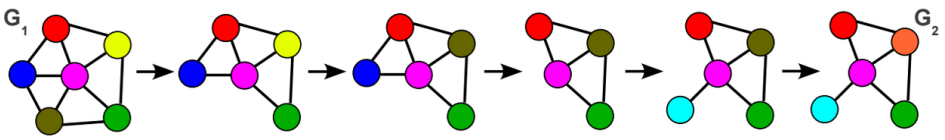


Figure 2.1: Examples of edit paths from one graph to another in terms of insertions, deletions and substitutions. Reprinted from [183].

The *Assignment Edit Distance* (AED) also known as *Bipartite Graph Matching* [181] is one of the most efficient methods for error-tolerant graph matching. It

is based on defining a matrix of edit costs between the nodes of both graphs. The best correspondence between nodes is found by a linear assignment method.

The matrix definition for the AED algorithm takes into consideration both, the local structure of the vertices and their attributes. Let $g_q = (V_q, E_q, \mu_q, \nu_q)$ be the input or query graph and $g_t = (V_t, E_t, \mu_t, \nu_t)$ be the target graph with $V_q = \{u_1, \ldots, u_n\}$ and $V_t = \{v_1, \ldots, v_m\}$, respectively. The cost matrix C is defined as:

$$C = \left[\begin{array}{cccc|cccc} c_{1,1} & c_{1,2} & \cdots & c_{1,m} & c_{1,\varepsilon} & \infty & \cdots & \infty \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} & \infty & c_{2,\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n,1} & c_{n,2} & \cdots & c_{n,m} & \infty & \cdots & \infty & c_{n,\varepsilon} \\ \hline c_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon,n} & 0 & \cdots & 0 & 0 \end{array}\right]$$

where $c_{i,j}$ denotes the cost of a node substitution $c(u_i \to v_j)$; $c_{i,\varepsilon}$ denotes the cost of a node deletion $c(u_i \to \varepsilon)$; and $c_{\varepsilon,j}$ denotes the costs of a node insertion $c(\varepsilon \to v_j)$. A suboptimal graph edit distance between g_q and g_t is computed by a linear assignment algorithm [159]. A key decision in the matching algorithm is the definition of the cost functions.

This algorithm is able to obtain an upper bound of the real GED with a cubic time complexity of $\mathcal{O}(n^3)$ where $n = |V_q| + |V_t|$.

Despite obtaining a good approximation, the time complexity of the AED algorithm is still a problem for some applications where the time is an important constrain. In order to alleviate this issue, Fischer *et al.* [82] proposed the *Hausdorff Edit Distance* (HED) which is a lower bound approximation of the real GED with a quadratic time complexity of $\mathcal{O}(n_1 \cdot n_2)$ where $n_1 = |V_q|$ and $n_2 = |V_t|$.

HED is based on the *Hausdorff distance* (HD) of two sets A and B on a metric space. With a given metric $d(a, b)$ where $a \in A$ and $b \in B$, HD is defined as

$$\mathrm{H}(A, B) = \max\big(\sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(a, b)\big).$$

For finite sets A, B the Hausdorff distance is defined as

$$\mathrm{H}(A, B) = \max\big(\max_{a \in A} \inf_{b \in B} d(a, b), \max_{b \in B} \inf_{a \in A} d(a, b)\big).$$

By definition, the Hausdorff distance is very sensitive to outliers. Hence, in their work they propose to replace the maximum operator with the summation operation, which forces the distance to take into account all nearest neighbour distances and becomes more robust to noise than the original one. Thus, the new distance

is defined as

$$\hat{\mathrm{H}}(A,B) = \sum_{a\in A} \min_{b\in B} d(a,b) + \sum_{b\in B} \min_{a\in A} d(a,b). \tag{2.1}$$

From Equation 2.1 a new distance on graphs is defined. Given two graphs $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ and a matching cost defined as $c = \{c_n, c_e\}$ for nodes an edges respectively, the HED is defined as,

$$\mathrm{HED}(g_1, g_2, c) = \sum_{u\in V_1} \min_{v\in V_2\cup\{\varepsilon\}} c_n^*(u,v) + \sum_{v\in V2} \min_{u\in V_1\cup\{\varepsilon\}} c_n^*(v,u) \tag{2.2}$$

where $c_n^*(u,v)$ is a modified node matching cost defined as

$$c_n^*(u,v) = \begin{cases} \frac{c_n(u,v)}{2}, & \text{if } (u \to v) \text{ is a substitution} \\ c_n(u,v), & \text{otherwise.} \end{cases}$$

This redefinition of the node matching cost is needed because HED does not enforce bidirectional substitutions. Equation 2.2 is composed by a summation, hence, only if both directions are considered, the full cost will be taken into account. The same matching algorithm is considered for the edge matching and incorporated in $c_n^*(u,v)$ if needed. Let us consider two sets of edges, $P = \{p_1, \ldots, p_{|u|}\}$ adjacent to u and $Q = \{q_1, \ldots, q_{|v|}\}$ adjacent to v, the edge matching cost is

$$\mathrm{c_e}(u,v) = \sum_{p\in P} \min_{q\in Q\cup\{\varepsilon\}} c_e^*(p,q) + \sum_{q\in Q} \min_{p\in P\cup\{\varepsilon\}} c_e^*(q,p) \tag{2.3}$$

with the corresponding modified edge matching cost c_e^*.

In this setting, the HED finds an optimal assignment per node instead of a global optimal assignment pretended by the real GED.

A typical drawback of GED approximation algorithm is that they only rely on local edge structures rather than global information. Some efforts have been made to improve the performance by increasing the node context at matching time [83]. However, obtaining a better knowledge on the relation of each node within the graph is still an open issue. Recently, novel deep learning-based approaches have been proposed in order to address this issue [176].

2.3 Graph Indexing

In spite of the great efforts to obtain efficient algorithms to compare two graphs, in many applications their use is still unfeasible due to the required number of comparisons. Thus, an open problem when dealing with these representations is how to efficiently process graph queries. Let us consider the graph retrieval problem which is defined as

Definition 2.3.1 (Graph Retrival Problem)**.** Given a query graph g_q and a database of graphs $\mathcal{D}_G = \{g_1, \ldots, g_T\}$, a *graph retrieval* problem consists in finding the (sub)graphs $g_i \in \mathcal{D}_G$ "similar" to g_q.

Thus, it is defined as finding inexact (sub)graph matchings between the query and the target graphs. Then, indexation strategies are required to prune the required number of graph comparisons.

Performing a sequential search by means of the graph matching algorithms introduced in the previous sections is unfeasible not only because of the inefficiency of graph matching, but also for the dataset size that has to be accessed. To solve this problem, graph indexing has been proposed in the literature. The idea is to build graph indices in order to assist in the processing of graph queries.

Generally, graph indexing is solved by graph factorization techniques where the dataset of graphs is decomposed into smaller ones representing a codebook of compounding structures. The indexation is therefore formulated in terms of indexing the constituent graphs organized in a look-up table structure. Usually, path-based methods are used to split the graphs into small redundant fragments.

For example, *GraphGrep* [199] enumerates all the existing paths up to a predefined length. This reduces the search space performing the exact matching using only few graphs. One of the most relevant and recent works in graph indexing was proposed by Yan *et al.* [237]. They propose to use frequent substructures instead of path-based methods as indexing features. Frequent graph fragments are obtained by graph sequentialization, according to a depth first search (DFS) traversal of the graph edges. Edge sequences are organized in a prefix tree called the *gIndex* tree. The approach is applied to protein graphs. Another approach proposes to organize the constituent graphs in a decision tree based in decompositions of permutations of the adjacency matrix [155]. At run time, subgraph isomorphisms are detected by means of decision tree traversal. The complexity for indexing is polynomial in the number of input graph vertices, but the decision tree is of exponential size. A similar approach based on the construction of a graph lattice was proposed in [193]. The performance for large scale retrieval is achieved by matching many overlapping and redundant subgraphs. Cheng *et al.* [44] proposed an indexing technique for graph databases. It is based on constructing a nested inverted-index, called *FG-index*, based on the set of frequent subgraphs. The above methods are good for graphs with single labels and without strong distortion degree.

2.4 Graph Kernels and Embeddings

Due to graphs symbolic and relational nature, some limitations arise if we compare them with the traditional statistical (vector-based) representations. For example, computing pairwise sums or products (which are elementary operations in many classification and clustering algorithms) is not defined in a standard way in the

graph domain. In the literature, a possible way this problem has been addressed is by means of embedding functions. Two families of graph embeddings can be distinguished, *explicit graph embeddings* [35, 93, 147, 193, 201] and *implicit graph embeddings* [61, 90, 108, 117]. For the sake of simplicity in the literature they are usually called *graph embeddings* and *graph kernels* respectively. In this dissertation we will follow that notation.

However, defining such embedding functions is extremely challenging, when the constraints on time efficiency and the preservation of the underlying structural information is concerned. The problem becomes even more difficult with the growing size of graphs, as the structural complexity increases the possibility of noise and distortion in structure, and raises risk of loosing information.

2.4.1 Graph Embedding

Explicit graph embedding refers to those techniques that aim to explicitly map graphs to vector spaces. Formally,

Definition 2.4.1 (Graph Embedding)**.** Given a graph domain $\mathcal{G}$, a *graph embedding* is a function which maps arbitrary graphs from $\mathcal{G}$ to a real vector space $\mathbb{R}^n$,

$$\begin{aligned} \varphi \colon \mathcal{G} &\to \mathbb{R}^n \\ g &\mapsto \varphi(g) \end{aligned}$$

The methods belonging to this category are further divided into four different classes according to its nature.

- **Graph probing** approaches [147], needs measuring the frequency of specific substructures, that capture content and topology, into graphs. Based on different graph substructures, such as, nodes, edges, subgraph etc., different embedding techniques have been proposed. For example, Shervashidze *et al.* [201] studied the non-isomorphic graphlets, albeit, node label and edge relation statistics are considered by Gibert *et al.* [93]. Saund in [193], introduced a bottom up graph lattice in order to efficiently extract the subgraph features in preprocessed administrative documents, while Dutta and Sahbi [66] proposed a distribution of stochastic graphlets for embedding graphs into a vector space.
- **Spectral graph theory** based methods [37, 114, 125, 127, 186, 232], which aims to analyze the structural properties of graphs in terms of the eigenvectors/eigenvalues of the adjacency or Laplacian matrices of a graph [232]. Recently, Verma and Zhang [219] proposed a family of graph spectral distances for robust graph feature representation. Despite of their relative successes, spectral methods are quite prone to structural noise and distortions.

- Those based on **Dissimilarity measurements** introduced in [167], in this context, Bunke and Riesen have presented several works on the vectorial description of a given graph by its distances to a number of pre-selected prototype graphs [25, 35, 182, 184].

- **Geometric deep learning** approaches motivated by the recent advances on deep learning and neural networks. Many researchers have proposed to utilize neural network for obtaining a vectorial representation of graphs [11, 54, 95, 123, 163].

2.4.2 Graph Kernel

Implicit graph embedding or graph kernel methods is primarily another way to embed graphs into a vector space. They are also popular for the ability to efficiently extend the existing machine learning algorithms to non-linear data, such as, graphs, strings, etc. The idea behind graph kernels is to find a function able to map the input graph into a *Hilbert space* which basically defines a way to compute the similarity between two graphs in terms of a dot product.

Definition 2.4.2 (Hilbert Spaces)**.** A *Hilbert space* $\mathcal{H}$ is a finite- or infinite-dimensional vector space with an inner product $\langle \cdot, \cdot \rangle$ such that the norm defined by

$$|f| = \sqrt{\langle f, f \rangle}$$

turns $\mathcal{H}$ into a complete metric space.

Definition 2.4.3 (Graph Kernels)**.** Given a graph domain $\mathcal{G}$, a *graph kernel*, also known as *implicit graph embedding* is a function $\kappa \colon \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that a mapping $\varphi \colon \mathcal{G} \to \mathcal{H}$ to a Hilbert space $\mathcal{H}$ exists, such that $\kappa(g_1, g_2) = \langle \varphi(g_1), \varphi(g_2) \rangle$ $\forall g_1, g_2 \in \mathcal{G}$.

Traditionally, graph kernel methods are roughly divided into three main categories:

- **Diffusion kernels** are based on the similarity measures among the subparts of two graphs, and then propagating them on the entire structure to obtain a global similarity measure for two graphs [130, 206].

- **Convolution kernels**, aim to measure the similarity of composite objects (modeled with graph) from the similarity of their parts (*i.e.* nodes) [227]. This type of graph kernel derives the similarity between two graphs g_1, g_2 from the sum, over all decompositions, of the similarity products of the subparts of g_1 and g_2 [161]. Recently, Kondor and Pan [126] proposed multi-scale Laplacian graph kernel having the property of lifting a base kernel defined on the vertices of two graphs to a kernel between graphs.

- **Substructure kernels** are based on the analysis of the common substructures that belong to both graphs. This family includes the graph kernel methods that consider random walks [90, 221], backtrackless walks [12], shortest paths [23], subtrees [201] and graphlets [203] as the substructure.

Different from the above three categories, Shervashidze *et al.* [202] proposed a family of efficient graph kernels on the Weisfeiler-Lehman test of graph isomorphism, which maps the original graph to a sequence of graphs. More recently, inspired by the successes of deep learning, Yanardag and Viswanathan [238] presented a unified framework to learn latent representations of substructures for graphs. They claimed that given a pre-computed kernel of graphs, their proposed technique produces an improved representation that leverages hidden representations of sub-structures.

2.5 Hierarchical Representations

Hierarchical graph representations depict explicitly relations at different levels, thus, complex concepts are build from simpler ones. Therefore, obtaining a hierarchical representation from a given graph explores explicitly abstract information that appears implicitly in the original graph.

Definition 2.5.1 (Hierarchical Graph)**.** Given a graph $h = (V, E, \mu, \nu)$, it is a *hierarchical graph* if,

- The graph edges are divided in two sets E_n and E_h namely, the neighbouring and hierarchical edges, where $E = E_n \cup E_h$ and $E_n \cap E_h = \emptyset$.
- Let L_{E_n} and L_{E_h} be a finite or infinite label sets for neighbouring and hierarchical edges. Then, we define two functions, $\nu_n \colon E_n \to L_{E_n}$ and $\nu_h \colon E_h \to L_{E_h}$ where,

$$\nu(v) = \begin{cases} \nu_n(e) & \text{if } e \in E_n \\ \nu_h(e) & \text{if } e \in E_h \end{cases}$$

- The hierarchical graph h can be decomposed in several subgraphs $g_i = (V^i, E^i_n, \mu, \nu)$ with $|V^0| < |V^1| < \cdots < |V^n|$ and
  - $V = \bigcup_{i=0}^{n} V^i$ and $\bigcap_{i=0}^{n} V^i = \emptyset$;
  - $E_n = \bigcup_{i=0}^{n} E^i_n$ and $\bigcap_{i=0}^{n} E^i_n = \emptyset$.
- The hierarchical edges only relate nodes belonging to contiguous subgraphs in the sequence, *i.e.* $E_h = \bigcup_{i=0}^{n-1} E^i_h$ and $E^i_h \subseteq V^i \times V^{i+1}$.

Hierarchical graph representations have been widely used in the literature. These structural representations have been proposed either to increase the information provided by a single graph or to obtain a robust representation dealing with deformations and noise. An elegant way to deal with these problems is to construct a hierarchical representation to handle different levels of detail, noise or abstraction. The idea of using a coarse-to-fine representation for graphs has its analogy in scale-space representations, like *maximally stable extremal regions* (MSER) [153] in images.

Jolion [112] proposed a graph decimation to built hierarchical graphs. Eggert *et al.* [69] present the idea of *Scale Space Aspect graph* for 3D objects. The *Aspect graph* is a data structure that incorporates information about a series of two-dimensional views of the 3D object. This representation captures the structure of an object using different scale of details. This methodology is able to deal with object recognition at different resolutions instead of assuming infinitely high resolution images. This is specially important when some of the features cannot be found if the image is too small. Ulrich and Steger [215] propose to combine the idea of scale-space aspect graphs with the idea of similarity-based aspect graphs to develop a fast 3D object recognition approach. Brun and Kropatsch [31] introduce a set of relationships between regions of a partition through irregular graph pyramids. This hierarchical representation is used within the segmentation framework to encode a hierarchy of partitions. Also a segmentation framework is presented in [98] by Goffe *et al.* They propose to use a tiled top-down pyramids allowing different resolution levels of segmentation. In [150], Marfil *et al.* present a survey on pyramid segmentation algorithms on regular and irregular pyramids. Deruyver *et al.* [56] proposes the use of a semantic graph describing a previous knowledge and integrate this graph in the decimation process of an adaptive pyramid. Broelemann *et al.* [27, 28] propose to deal with noise such as spurious nodes and edges through a hierarchical representation of plausible graphs. Ahuja and Todorovic [3] present a region based approach for object recognition based in multi-scale region segmentation. Conte *et al.* [49] propose a similar graph multi-resolution approach in order to improve the object tracking in a video. Recently, Mousavi *et al.* [158] have proposed a hierarchical representation that is later used to create an embedding combining the different levels of abstraction. Their work focuses on enriching a graph embedding with the information provided by hierarchy levels.

Lately, taking advantage of the new learning-based methodologies, *i.e.* geometric deep learning [30], Ying *et al.* [242] propose to use a graph neural network (GNN) to learn a hierarchical representation for graph classification.

3 | A Graph-based Representation for Handwritten Words

If you only read the books that everyone else is reading, you can only think what everyone else is thinking.

– Haruki Murakami

Graph-based representations have been widely proposed in different fields such as pattern recognition, bioinformatics or chemistry. Graphs are powerful and flexible representations able to capture the structural information of the data. This chapter introduces the concept of graph-based representations for patter recognition, and, in particular, the application scenario is word spotting for handwritten documents. Contrary to most word spotting techniques, which use statistical representations, we propose a structural representation suitable to be robust to the inherent deformations of handwriting. Attributed graphs are constructed using a part-based approach. Graphemes extracted from shape convexities are used as stable units of handwriting, and are associated to graph nodes. Then, spatial relations between them determine graph edges. Spotting is defined in terms of an error-tolerant graph matching, namely assignment edit distance. Historical documents are used as experimental framework. The approach is comparable to statistical ones in terms of time and memory requirements.

3.1 Introduction

Graph-based representations have been widely used in different fields such as pattern recognition [50, 85, 218], bioinformatics [95], etc. Graphs are powerful and flexible representations able to describe shapes in terms of relationships between constituent parts or primitives. However, in some cases, and, in particular in pattern recognition and computer vision, obtaining the graph representation from the input image is not straightforward.

The practical success of machine learning methods applied to image representations faded away other schemes representationally richer but with a higher complexity. However, to tackle complex recognition problems, methods not exclusively based on appearance but enriched with more abstract visual information, such as visual structure of objects, are required. Although the first attempts of part-based

descriptors suggesting graph representations were presented long ago [84], it has been in the last decade when structural models have gained importance in computer vision. Graph representations are implicitly or explicitly drivers of more powerful approaches for visual recognition and retrieval.

Traditionally, obtaining a graph representation from an image is achieved by means of some preprocessing steps. Specifically in document analysis where graphical documents are used, vectorization strategies were quickly adopted. *Vectorization* is defined as approximating the binary images to a polygonal representation using some points, lines, arcs and other entities. A classical vectorization algorithm is the one proposed by Rosin and West [187]. This algorithm, provides a set of critical points and the lines codifying whether any two points are connected in the image skeleton. From this data, several strategies can be used to construct a graph, for instance, points and lines can be directly considered as nodes and edges. However, other paradigms can be adopted.

Recently, *Scene Graphs* [111, 234, 239], defined as a structured representation of an image, where nodes correspond to object labels and bounding boxes, and edges correspond to the pairwise relationships between objects, have been adopted as a new step towards scene understanding. In comparison to other approaches used for image understanding, such as object detection [174] or image captioning [235], scene graph generation provide a more detailed knowledge of the image contents as well as a detailed view of its relations. Lately, with the emergence of deep learning techniques, methodologies involving scene graphs are gaining importance. Figure 3.1 presents a a comparison between the information provided by an object detection approach and a scene graph generation framework.

In this chapter, we propose to tackle the problem of getting a robust and meaningful representation of an image. This problem is studied in the context of DIAR, in particular, in the word spotting application. Therefore, a desirable graph representation must be able to capture the structure of handwritten words while tolerating the deformations inherent to the different writing styles. In that direction, we propose a graph representation in terms of a codebook of graphemes [51] or allographs [196] and their spatial relationships.

The rest of the chapter is organized as follows. First, Section 3.2 introduces the concept of word spotting and its literature relevant to the current framework. Then, Section 3.3 describes the graph model used to represent handwritten words as well as the edit cost operations that have been considered in the error-tolerant graph matching approach. Section 3.4 reports on the experimental evaluation. Finally Section 3.5 draws the conclusions and delineates the limitations of the proposed approach.

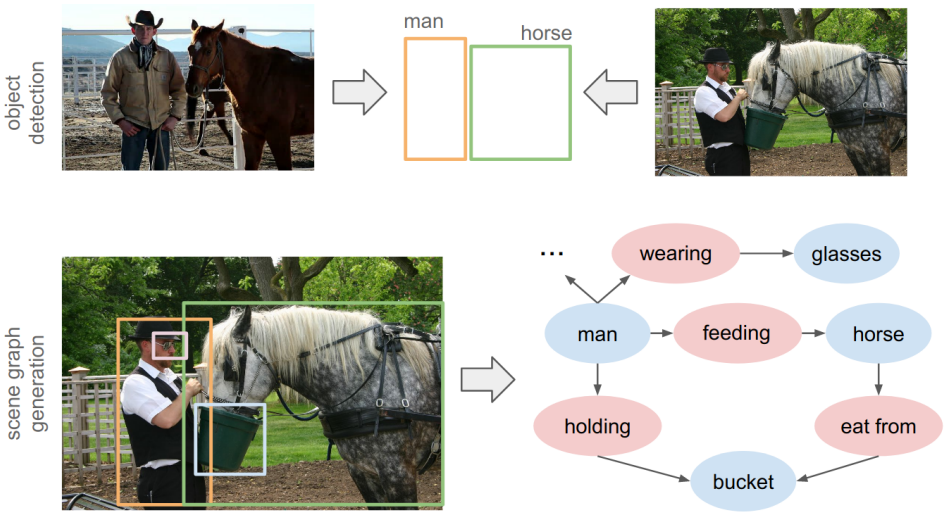
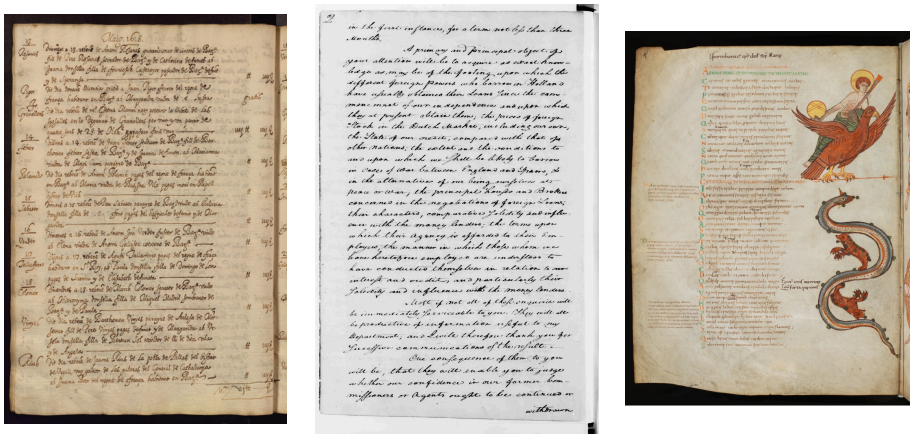


Figure 3.1: Object detection and scene graph generation frameworks comparison. Object detection produce similar results for two semantically different images, in comparison, the scene graph approach is able to capture the objects (blue nodes) and their pairwise relations (red nodes). Reprinted from [234].

3.2 Word Spotting

Word spotting, also known as *Keyword spotting* (KWS), is a content-based image retrieval strategy where, due to the impossibility of a recognition process with enough quality, leans to a visual object detection approach. The key idea of word spotting relies upon obtaining a robust word image representation and a subsequent retrieval scheme. It consists in locating a particular keyword within a document. One of the first successful solutions was proposed by Manmatha *et al.* [149].

KWS has emerged in the last years as a highly effective technique for large scale document image retrieval in manuscript databases. In some cases, in particular in historical documents, a strategy based on full transcription using handwriting recognition approaches, and a subsequent search is nowadays far from being feasible. Hence, word spotting is a practical alternative. Most of the existing word spotting techniques use statistical representations (*e.g.* SIFT, HOG) of the word images [6, 189]. However, some structural representations have been proposed [208, 224]. The main motivation is that the nature of handwriting suggests than the structure is more stable than the pure appearance of the strokes. This is especially important when dealing with the elastic deformations of different handwriting styles. As stated in the comparison of statistical versus structural representations for handwritten word spotting reported in [142], the main disadvantages of structural approaches are the time complexity and scalability to large



(a) **(b)** **(c)**

Figure 3.2: Example of historical document collections. (a) Llibre de les Esposalles; (b) George Washington Papers; (c) Abbey Library of Saint Gall.

document collections.

KWS have been specially successful in historical document collections where the scarcity of available data and the degradation these documents have suffered due to conservation issues makes other approaches impractical. Figure 3.2 shows some examples of historical document collections from different countries and ages.

A word spotting system is defined by two components, (i) a document collection, also known as gallery; and (ii) an input query word. The output of the system is the location in the document collection of words with the same contents as the query. Several divisions of word spotting exist depending on the query, the gallery and the search algorithm.

Firstly, we have two families of algorithms according to the required or available data:

- *Learning-free*: This family of approaches are mostly based on image matching. These methodologies are worth to explore in those settings were labeled training data is not available or it is hard to obtain.
- *Learning-based*: These methodologies require a labeled training set to learn the proper features to use. With the advances on deep learning approaches, learning-based methods have become more popular in the literature, however, in a practical scenario, and specially in historical manuscripts, the amount of data in a collection makes these systems, which are data hungry, unable to learn meaningful representations.

Secondly, according to the user query, word spotting techniques are divided in two

groups:

- *Query-By-String* (QBS): The input query is a string. Usually, these systems try to find a common space between string models build on character, bigram or trigram level and some image features [78].
- *Query-By-Example* (QBE): The input is an image of the word to search [173].

Even though QBS is usually the desired setting for the final user, it requires labeled datasets in order to train the matching approach. This requirement is not necessary in QBE methods where learning-free image matching approaches can be directly used.

Finally, the document collection will define the required feature representation:

- *Segmentation-based*: This set of approaches assume a previous segmentation of the words in the documents. Hence, it expects a previous knowledge of the document structure. In a real scenario, this segmentation is of key importance due to the impact it has on the final performance as reported by Dey *et al.* [57].
- *Segmentation-free*: In these methodologies, each query is directly searched in the whole document. Usually these techniques make use of a sliding window approach. Thus, this second approach is computationally more expensive but avoids a previous segmentation stage.

The proposed graph-based word spotting system follows the classical setting for a learning-free QBE approach with segmented words.

Graphs are robust representations able to describe shapes in terms of the relationships between constituent parts or primitives. They are able to capture the structural invariance among writer styles or the inherent stroke variability of handwriting. One of the first attempts to use graphs-based representations for handwritten words was proposed by Fischer *et al.* [80] which defines a *Hidden Markov Model* (HMM) based on graph similarity features. However, this approach only takes into account the graph nodes for the similarity computation.

To the best of our knowledge, the first approach using graphs for the word spotting problem was proposed by Wang *et al.* [223] and later improved in [224]. The authors propose a coarse-to-fine graph matching. Firstly, a graph embedding approach is used to find words likely to be the query one (coarse selection), afterwards an inexact matching algorithm is employed between connected components and aligned with *Dynamic Time Warping* (DTW) to verify the true positives.

At the same time our approach was proposed, Bui *et al.* [33] developed a graph-based representation based on invariants (prototype strokes). Edges are created by means of four spatial encodings (top, bottom, left and right) and a special symbol

indicating whether they belong to the same connected component or they are the closest nodes in two adjacent connected components.

Later, Stauffer *et al.* [208] developed a KWS system based on several graph representations, namely, *Keypoint*, *Grid*, *Projection* and *Split*. They propose to use an ensemble of those representations to compute a final graph distance. Thus, four Graph edit distances are computed and aggregated together using different strategies such as *minimum*, *maximum* or *mean*.

In general, the use of graph matching for word spotting has to deal with two factors:

1. To obtain robust representation able to tolerate the deformations of handwriting without losing the expressiveness in terms of the topology.
2. To overcome the computational cost of traditional graph matching algorithms. Usually, KWS deals with large amounts of data, hence, making use of these algorithms is unfeasible in a real scenario where the user expects an answer in a matter of seconds.

3.3 A Graph-based Word Spotting Framework

Even though the structure is the main feature representing handwritten words, a pure structural approach is unrealistic due to the deformations present in the data. Thus, the flexibility of graphs allow us to incorporate appearance-based features to the structural representation. Hence, we have to tackle two problems. First, the graph construction should be properly defined. This is a crucial step to obtain a robust representation able to encode the necessary information to build a proper word spotting system. Afterwards, the graph comparison problem dealing with this specific representation should be designed.

3.3.1 Graph Construction from a Word Image

A good representation of the word image is crucial to be able to codify the most characteristic features of each individual handwritten word in order to distinguish the set of categories by means of graph matching techniques. Moreover, it is also important to keep the graphs stable against the deformations of handwriting but robust enough to represent the topology of words in terms of their constituent primitives. Thus, we propose a graph-based representation for handwritten words by means of grapheme graphs.

Definition 3.3.1 (Grapheme)**.** A *grapheme* is each one of the stable constituent units of handwriting, in other words, it is the smallest unit used in describing the writing system of a language.

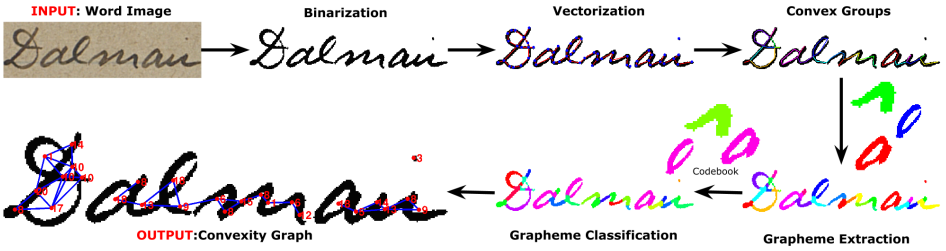


Figure 3.3: Outline of the graph construction process.

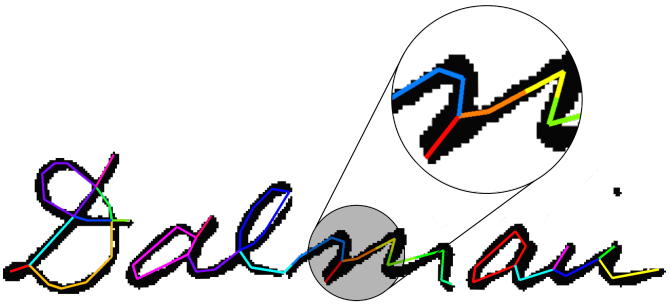


Figure 3.4: Graphemes are extracted from convex groups of the skeleton.

Several works in the literature represent handwriting in terms of graphemes [51] or allographs [196]. In this dissertation, we propose a novel glyph extraction based on the convexities found in the vectorization of the handwritten word image.

Figure 3.3 shows the outline of the proposed graph extraction process. Firstly, the input image is binarized using the classical algorithm proposed by Otsu [164]. The algorithm assumes that the image contains two classes of pixels *i.e.* black and white, then it calculates the optimum threshold value separating both classes. Thus, this algorithm uses the global image information and might be strongly affected by noise caused by degradation or show-through. Secondly, the binary image is vectorized by means of the Rosin West vectorization algorithm [187].

In this dissertation, we consider as graphemes the set of convexities present in the binarized word. To extract the convex paths, the mathematical definition is used,

Definition 3.3.2 (Convex set)**.** We say that a set S is *convex* if, for all $x, y \in S$, the line segment connecting x and y is included in S.

Hence, a pair of points in the vectorized skeleton are considered within the same convex group, if the straight line segment that joins them does not cross any part of the word skeleton. Figure 3.4 illustrates the convex paths given the word "Dalmau". Graphemes are defined as the part of the image foreground extracted from the

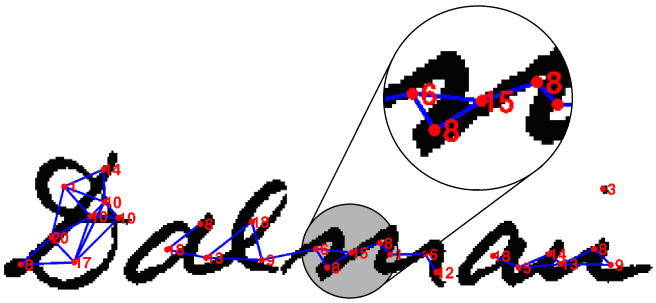


Figure 3.5: Final graph representation of the proposed construction.

geodesic reconstruction from the convex groups. A codebook of graphemes is defined according to a clustering in terms of the *Blurred Shape Model* (BSM) descriptor [71]. The clusters have been generated using all the graphemes in the database and the *k-means* algorithm with k different classes.

Using this set of clusters, a codebook of graphemes is extracted. Finally, graph nodes are associated to the grapheme codewords, and graph edges represent adjacency and proximity relations.

As a result, a graph $g = (V, E, \mu, \nu)$ is generated. Graph nodes V correspond to graphemes, attributed with the corresponding label codeword of the codebook of graphemes. Graph edges E represent adjacency relations between nodes, *i.e.* the corresponding graphemes are connected in the image. The edge attributes are the overlap between the corresponding convex groups (we will refer it as edge weight), the angle and the length. Figure 3.5 depicts the final graph representation with the corresponding codewords as node attributes.

3.3.2 Graph Matching for Word Spotting

The graph edit distance paradigm has been chosen as a tool to compare two graphs, in particular, the *Assignment Edit Distance* (AED) [181] which is a cubic suboptimal approximation of the real graph edit distance. As it has been introduced in Section 2.2.1, AED is based on defining a matrix of edit costs between the nodes of both graphs. The best correspondence between nodes is found by a linear assignment method.

Let $g_q = (V_q, E_q, \mu_q, \nu_q)$ be the graph of the query word and $g_t = (V_t, E_t, \mu_t, \nu_t)$ be the target graph with $V_q = \{u_1, \ldots, u_n\}$ and $V_t = \{v_1, \ldots, v_m\}$, respectively. In this chapter, we will consider a cost function for both nodes and edges. In both cases, insertion, deletions and substitutions are considered.

Insertion and deletion costs. Both, node insertion and deletion costs are computed in the same way, *i.e.* both are seen as deletions in one graph or the

other, and therefore, symmetric. Intuitively, the cost is computed in terms of the local configuration of the node defined by the incident edges. If the node is strongly connected the cost will be higher than for example a simple node that appears disconnected. Following this idea, the designed insertion cost has three terms:

$$c(\varepsilon \rightarrow v_j) = c(u_i \rightarrow \varepsilon) = \mathrm{we}_0\, C_{\mathrm{weightEdges}} + \mathrm{we}_1\, C_{\mathrm{edges}} + \mathrm{we}_2\, t_{\mathrm{vertices}}, \tag{3.1}$$

where we_i are weighting factors; $C_{\mathrm{weightEdges}}$ is the sum of the attributes of the edges incident to the node being deleted, this cost indicates how much the node is sharing part of its grapheme with the neighboring ones; C_{edges} is a measure of the density of the node computed as the ratio between the number of incident edges and the total number of graph nodes; t_{vertices} is a constant value experimentally set as a baseline cost for the edit operation.

Substitution costs. These are computed in terms of the node position, their label according to the codebook and the similarity of the local structure. The different terms are weighted by factors denoted with wn_i. Formally:

$$c(u_i \rightarrow v_j) = \mathrm{wn}_0\, D_{i,j} + \mathrm{wn}_1\, C_{\mathrm{BSM}} + \mathrm{wn}_2\, C_{\mathrm{local\_structure}}, \tag{3.2}$$

where wn_i are different weights; $D_{i,j}$ is the euclidean distance between the spatial position of the nodes u_i and v_j position. This distance is normalized by the maximum node position of the both graphs; C_{BSM} is the L_1 distance between the corresponding BSM shape descriptor of the node graphemes. The value C_{BSM} is computed with the distance between the centroids of the k-means clustering when the codebook is extracted; finally, $C_{\mathrm{local\_structure}}$ is the edit operation cost on the incident edges of the corresponding nodes.

Matching of the incident edges. In order to compute the edit cost on the adjacent edges $C_{\mathrm{local\_structure}}$, the AED algorithm has been used again. Firstly, a matrix of edit costs between the incident edges of both nodes u_i and v_j is defined, namely C_e, which shares the same structure as C. In this case, the cost of edge insertion and deletion is a constant t_{edges}. The substitution costs are computed in terms of the edge attributes *i.e.* weight, angle and length, for both edges to substitute. Therefore, given two edges $e_i \in E_q$ and $f_j \in E_t$, the edge substitution cost is formally defined as

$$c(e_i \rightarrow f_j) = \mathrm{we}_0\, C_{\mathrm{weight}} + \mathrm{we}_1\, C_{\mathrm{angle}} + \mathrm{we}_2\, C_{\mathrm{length}}, \tag{3.3}$$

where we_i are weighting factors; C_{weight} is the difference between the weight of the two edges; C_{angle} is the angle between them and C_{length} is equivalent to $1 - e_{\mathrm{short}}/e_{\mathrm{long}}$, where e_{short} denotes the length of the shorter edge and e_{long} the length of the longer one.

The corresponding values the weights and constants have been empirically set as follows: $t_{\mathrm{vertices}} = t_{\mathrm{edges}} = 0.5$ and $\mathrm{wn}_0 = 2/5$, $\mathrm{wn}_1 = 1/5$, $\mathrm{wn}_2 = 2/5$, $\mathrm{we}_0 = 1/5$, $\mathrm{we}_1 = 2/5$ and $\mathrm{we}_2 = 2/5$. All the cost computations are scaled into the range $(0, 1)$.

With the above considerations, given a query word represented by a graph g_q and a set of target images of handwritten documents represented by graphs g_t^i, word spotting is computed as an inexact subgraph matching where several subgraphs of g_t^i are found as being similar to g_q.

3.4 Experimental Validation

For validating our approach, a historical keyword spotting application has been considered as our main application scenario. In this section, we carefully validate our graph representation for this task.

3.4.1 Experimental Setup

The experimental setup consists of a segmentation-based word spotting scenario. We have used a set of pre-segmented words with the aim of comparing our approach with other methods in the literature [142, 223]. In particular, we show that a graph-based word spotting achieves comparable performance in comparison to other well-known learning-free approaches.

In this experiment, we have used a subset of the *Barcelona Historical Handwritten Marriages Database* (BH2M) [76], which was written in the 17th century. We have evaluated the performance of our method using the 27 pages used in [142]. This set contains $6,544$ segmented words from $1,751$ different transcriptions. All the words having at least three characters and appearing at least ten times have been selected as queries. Thus, there are 514 queries corresponding to 32 different words.

In terms of size, graphs corresponding to query words have an average of 25 nodes. It means that since graph matching is a computationally costly process, it might result in unrealistic elapsed time responses from the application point of view.

The performance has been measured by the *mean Average Precision* (mAP), a classic information retrieval metric [190]. First, let us define *Average Precision* (AP) as

$$\mathrm{AP} = \frac{\sum_{n=1}^{|\,\mathrm{ret}\,|} P@n \times r(n)}{|\,\mathrm{rel}\,|}, \tag{3.4}$$

where $P@n$ is the precision at n and $r(n)$ is a binary function on the relevance of the n-th item in the returned ranked list, rel is the set of the relevant objects with regard to the query, and ret is the set of retrieved elements from the dataset. Then, the mAP is defined as:

$$\mathrm{mAP} = \frac{\sum_{q=1}^{Q} \mathrm{AP}(q)}{Q}, \tag{3.5}$$

where Q is the number of queries.

3.4.2 Spotting Evaluation

Table 3.1 shows the quantitative results and compares them to some other methods in the literature. The proposed approach outperforms most of the methods representing classical families of approaches in the literature (statistical, structural and pseudo-structural). We must notice that the aim of this work is to propose graph matching as a valid alternative for word spotting, in front of the more widespread techniques usually inspired by statistical pattern recognition.

Method	**mAP**
DTW [142]	19.20
Graph-based [223]	24.60
BoVW [142]	30.00
Loci-based [77]	40.06
nrHOG [5]	56.06
Proposed	51.62

Table 3.1: Word Spotting results.

Some qualitative results are shown in Figure 3.6. It is interesting to notice that most words have been correctly retrieved. This example takes the name "Farrer" as a query. The system correctly retrieves the first 11-th words, whereas the 12-th retrieved word corresponds to the name "Barrer". This word indeed corresponds to a very similar word which has the same character length, and only one different letter.

Figure 3.6: Qualitative results for the query "Farrer".

(a) **(b)** **(c)**

Figure 3.7: Problems of the proposed graph-based word spotting approach. (a) Binarization; (b) Shared letters; (c) Lexical variations.

In particular, some typical problems are easily explainable by the structure of the text:

Binarization Problem: The binarization step is crucial as it might provoke degradations in the generation of the graph. This problem appears in one of the queries "Eularia" showed in Figure 3.7(a).

Sharing Parts: Two different words that share most of their letters may have a smaller cost than the correct word with a different writing style. For example, in Figure 3.7(b) "Maria" is retrieved when searching the query "Eularia".

Lexical Variations: This problem is similar to the last one. In this case, the word appears with different lexical variations, for example for plural or feminine. As it is shown in Figure 3.7(c), the query "defunct" has a lower performance than the others due to this fact.

3.5 Conclusions

In this chapter we have described a graph-based representation for hanwritten words. In addition, the robustness of this representation has been evaluated in the classical pipeline for graph-based image retrieval. In particular, the application scenario is word spotting. First, we have shown that graphemes based on convexities are stable under the deformations of handwriting. Second, error-tolerant graph matching approaches have been used to obtain the final ranking. It is important to notice that this matching approach should be properly defined according to the provided graph construction. Hence, the edit costs have been properly defined for our application. Finally, the experimental results demonstrate that our structural approach is comparable to statistical approaches in terms of performance.

However, two limitations emerge when dealing with graphs. Firstly, graph-based representation are very sensitive to noise. In the proposed case, noise appears specially in two construction steps (i) image binarization; and (ii) vectorization. Binarization generates heavy distortions in case of degraded documents, cluttered

backgrounds or vanishing ink. In case of vectorization, spurious edges may appear specially close to the end points. These spurious edges will generate extra nodes that should be managed by the selected matching approach.

The second limitation appears when applying our system to a large-scale retrieval problem. Even though, in terms of performance it is comparable to statistical approaches, the time complexity is unacceptable for a final application. This makes the proposed approach unfeasible for large-scale retrieval.

4 | Information Spotting by Graph Indexing

It is not clear that intelligence has any long-term survival value.

– Stephen Hawking

The huge increase of user-generated contents spread in the cloud has resulted in a need for services including algorithms for searching by content in large databases. In the setting introduced in the previous chapter, retrieving a query graph from a large dataset of graphs implies a high computational time complexity. Hence, it is unfeasible for real applications. With this in mind, in this chapter we propose a graph node indexation formalism applied to the visual retrieval task. For this purpose, binary embeddings are defined as hashing keys for graph nodes. Given a database of labeled graphs, graph nodes are complemented with vectors of attributes representing their local context. Then, each attribute vector is converted to a binary code applying a binary-valued hash function. Therefore, graph retrieval is formulated in terms of finding target graphs in the database whose nodes have a small Hamming distance from the query nodes, easily computed with bit-wise logical operators. As an application example, we validate the performance of the proposed methods in different real scenarios such as handwritten word spotting in images of historical documents or symbol spotting in architectural floor plans.

4.1 Introduction

Content-based image retrieval (CBIR) systems [135] have become more complex in the last years with the increase of the volume of data spread in the cloud. The huge increase of user-generated contents, *e.g.* image repositories and videos in social networks, has resulted in a need for services including algorithms for searching by content in large databases. At the heart of any retrieval system, a visual recognition task is present that efficiently searches for similarities between query and target images. In previous chapters, we have already discussed about the graph construction problem and the main difficulties faced working on these flexible representations for CBIR, mainly the time complexity involved at the graph comparison stage. However, with the increase of data in the cloud, we require of methodologies able to deal with a large-scale setting where graphs have been usually avoided.

Graphs are robust representations offering a paradigm able to deal with many-to-many relationships among visual features and their parts. The use of (sub)graph matching is an effective solution to deal with visual recognition, and in particular content-based visual retrieval. However, one of the major disadvantages of graph matching in visual pattern recognition is the need to deal with the huge complexity of these algorithms. New approaches based on graph embeddings and graph kernels have emerged rapidly [60, 161]. These methods are based on finding an explicit or implicit transformation of the graph to a n-dimensional space so the problem of graph similarity is elegantly reduced to a machine learning problem using classical classification schemes, *e.g. Support Vector Machines* (SVM). Other solutions to reduce the complexity of graph matching are based on graph serialization [64, 185] consisting in reducing the graph to a sequence so the problem can be solved by an alignment algorithm in quadratic time. More recently, an efficient approach based on graph factorization applied to deformable object recognition and alignment have been proposed [245]. Nowadays, deep learning-based approaches are being studied for learning graph structure similarities [138].

As stated before, structural information can play an important role in CBIR. In this scenario, the number of graphs in the database may be extremely large, and also the graphs may be of a considerable size. Although many suboptimal methods for graph matching exist, it is unfeasible to compare a query graph with thousand or million graphs of the database in a sequential way. Therefore, graph indexing approaches must be introduced. Graph indexing aims at reducing the set of graphs that must be tested, pruning non promising matchings in the dataset. A good indexation strategy must be inexpensive in terms of space and time, and must have a high recall *i.e.* low number of false negatives.

In this chapter we propose a graph indexing approach in the application domain of information spotting in document images. Information spotting in document databases can be defined as retrieving the pages or page regions containing a given textual or graphical query. Due to the morphology of textual and graphical symbols, graphs are suitable structures to represent such signs of information. Efficient information spotting approaches based on graph representations require indexation strategies that involve error-tolerant isomorphisms. Our approach is inspired by the ideas of binary encoding for CBIR. The key idea is to extend the attributes associated to graph nodes by an embedding function describing the local context of the node. This vector of attributes is converted to a binary code applying a binary-valued hash function. Therefore, graph retrieval is formulated in terms of finding target (sub)graphs in the database whose nodes have a small Hamming distance from the query nodes. These (sub)graphs suggest the image regions likely to contain an instance of the query. Let us refer them as *candidate regions*.

The problem to be solved is based on the concept of *focused graph retrieval* which is defined as, let $G = \{g_1, g_2, \ldots, g_n\}$ be a graph dataset and g_q be a query graph, the task aims to find candidate regions $\mathrm{cr}_i^j \subseteq g_j,\ j = \{1, 2, \ldots, n\}$ where g_q is

"similar" to cr_i^j, *i.e.* inexact (sub)graph matching. Hence, we want to perform an inexact (sub)graph isomorphism in order to find candidate regions cr_i^j that are likely to contain similar structures to the given query.

The application scenario of our work, information spotting in document images that often contain handwritten information, requires tolerance to high degrees of distortion. It is also a requirement to capture different visual features by means of attribute vectors associated to nodes and edges. To achieve these requirements, we propose a more flexible graph indexing approach based on associating context descriptors to nodes based on their morphology. These descriptors are encoded by binary codes. Binary codes are compact descriptors that capture the local context of an image keypoint, according to a local neighborhood pattern, and represent it with a vector of bits. One of the most promising local descriptors is the efficient BRIEF descriptor [39]. BRIEF is a binary descriptor that aims at quickly comparing local features while requiring few amounts of memory. The BRIEF descriptor outputs a set of bits obtained by comparing intensities of pairs of pixels within the local key-region. However, other binary descriptors have been proposed with huge success, some relevant examples are, ORB [188], BRISK [134] and FREAK [4]. The good property of binary codes is that, since they are represented as vectors of bits, the comparison between two of them can be quickly computed with basic logical operations, usually XOR, using directly the CPU features.

The rest of this chapter is organized as follows. In Section 4.2 we describe the scientific contribution of our work. Section 4.3 presents the application scenario consisting in information spotting in document image databases. Experimentally, we illustrate the performance of the approach in different databases, namely graphical documents and handwritten ones. Finally Section 4.4 draws the conclusion and the main difficulties faced by this methodology.

4.2 Binary Embedding Formulation

This section describes the proposed binary embedding formulation which consists in encoding the local topological context of graph nodes with binary vectors. It allows to construct a fast indexing scheme in terms of the Hamming distance. Figure 4.1 shows the pipeline that follows the proposed system. First, the binary embedding is computed and afterwards the indexation is applied.

4.2.1 Binary Topological Node Features

The *Morgan index* M of a graph $g \in \mathcal{G}$ is a node feature, originally used to characterize chemical structures [157], that computes the node context in terms of its local neighborhood.

Definition 4.2.1 (Morgan Index)**.** Given a graph $g = (V, E, \mu, \nu)$, the *Morgan*

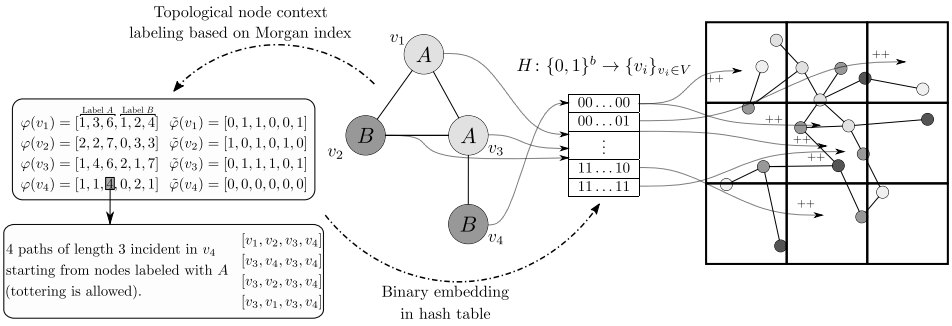


Figure 4.1: Overview of the whole system.

index $M(v,k)$ of k-th order of a node $v \in V$, counts the number of walks of length k incident to node v and starting somewhere in the graph. This index is iteratively computed as follows

$$M(v,k) = \begin{cases} 1, & \text{if } k = 0 \\ \sum_{u \in \mathcal{N}(v)} M(u, k-1) & \text{otherwise.} \end{cases} \tag{4.1}$$

Observe that the Morgan index can be computed using the values obtained from the exponentiation of the adjacency matrix A. An interesting property of the adjacency matrix A of any graph g is that the (i,j)-th entry of A^n denotes the number of walks of length n from the node v_j to the node v_i. Therefore, the Morgan index of order k from a node v_i is equivalent to the sum of the elements on the i-th row of the matrix A^k, formally

$$M(v_i,k) = \sum_{j=1}^{|V|} A^k(i,j). \tag{4.2}$$

Following the Definition 2.4.1, an embedding function $\varphi\colon \mathcal{G} \to \mathbb{R}^n$ transforms a graph $g \in \mathcal{G}$ to an n-dimensional feature vector. Hence, the distance between two graphs is computed by a distance in a given metric space, and the problem of graph classification is solved by a statistical learning approach in a faster way. Inspired by the topological node features proposed by Dahm *et al.* [52], we define the *local context* of a node v as a node embedding function computed in terms of the topological information of a sub-graph centered at v and considering those nodes at most at a distance of k hops. Figure 4.2 shows the considered subgraph, in green, when defining the *local context* of the node v, with $k = 3$. We propose to describe this local context in terms of the Morgan index enriched with the information of the labels of the neighboring nodes.

Firstly, let us explain a variation of the Morgan index concept. Afterwards, we will define our node embedding function. Given a graph $g = (V, E, \mu, \nu)$, let us

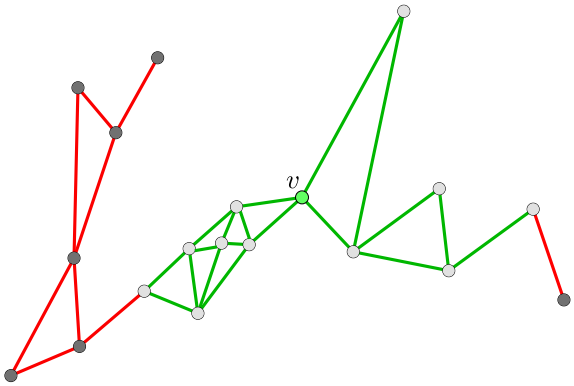


Figure 4.2: Local context of v (in green) of length $k = 3$.

denote as $M_l(v, k)$ the Morgan index of node $v \in V$, order k and label l which counts the number of walks of length k incident at node v and starting at nodes labeled as l. According to this definition, we formally define the *local context* of a node v as

$$\varphi(v) = \Big\|_{k=1}^{K} \left(\Big\|_{l \in L_V} M_l(v, k) \right), \tag{4.3}$$

where $\|$ is the concatenation operation and K is the maximum length of the walks incident to v that are considered. The value of K is dependent of each experimental set-up and defines the size of the context considered per each node. Thus, every graph node is associated to a $K \cdot |L_V|$ dimensional feature vector characterizing the number of walks incident to v of lengths up to K and starting at nodes labeled with the different possibilities in L_V. Remember that according to Definition 2.1.1, L_V is the finite label set for nodes.

Afterwards, the context vector $\varphi(v)$ is converted to a binary code defined as $\tilde{\varphi}(v) = \{0, 1\}^{K \cdot |L_V|}$ in terms of a list of corresponding threshold values $T = \{t_i\}_{i=1}^{K \cdot |L_V|}$. These values are application dependent, and in the use case described in Section 4.3 are set to the mean per each dimension. This binary code formulation, instead of $\varphi(v)$, speeds up the indexation process with some loss of information.

Figure 4.3 illustrates the computation of the binary codes. In this example, we have used $L_V = \{A, B\}$ and $K = 3$, hence, the codes associated to nodes have length 6 ($|L_V| = 2$). The threshold values are set to the mean of each value $M_l(v_i, k)$ for all $v_i \in V$, therefore, $T = [\frac{5}{4}, \frac{10}{4}, \frac{33}{4}, \frac{3}{4}, \frac{8}{4}, \frac{15}{4}]$.

Until now, we have defined the binary vectors to encode the information of the topology around a node v. However, node v itself has not been codified. To solve this, we optionally compute the walks of length 0. It is interpreted as adding a flag to the vector indicating the label of the node. Figure 4.4 shows the binary code computation from the graph of Figure 4.3 adding the label information of v.

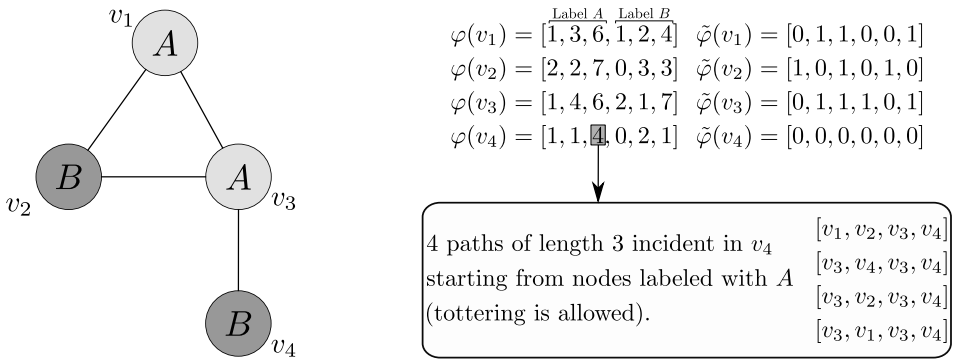


Figure 4.3: Example of the binary code computation from a labeled graph.

$$\begin{aligned}
\varphi(v_1) &= [\overbrace{1,1,3,6}^{\text{Label } A}, \overbrace{0,1,2,4}^{\text{Label } B}] \;\; \tilde{\varphi}(v_1) = [1,0,1,1,0,0,0,1] \\
\varphi(v_2) &= [0,2,2,7,1,0,3,3] \;\; \tilde{\varphi}(v_2) = [0,1,0,1,1,0,1,0] \\
\varphi(v_3) &= [1,1,4,6,0,2,1,7] \;\; \tilde{\varphi}(v_3) = [1,0,1,1,0,1,0,1] \\
\varphi(v_4) &= [0,1,1,4,1,0,2,1] \;\; \tilde{\varphi}(v_4) = [0,0,0,0,1,0,0,0]
\end{aligned}$$

Figure 4.4: Example of the binary code computation from the same graph from Figure 4.3 adding walks of length 0.

The proposed embedding takes into account walks starting and ending in the same node. These walks may add some redundant and noisy information. The fact of not considering such cyclic walks, generates a new embedding formulation. Thus, let us denote as $\hat{M}_l(v,k)$ the Morgan index of node v, order k and label l which counts the number of walks of length k incident to node v starting at nodes labeled as l but disregarding those ones starting at v, *i.e.* cycles. Then we optionally allow to use this new definition on our embedding computation. Following the example of Figure 4.3, Figure 4.5 shows the new embedding generated by discarding the cycles during the Morgan Index computation. The different embedding variants will be compared in the experimental section.

$$\begin{aligned}
\varphi(v_1) &= [\overbrace{1,1,4}^{\text{Label } A}, \overbrace{1,2,4}^{\text{Label } B}] \;\; \tilde{\varphi}(v_1) = [0,0,0,1,1,1] \\
\varphi(v_2) &= [2,2,7,0,1,1] \;\; \tilde{\varphi}(v_2) = [1,1,1,0,0,0] \\
\varphi(v_3) &= [1,1,4,2,1,7] \;\; \tilde{\varphi}(v_3) = [0,0,0,1,0,1] \\
\varphi(v_4) &= [1,1,4,0,1,1] \;\; \tilde{\varphi}(v_4) = [0,0,0,0,0,0]
\end{aligned}$$

Figure 4.5: Example of the binary code computation from the same graph from Figure 4.3 disregarding cyclic walks.

4.2.2 Indexing

Given the node embedding $\tilde{\varphi}(\cdot)$, we propose an indexation scheme based on the concept of *focused graph retrieval* previously introduced. In terms of a visual retrieval application, this process is understood not only retrieving the images of a database where a query object is likely to appear, but finding a coarse position in each retrieved image.

An inverted file indexing architecture, in terms of node contexts, is constructed. It stores a mapping from the binary topological features to the nodes of the target graphs in the database. This inverted file is therefore formulated as a lookup table $H\colon \{0,1\}^b \rightarrow \{v_i\}_{v_i \in V}$ that indexes a b-bit vector and returns a list of nodes whose context, defined as the binary code provided by $\tilde{\varphi}(\cdot)$, is similar to the input code.

The last step is the actual (sub)graph matching process. With the indexing table H we only retrieve individual nodes, so it is necessary to implement a node consistency verification. With this purpose, we define a *partition* P of a graph g as a decomposition of it in n small (sub)graphs, $P(g) = \{g_1, \ldots, g_n\}$, where $g_i \subseteq g$. Hence, the lookup table H is reformulated as a hashing function that instead of returning nodes similar to the input binary code, returns (sub)graphs where these target nodes appear. Formally, given a query graph $g_q = (V_q, E_q, \mu, \nu)$ and a database of graphs $\{g_1, \ldots, g_T\}$, for each node of the query graph $v \in V_q$, the indexation function H returns the (sub)graphs of the database, containing this vertex $H(v) = \{g'_{q_1}, \ldots, g'_{q_n}\}$, where g'_{q_i} are n-(sub)graphs of the target graphs $\{g_1, \ldots, g_T\}$ contained in the partitions $\{P(g_1), \ldots, P(g_T)\}$. The definition of the partition under which the database of graphs is decomposed in small graphs is application dependent. In our indexation framework, the (sub)graphs g_i defined by the partition P are seen as voting bins, according to a Hough-based principle. Thus, the final result consists of the (sub)graphs receiving a high number of votes.

Concerning the practical implementation of H, the similarity between binary codes is computed using the Hamming distance. The most straightforward solution is a brute-force linear scan, *i.e.* to compute the Hamming distance between the query vector and each vector in the database. Computing the Hamming distance between two vectors consists in computing the XOR and counting the number of 1's in the resulting vector. This is computed very fast on modern CPU's, with logic operations being part of the instruction set. A fast hashing process like *Locality Sensitive Hashing* (LSH) [109] can be added to speed up the final indexation.

4.3 Experimental Validation

This section is devoted to carefully validate the performance of the proposed graph indexing approach. The proposed framework is validated in different tasks in order to provide an overview of its strengths.

4.3.1 Experimental Setup

Three experiments have been performed to test the proposed approach. The first one consists in a graph comparison problem in order to show its effectiveness as a fast pre-processing step to discard true negatives without losing many true positives. The next two experiments evaluate the indexation approach in a real scenario where the graphs appear with several deformations. Moreover, these real scenarios evaluate our method in the subgraph matching problem. Hence, the selection of candidates avoids a computationally expensive subgraph matching calculation between a small query graph and very large target graph. The second experimental case is addressed to show the efficiency in a subgraph isomorphism scenario. It consists in locating graphical symbols in architectural floor plan images. Due to its line-drawing format, these documents are usually represented by graphs, and hence symbols are located using a subgraph isomorphism. The third case continues the word spotting experiment in handwritten document images presented in the previous chapter. The three experimental cases are presented in an incremental order of difficulty. The first one consists of retrieving individual graphs from a database. In the second one, graph indexing involves subgraph isomorphism where the geometric attributes of the queries are compared with rotation and scale tolerance. In the last one the graphs are distorted because of the noise and deformations that is present in handwritten text. For the experiments based on spotting, the objective is to find a good trade-off between the loss in performance and the speed-up in terms of time.

The evaluation proposed in this work uses the classic metrics used in the context of information retrieval scenarios [190].

Let ret be the set of retrieved elements from the dataset and rel is the set of the relevant objects with regard to the query. Let *True Positive* (TP) be the set of (correctly) relevant retrieved elements ($|\,\mathrm{ret} \cap \mathrm{rel}\,|$), *False Positive* (FP) be the set of incorrectly retrieved elements ($|\,\mathrm{ret} \cap \overline{\mathrm{rel}}|$), *True Negative* (TN) be the non relevant elements that have not been retrieved ($|\overline{\mathrm{ret}} \cap \overline{\mathrm{rel}}|$) and *False Negative* (TN) be the relevant elements that have not been retrieved ($|\overline{\mathrm{ret}} \cap \mathrm{rel}\,|$).

Three metrics have been chosen for the performance evaluation of the graph indexing framework, *Precision*, *Recall* and *Specificity*. *Precision* (P) is the probability that a (randomly selected) retrieved element is relevant; *Recall* (R) is the probability that a (randomly selected) relevant element is retrieved. Finally, *Specificity* (S) is the probability that a non relevant element is properly identified.

$$\text{Precision} = \frac{|\,\mathrm{ret} \cap \mathrm{rel}\,|}{|\,\mathrm{ret}\,|} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{|\,\mathrm{ret} \cap \mathrm{rel}\,|}{|\,\mathrm{rel}\,|} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{|\overline{\text{ret}} \cap \overline{\text{rel}}|}{|\overline{\text{ret}}|} = \frac{TN}{TN + FP}$$

In addition, the *mean Average Precision* (mAP) introduced in the previous chapter has been used as the performance metric of the proposed word spotting approach.

4.3.2 Graph Classification

Nowadays, graph-based representations have experienced an increase usage in pattern recognition due to their power to represent objects keeping the information of its structure. Statistical approaches that represent the objects using feature vectors will fail in those cases where the structure is the driver of the main information. The IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning [180] presents ten graph sets with different characteristics. Figure 4.6 shows two examples from two distinct classes. These sets come from real problems that are faced through graph-based representations.

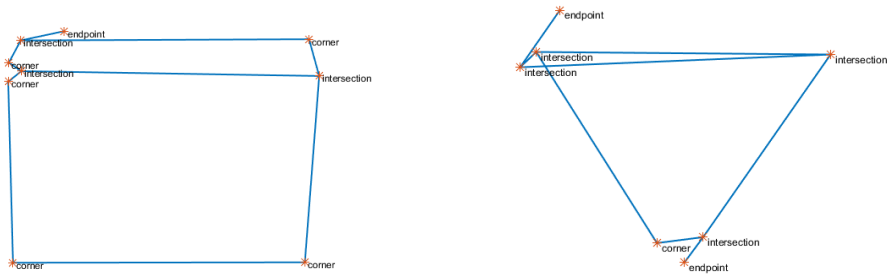


Figure 4.6: Examples of graphs from two classes of the dataset.

For this evaluation we have chosen the *GREC* subset as it has discrete labels on the nodes. This dataset consists of graphs representing symbols from architectural and electronic drawings. These graphs are divided in 22 classes and grouped in three subsets, on the one hand, the training and validation sets of size 286 each, and on the other hand, a test set of size 528. Graph nodes represent *end points*, *corners*, *intersections* and *circles* of the original symbol. Moreover, they are labeled with a two-dimensional attribute giving their position. Edges are labeled as *line* or *arc* and an additional attribute specifies the angle with respect to the horizontal direction or the diameter in case of arcs.

Table 4.1 shows the results of our graph indexing approach on the *GREC* dataset using different configurations in terms of absolute number of true positives, false positives, true negatives, false negatives and recall. First, we have fixed the local context size (LC) which corresponds to walks of length up to 3. In addition, each configuration is represented by node flag (NF) a binary attribute that considers 0-length paths or not and avoid cycles (AC) that disregards cyclic walks or not.

From this table, we observe that the highest recall is achieved considering 0-length paths and cyclic paths. However, avoiding cyclic paths leads to a more restrictive scenario where more TN are detected without losing too much recall.

Table 4.1: Comparison of the four proposed embeddings with a fixed size of the local context ($LC = 3$).

Configuration		Metrices				
AC	NF	TP	FP	TN	FN	Recall
-	-	6090	62531	81613	774	88.72%
-	✓	**6222**	56914	87230	**642**	**90.65%**
✓	-	5980	59461	84683	884	87.12%
✓	✓	6065	**49074**	**95070**	799	88.36%

Considering the best configuration in terms of AC and NF, Table 4.2 shows a comparison changing the size of the local context. From the table, we conclude that changing the size of the local context will cause our approach to be more restrictive. Hence, recall measure will decrease while avoiding false positives. This parameter should be set depending on the problem to be solved.

Table 4.2: Embedding performance using several configurations.

LC	TP	FP	TN	FN	Recall
2	6481	86928	57216	383	94.42%
3	6222	56914	87230	642	90.65%
4	5941	49909	94235	923	86.55%
5	5765	43956	100188	1099	83.99%

Table 4.3 shows a comparison between a state-of-the-art graph matching method [133] and the same approach but previously filtering the graphs with the proposed indexation framework. In addition, the results are provided according to different sizes of the local context (LC). To compute the distances between graphs we have used the software provided by Riesen *et al.* with the settings reported in [133]. The classification rate is obtained using a k-nearest neighbors (k-NN) classifier where the number of neighbors value ranges between 1 and 9. The indexation approach uses the same configuration as Table 4.2. It must be noticed that our analysis reports on how the proposed graph indexation keeps the performance when it is applied before the baseline graph matching.

Nonetheless, we notice that the best performance is achieved is using 3-NN. In addition, we keep a similar classification rate applying the proposed indexation. As it is observed, when the indexation uses a higher local context size, the best performance is achieved considering only 1 neighbor.

Table 4.3: Comparison in terms of classification rate.

k-NN	Original [133]	LC 2	LC 3	LC 4	LC 5
1	0.9867	0.9867	0.9867	0.9830	0.9830
3	0.9924	**0.9905**	0.9867	0.9697	0.9602
5	0.9811	0.9697	0.9545	0.9394	0.9186
7	0.9621	0.9413	0.9356	0.9186	0.8864
9	0.9489	0.9299	0.9129	0.8996	0.8409

In terms of time, the original setting takes 182.37 seconds to make 151,008 comparisons. Using $LC = 2$ these comparisons have been reduced to 93,409 (more than 42% reduction) that in average takes 112.58 seconds. For the indexation step takes 30.55 seconds without optimization using Matlab. Therefore, we have reduced the time in almost 39 seconds even though the dataset is not very large and the graph matching is developed in Java which is faster than Matlab.

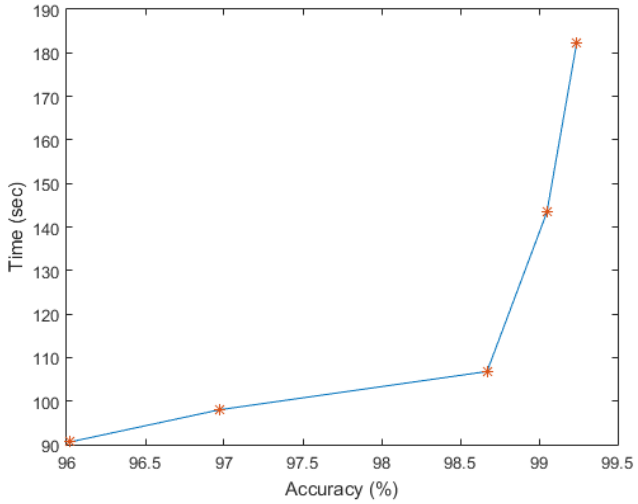


Figure 4.7: Accuracy vs time comparison.

Figure 4.7 shows a plot according to the obtained accuracy and its corresponding computation time. Observe that the computation time is drastically reduced while obtaining an acceptable loss of performance in terms of accuracy.

4.3.3 Architectural Symbol Spotting

Nowadays, one of the main interests is information spotting and retrieval in large document collections. This task is defined as detecting the query information in a document collection without explicitly recognizing these documents. In this scenario, one of the most challenging application field is searching and browsing symbols in graphical documents, known as symbol spotting.

Several structural representations have been proposed for symbol spotting in graphical documents, such as architectural floor plans. Although there are some string-based representations, *e.g.* Rusiñol *et al.* [191] encodes symbols using attributed strings, most of the approaches use graph-based representations. For example, Dutta *et al.* [64] proposed a graph serialization by computing acyclic graph paths; Luqman *et al.* [147] proposed to encode graphs by using fuzzy histograms; and Dutta *et al.* [63] proposed a dual edge graph representation.

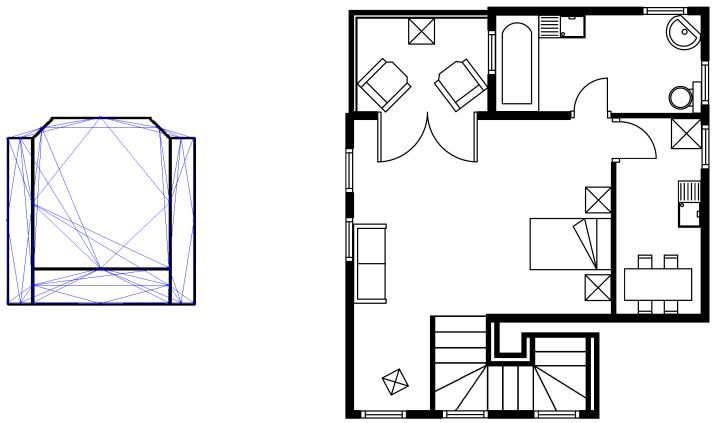
This experiment is an extension of the previous one. The main difference is that now, the symbols appear in their real context, therefore, a subgraph matching is required.

We use the dual graphs proposed by Dutta *et al.* [63], which are constructed from the SESYD dataset, created by Delalandre *et al.* [55]. This dataset contains 10 different subsets and 16 query symbols. Figure 4.8 shows two example images from the SESYD dataset. Each one of the subsets contains 100 synthetically generated floor plans. All the floor plans in a subset are created from the same floor plan template by placing different model symbols in different locations in random orientation and scale. For our experiments, we have taken 20 graphs from each subset and all the query symbols. Dual graph nodes are labeled with the vector of Hu moments invariants. Hence, we must discretize it applying a clustering technique such as k-means (for this experiment we selected 7 clusters). Dual nodes represent a path in the vectorized image.

As we see in Figure 4.8(a), this representation highly depends on the vectorization. Hence, the graphs differ from one symbol instance to another one, especially when it is contained in a floor plan image. Architectural floor plans are difficult datasets for indexing approaches such as the one proposed in this chapter. On the one hand, symbols of different classes are very similar and difficult to distinguish unless the context is used. On the other hand, the fact that symbols appear connected to building elements such as walls introduces a high degree of distortion (see Figure 4.9).

In more detail, first, we apply the indexation framework using the graph of a query symbol against the graph of a floor plan. Second, we obtain a set of votes in different nodes using a threshold based on the Hamming distance. Third, these votes are casted in a partition arranged in a grid (of size 100) that is used to extract the candidate regions.

To decide that one cell of the grid is part of the symbol, it must contain a minimum



(a) **(b)**

Figure 4.8: Examples of the elements in our database. (a) graph representation of a query; (b) floor plan image where the query should be spotted.

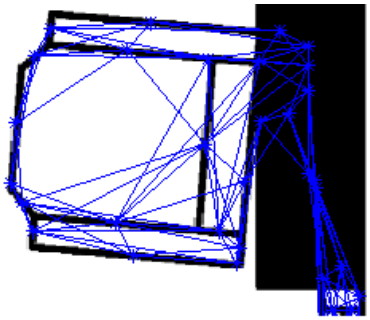


Figure 4.9: Examples of wall problems.

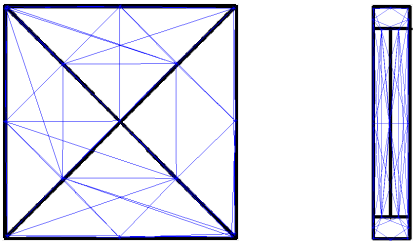
number of votes. This minimum value is set to the 10% of the number of nodes of the query graph. Then, the image is divided into candidate regions and the bounding box is computed. The isolated cells are discarded because they are too small to contain the query symbol. To consider as true positives the retrieved image regions in comparison to the ground truth regions, we consider an overlapping of minimum 25%. Note that we aim for a coarse detection rather than to an accurate matching.

Table 4.4 shows the performance of our approach. The threshold indicates that the method votes those nodes with a Hamming distance lower than this threshold value. Here we are facing a subgraph matching problem, so the threshold will change with respect the previous experiment. These results have been computed using a local context of size 3, counting the cyclic paths and the 0-lengths paths. By

Table 4.4: Comparison of the embedding performance for floor plans.

Threshold	Precision	Recall
5	12.62%	49.02%
6	10.82%	49.34%

changing the threshold, the precision decreases whereas the recall rapidly increases. The results report that this application scenario presents several difficulties to be solved with the proposed method. First, symbols tend to be adjacent, and the topology is affected. Furthermore, the results are sensitive to the thresholds set beforehand. Some symbols get too few votes, but if we increase the threshold, other symbol queries would obtain many false positives detecting big regions.



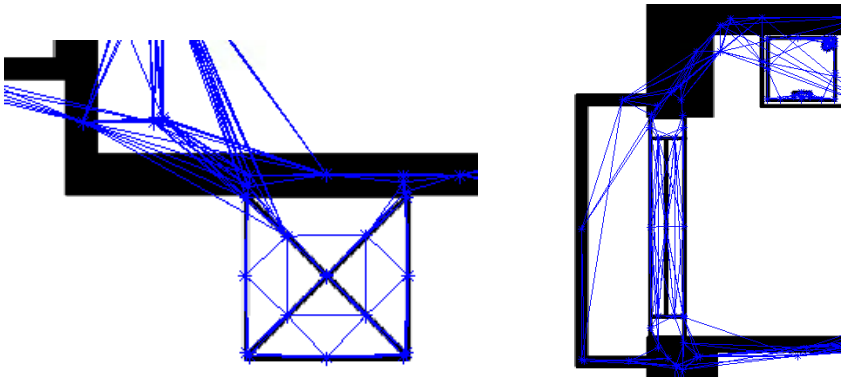
(a) **(b)**

Figure 4.10: Examples of elements that are not correctly detected.

Figure 4.10 shows two problematic symbols that clearly influence the results. These symbols are symmetric and they usually appear next to walls or other symbols that makes their topology to change a lot. Figure 4.11 shows the same symbols in their context. As expected, they appear next to the walls, and their topology is different form the query ones. If we exclude these symbols, the performance highly increases, as it is shown in Table 4.5.

Table 4.5: Comparison of the embedding performance for floor plans without the two problematic query symbols.

Threshold	Precision	Recall
5	13.06%	67.39%
6	11.29%	68.89%



(a) (b)

Figure 4.11: Examples of elements that are not correctly detected in their context.

4.3.4 Handwritten Word Spotting

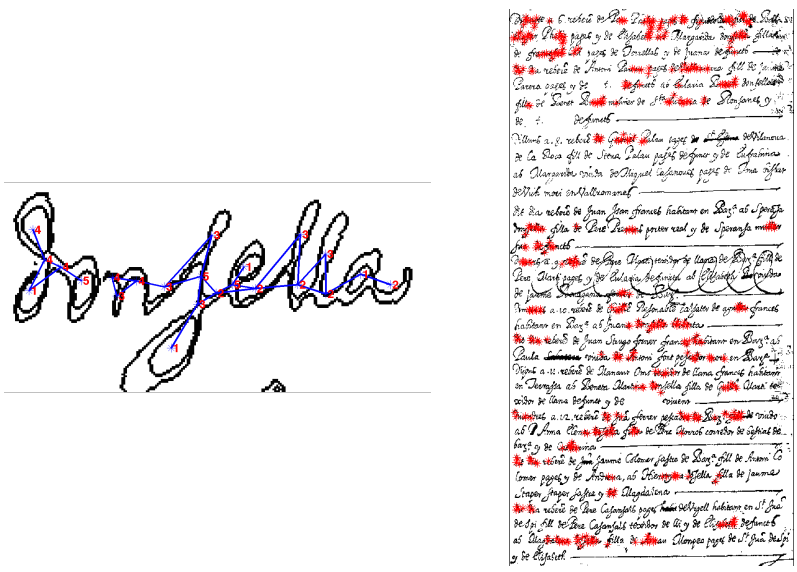
The third experiment is contextualized in to the area of digital humanities, in particular the preservation of historical manuscripts stored in archives, libraries and museums. Once large amounts of documents are digitized, the challenge is the extraction of information for consultation purposes. Full transcription using handwriting recognition techniques is not always feasible nowadays because of the variability of the text styles, the bad physical preservation of the sources, and because handwriting recognition techniques require large amounts of annotated images to train, not always available. Word spotting, introduced in the previous chapter, is an alternative for content indexing. Word spotting is the task of retrieving the instances of a given query word from a document collection. It is usually formulated in terms of a visual object detection problem, where the query word and the image words are represented by features invariant to visual distortions.

In the comparison of statistical versus structural representations for handwritten words reported in [142], the main disadvantages of structural approaches are the time complexity and scalability to large collections. Methods such as [81] only use the graph nodes (no edges), and [224] proposed an embedding using a bag of graphlets (codebook of small graphs, with order 2 or 3). However these approaches are still not able to efficiently cope with large databases while preserving the graph structure. Our hypothesis is that the proposed method based on graph indexing with binary embeddings is of key importance to make structural approaches comparable to statistical ones in terms of time and memory requirements for large document collections.

In our experiments, we have used images from a collection of marriage records from the Archive of the Barcelona Cathedral. The use of word spotting in this collection

allows to search names, places, occupations, etc. To have a clear evaluation of the indexation system, differently from the previous chapter, we have use the whole dataset named, *The Barcelona Historical Handwritten Marriages Database* (BH2M) [76], which consists of 174 pages from the 17th century. The dataset is divided in 3 sets, train, validation and test with 100, 34 and 40 page images respectively. The ground truth consists of the bounding box and transcription of each word. We used 5170 query words, cropped from these manuscripts. Here, the word images are represented by attributed graphs where nodes correspond to basic primitives (graphemes) like loops, vertical lines, arcs, etc. and edges represent adjacency relations between primitives. This graph representation has been proposed in the previous chapter.

The experimental case consists in using the graph indexing scheme to retrieve instances of a given query word. The partition of the target graphs consists in the subgraphs corresponding to the segmented words. Thus, the votes are accumulated in the image regions containing words likely to be the query one. Then, we perform a more accurate search in a partition of the graph around these regions. The true positives are those regions that contain the correct word and have at least a minimum amount of votes.



(a) **(b)**

Figure 4.12: Qualitative results of the indexation scheme on a whole page. (a) a query word and its corresponding graph; (b) a full page and the locations where query nodes are detected.

Table 4.6 shows the performance on the BH2M database. There, the specificity and recall are provided, specificity evaluates how many elements are properly discarded

whereas recall takes care of how many relevant elements are considered for the matching stage. We have used the best configuration among the ones proposed in subsection 4.3.2, and show the recall and specificity. For this approach, we are not focusing on the precision but on the rejection of incorrect words in a fast way. The threshold used in these experiments is 10 and the bins (bounding boxes of words) must contain at least n votes to consider them as a positive result.

Table 4.6: Performance comparison on the BH2M database, changing the minimum number of votes for accepting a bounding box.

Min. num. of votes	**Specificity**	**Recall**
2	36.17%	93.06%
4	58.53%	80.42%

This table shows that by increasing the minimum number of votes, the specificity increases, although the recall falls down. This is because there are words of different sizes, so if we search a graph with only 5 or 6 nodes it is very difficult to get enough votes in these regions. To solve that, we normalize the number of votes adapting it depending on the quantity of nodes in the query. Let us define this value as $|V_q| \cdot x$ where x is a parameter dependent on the application. The results changing this parameter are presented in Table 4.7, using the same configuration and threshold. We correctly detect 50.98% and 61.54% respectively of the non relevant elements whereas we only miss 10.39% and 15.87% of the relevant ones.

Table 4.7: Performance comparison on the BH2M database according to the cutting value to accept a bounding box.

x	**Specificity**	**Recall**
0.15	50.98%	89.61%
0.20	61.54%	84.13%
0.25	70.00%	77.15%

Despite the graph variability in handwritten words, our proposed indexation discard more than half of TN without a significant decrease in the TP. Hence, is could be used as a pre-processing step for speeding-up graph-based word spotting techniques without decreasing the performance.

Table 4.8 shows the word spotting performance in terms of *mean Average Precision* and the percentage of required comparisons. The performance has decreased 4 and 7 points with $x = 0.15$ and $x = 0.20$ respectively regarding the word spotting that performs all comparisons. In this retrieval scenario, it is usually required to detect the top K similar images, hence having a small drop in our performance is not a big issue. However, we have avoided more than half of the comparisons. Hence, it is a good technique for time saving. Indeed, the computation time is reduced

Table 4.8: Performance comparison of word spotting on the BH2M database.

Method	mAP	Comparisons
Original [177]	67.57%	100%
+Indexation $x = 0.15$	63.27%	49.19%
+Indexation $x = 0.20$	60.19%	38.65%

using our indexation approach. Taking 100 random queries and comparing each query against all the words in the 40 pages of test, it takes 1841.47 seconds. The comparison between word graphs is performed with a C implementation of the assignment edit distance proposed in [181]. The indexation process (implemented in Matlab without being optimized) for the same queries and pages, takes only 325.47 seconds to find the candidates to match and it computes only 52.05% of comparisons. Hence the time for comparisons is reduced to 1,149.36 seconds.

4.4 Conclusions

In this chapter, we have presented an approach for computing a fast indexation to speed-up the inexact subgraph matching process for large scale retrieval purposes. The main contribution of the proposed approach is the definition of a binary embedding for graph nodes based on the local context. The local context of a node $v \in V$ has been defined as the topology of the paths of order k incident in the node v and coming from nodes of a given label. Then, a hashing architecture has been designed using binary codes as indexation keys in terms of Hamming distance.

Several experiments show the performance of the binary embedding in real scenarios involving complex graphs. First, a validation using a well-known database of small graphs has been done. We could obtain a relevant decrease of comparisons needed for next steps, showing a big potential to quickly discard non-relevant elements (*i.e.* detect true negatives). Second, the application to information spotting such as symbol detection in floor plans showed encouraging results, although specific techniques for dealing with symmetric touching symbols should be investigated. Finally, it has been applied to word spotting in historical handwritten documents. In terms of a retrieval problem, high recall values are obtained, with a specificity higher than 50% in most cases. High recall values allow to identify relevant items whereas the specificity shows that the non-relevant ones are correctly identified. In cases where there are more non-relevant than relevant items, it discards more than 50% of false positives without missing many relevant items. So, it could be applied as a pre-processing step before using a more accurate technique such as graph edit distance.

The time complexity of the indexation step is linear in terms of the number of

nodes in the database. It leads us to conclude that our graph indexation scheme is very useful to compute inexact subgraph matchings in large-scale scenarios as a filtering step for pruning the database, before using a more accurate matching method only in the retrieved subgraphs. Finally, in terms of the application, we have demonstrated that compact structural descriptors are useful signatures for handwriting recognition, despite the variability of handwriting.

The main limitation of this approach comes from the noise that spurious nodes may generate. Even though it should be mitigated with the binarization scheme, this is not true if the noise present in the data is important. A possible solution to deal with distortioned graphs is to find an abstract representation which removes these problematic distortions.

5 | Hierarchical Representation for Robust Matching

The only way of discovering the limits of the possible is to venture a little way past them into the impossible.

– Arthur C. Clarke

With the aim to cope with both the time complexity and the variability, deformations and noise present in the generated graphs, in this chapter we propose to construct a novel hierarchical graph representation. The previous chapter exploits the local node neighborhood leading to local binary embeddings which are not able to capture the global connectivity of the graph. Graph clustering techniques adapted from social media analysis have been used in order to iteratively contract a graph at different abstraction levels while keeping information about its topology. Abstract nodes attributes summarize information about the contracted graph partition. For the proposed representations, a coarse-to-fine matching technique is defined. Hence, small graphs are used as a filtering before more accurate matching methods are applied. This approach has been validated in real scenarios such as image classification or retrieval of handwritten words, i.e. *word spotting.*

5.1 Introduction

As reported in previous chapters, graph-based representations leads to high computational complexities usually dealt by graph embeddings, approximated matching techniques or graph indexing approaches. However, despite their representational power, graphs representations are very sensitive to noise and small variations. Moreover, as it has been seen in the previous chapter, the proposed indexing framework takes local node embeddings which are not able to capture the global graph structure.

Despite the efforts trying to reduce the time complexity of the matching frameworks, they are only suitable for quite small and restricted scenarios, in the number of instances and size of the graphs. For instance, the previous chapter indexation strategies have been proposed to prune the number of graph comparisons performed by these techniques, however, indexation methods rely on local structures rather than global knowledge of the graph. Therefore, it results in the retrieval of

many false positive responses due to the lack of a global similarity measure. An interesting strategy to address this drawback is to use a graph scale-space approach where the input data is hierarchically organized, summarizing it in order to prune complex graph comparisons. Hence, while fine grained scales describe the detailed structures, coarse scales describe the global ones.

Processing information using a multi-scale representation is successfully employed in computer vision and image processing algorithms, which is mostly inspired by its resemblance with human visual perception [1]. It is observed that a naturalistic visual interpretation always demands a data structure able to represent scattered local information as well as summarized global facts [113]. Hierarchical representation is often used as a paradigm to efficiently extract the global information from the local features. Apart from that, hierarchical models are also believed to provide time and space efficient solutions [215]. Motivated by the above mentioned intuition and the existing works in the related fields, many authors have come up with different hierarchical graph structures for solving various problems [73, 74, 151, 215]. In this sense, it is worth to mention the work of Mousavi *et al.* [158], who presented a hierarchical framework for graph embedding, although they did not explore the complex encoding of the hierarchy itself.

The main contribution of this chapter is the proposal of a hierarchical graph representation and its associated matching technique able to discard non-promising structures in a coarse-to-fine fashion. Following the scale-space principle, subgraphs sharing some properties at level i are contracted in a single node at level $i+1$. Thus the structural information for each level, characterized by the graph topology and the corresponding attributes, flows through the hierarchical edges to the contracted node. Our hierarchical information avoids a direct and costly matching at the original graph level, performing first the comparisons at abstract levels of the hierarchy, speeding up the process. The hierarchy is designed to perform a big reduction of the graph size leading to a drastic reduction of the matching time at the new levels. The proposed approaches are experimentally validated using real scenarios such as classification of color images and handwritten word spotting. As far as we know, this is the first time where hierarchical representations have been used as a pruning mechanism for efficient large-scale graph retrieval.

To sum up, let us place our work into the current literature. First, graph matching has not taken into account the hierarchical graph information in order to speed-up the matching of two graphs. Second, several indexation frameworks have been proposed, however, they cope with local graph information instead of obtaining global representations. Third, the existing hierarchical graph representations usually propose a change of the graph scale rather than obtaining a real abstraction. Finally, the proposed approach is learning free, therefore, our method does not require any training data. In the next sections we will describe our proposed methodology, which tries to solve the above mentioned limitations.

The rest of the chapter is organized as follows. Section 5.2 formalizes a hier-

archical graph representation that is used in the rest of the work. Afterwards, Section 5.3 proposes a coarse-to-fine matching technique able to deal with the previous representation. Section 5.4 is devoted to evaluate the proposed techniques and to illustrate the performance in a real scenario. Finally, Section 5.5 draws the conclusions and outlines the limitations of the current approach.

5.2 Hierarchical Attributed Graph Representation

An attributed hierarchical graph model with the ability of summarizing the relevant features in a compact way is able to increase the information present in the original graph including scale and abstract features. For the sake of simplicity, let us firstly reformulate the definition of *hierarchical graph* introduced in Definition 2.5.1. With abuse of notation,

Definition 5.2.1 (Hierarchical Graph)**.** A *hierarchical graph* h is defined as a 6-tuple $h = (V, E_n, E_h, \mu, \nu_n, \nu_h)$ where V is the set of nodes; $E_n \subseteq V \times V$ are the neighborhood edges; $E_h \subseteq V \times V$ are the hierarchical edges; μ, ν_n and ν_h are three labeling functions defined as $\mu \colon V \to L_V$, $\nu_n \colon E_n \to L_{E_n}$ and $\nu_H \colon E_h \to L_{E_h}$, where L_V, L_{E_n} and L_{E_h} are three sets of attributes or labels for vertices and edges, respectively.

5.2.1 Hierarchical Construction

A hierarchical graph representing information at different abstraction and scale levels allows to perform a huge variety of tasks taking into account such information. In particular, abstract levels encode a high level information that allows to perform a coarser version of the task. Note that for each level, the graph is reduced in terms of the number of nodes and edges, therefore, time complexity on those graphs is drastically reduced.

The hierarchical graph construction consists of three stages. First, to find groups of nodes representing graphlets which encode highly related information, namely graph clustering; second, to create a new node which encodes the graphlet information and its hierarchical edges defining the relation between abstract levels; and finally, to create neighborhood edges between the new nodes according to the connections in the previous level. Therefore, given a graph $g = (V, E, \mu, \nu)$ we define two functions to construct their corresponding hierarchical representation $h = (V', E'_n, E'_h, \mu', \nu'_n, \nu'_h)$.

- *Embedding:* The embedding function is a signature of the subgraph that summarizes the information from one level to another. Let $\varphi \colon \mathcal{G} \to \mathbb{R}^n$ be the embedding function, it returns a vectorial representation of a given graph. The embedding φ enables the information propagation between levels, from the original graph to its abstract representations. This function

codifies both, topological connections and node attributes. The embedding function is application dependent and it is specified for each particular case in Section 5.4. Note that the subgraphs that will be contracted will be small in terms of the number of nodes. As shown by Dutta *et al.* [66] small subgraphs are accurately represented by really simple embedding functions. However, there is no restriction in which embedding function should be used [38].

- *Contraction:* The graph contraction function drives the hierarchical graph generation. Let $c_\varphi \colon \mathcal{G} \to \mathcal{H}$, where $\mathcal{H}$ is the space of hierarchical graphs, be the contraction function. It defines a hierarchical graph given a graph g and a fixed embedding function φ. This function finds groups of nodes that are gathered together into one node in the next level. The contraction process can follow different criteria such as topology, features of the nodes or edges, etc. In this work, we propose to use a classic graph clustering technique that has been widely used for community detection. However, other techniques may be used depending on other needs.

Algorithm 5.1 proposes a general pipeline to construct a hierarchical or pyramidal graph given an input graph g. In this algorithm, L is the number of hierarchical levels of the output graph h and ε and δ are the contraction and connection ratio respectively. Moreover, the ConsiderAsVertex and ClusterGraph functions are the embedding and clustering functions previously introduced, which are application dependent.

Algorithm 5.1 Hierarchical Graph Construction given an input graph g

Input: $g = (V, E)$, L, ε, δ
Output: $h = (V, E_n, E_h)$

$h \leftarrow g$; $g_\mathrm{c} \leftarrow g$
for $i = 0$ to L **do**
  $K = \lfloor \varepsilon \cdot |g_\mathrm{c}.V| \rfloor$
  $\{c_0, \ldots, c_{K-1}\} \leftarrow \textsc{ClusterGraph}(g_\mathrm{c}, K)$
  $g_\mathrm{n}.V = \{\textsc{ConsiderAsVertex}(c_j) : j = \{0, \ldots, K-1\}\}$
  for $(u, v) \in g_\mathrm{n}.V \times g_\mathrm{n}.V$ **do**
    if $\textsc{ConnectionRatio}(u, v) > \delta$ **then**
      $g_\mathrm{n}.E \leftarrow g_\mathrm{n}.E \cup (u, v)$
    end if
  end for
  for $j = 0$ to $K - 1$ **do**
    $E_h \leftarrow E_h \cup \{(u, w) \in g_\mathrm{c}.V \times g_\mathrm{n}.V : \forall u \in c_j \text{ and } w = g_\mathrm{n}.V_j\}$
  end for
  $g_\mathrm{c} \leftarrow g_\mathrm{n}$
  $h \leftarrow \textsc{IncludeToHierarchy}(h, g_\mathrm{c}, E_h)$
end for

Given a graph g and a clustering $C = \{c_0, \ldots, c_{K-1}\}$, each cluster is summarized

into a new node with a representative label (see line 5). Let us consider that this label is defined as the result of an embedding function applied to the subgraph defined by the clustered nodes and their edges. Moreover, edges between the new nodes are created depending on a connection ratio between clusters. That means that an edge is only created if there are enough connections between the set of nodes defined by both clusters (see line 7). Finally, hierarchical edges are created connecting the new node v_{c_i} with all the nodes belonging to the summarized cluster c_i (see line 12). The proposed hierarchical construction is similar to the one proposed by Mousavi *et al.* [158] but including explicitly the summarization generated by the clustering algorithm by means of the hierarchical edges. Thus, the proposed hierarchical construction obtains a representation which encodes abstract information by means of the clusters while keeping the relation with the original graph.

The following sections are devoted to explain in detail the used contraction function and a proposed improvement dealing with particular cases on our graph representation.

5.2.2 Graph Clustering

Graph clustering has been widely used in several fields such as social and biological networks [96], recommendation systems [91, 136] etc. It is roughly described as the task of grouping graph nodes into clusters depending on the graph structure. Ideally, the grouping should be performed in such a way that intra-cluster nodes are densely connected whereas the connections among inter-cluster nodes are sparse. For example, Girvan and Newman [96] proposed a graph clustering algorithm to detect a community structures for studying social and biological networks. Li *et al.* [91, 128, 136, 137] have proposed several graph clustering techniques for recommendation systems based on different strategies: context awareness [91], inclusion of frequency property [136], distributed clustering confidence [128], etc. Here we do not further review on graph clustering algorithms since it is not within the main scope of this dissertation. However, it is worth remarking that one of the most important aspects of graph clustering is the evaluation of cluster quality, which is crucial not only to measure the effectiveness of clustering algorithms, but also to give insights on the dynamics of relationships in a given graph. For a detailed overview on effective graph clustering metrics, the interested readers are referred to [7].

Even though any graph clustering algorithm can be used, we selected a standard divisive-based algorithm named *Girvan-Newman* [96]. It provides structurally meaningful clusters of a given graph. In addition, the *Girvan-Newman* algorithm is an intuitive and well-known algorithm used for community detection in complex systems. It is a global divisive algorithm which removes the appropriate edge iteratively until all the edges are deleted. At each iteration, new clusters emerge by means of connected components. The idea is that the edges with higher centrality

are the candidates to be connecting two clusters. Therefore, *betweenness centrality* measure of the edges [86] is used to decide which edge is being removed.

Definition 5.2.2 (Betweenness Centrality)**.** Given a graph $g = (V, E, \mu, \nu)$, the betweenness centrality of a node $v \in V$ is defined as the sum of the fraction of all-pairs shortest paths that pass through v. Formally,

$$g(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)},$$

where $\sigma(s,t)$ is the number of shortest paths between s and t; and $\sigma(s,t|v)$ is the number of those paths passing through v. If $s = t$, $\sigma(s,t) = 1$, and if $v = s$ or $v = t$, $\sigma(s,t|v) = 0$.

Similarly, the betweenness centrality of $e \in E$ is defined as the ratio of shortest walks between any pair of nodes that cross e. The idea is that the edges with higher centrality are candidates to connect two distinct clusters. This algorithm follows the hard assignment paradigm, therefore, there are no overlapping clusters. This algorithm consists of 4 steps:

1. Calculate the betweenness centrality for all edges in the network.
2. Remove the edge with highest betweenness and generate a cluster for each connected component.
3. Recalculate the betweenness centrality for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

The output of this algorithm is a dendrogram providing a hierarchical clustering of the graph nodes. In case of ties, *i.e.* several edges have the same betweenness centrality, the edge with more connections in their compounding nodes is deleted.

In this chapter, instead of fixing the number of nodes at each level (see line 3), we propose to contract clusters containing at least two nodes on the final dendrogram, *i.e.* only disconnected nodes will remain the same at different levels. This means that all the clusters will contain at least two nodes. Therefore, the reduction ratio is at least of 2. The nodes belonging to a cluster, are contracted into a new vertex which is labeled with the embedding function φ applied to the corresponding subgraph. Finally, connected communities will create connected nodes. Therefore, following the notation introduced in the Algorithm 5.1, δ is set to 0 and the contraction ratio is dynamically set.

Regarding the embedding function, it is computed on the subgraphs defined by the clusters or communities. Therefore, once the graph communities have been detected, a connected subgraph is defined for each cluster. In the new abstract level, each cluster is represented by a new node with its corresponding graph embedding.

5.2.3 Splitting of Articulation Points

Although *overlapping community detection* techniques have been developed [166], they generate redundant information leading to a bad abstraction. Moreover, they increase the problem complexity. In the present work, we did not face this problem, however, there are cases where slight deformations in the input graphs lead to completely different hierarchies. Symmetries in the original graph is a configuration that easily lead to bad contractions. In particular, in real cases, we find a particular configuration that produces ambiguous contraction, see Figure 5.1. In this particular case, the symmetry is produced by *articulation points* which are defined as follows,

Definition 5.2.3 (Articulation point)**.** A node in an undirected graph is an *articulation point* if and only if in case of removing it, the number of connected components of the graph increases.

Even tough articulation points are not the only source of problems, analyzing real graphs coming from handwritten images, which is our application domain, has shown that they heavily affect performance.

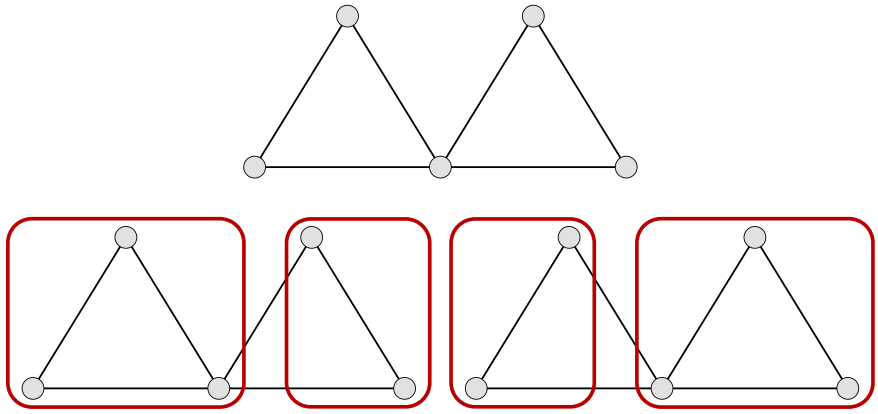
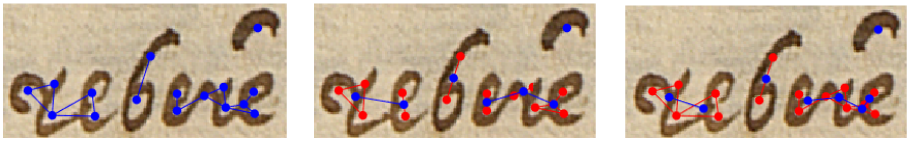


Figure 5.1: Ambiguity configuration that significantly influence in the hierarchy construction, in red two possible clustering of nodes from the contraction function. A slight change in one of the nodes will change completely their abstraction.

These nodes are of key importance, if they are classified in an incorrect cluster, they change drastically the topology in the following levels. Thus, we propose to split the articulation points of the graphs creating *virtual* nodes and disconnecting them. Hence, the hierarchical representation is stabilized without introducing noise to the data. In other words, the articulation points are divided. Therefore, these nodes will belong to two or more clusters. Introducing this modification to the contraction function, a more stable hierarchy is generated. Figure 5.2 shows the splitting process in a real scenario where graphs represent skeleton features



(a) **(b)** **(c)**

Figure 5.2: Example of hierarchy construction for a real graph, in blue the corresponding graph and in red the vertices that are contracted, (a) input graph, (b) hierarchy for the Girvan-Newman based contraction function, (c) hierarchy for the Girvan-Newman based contraction function splitting the articulation points.

in handwritten word images. Note that splitting the articulation point leads to a robust representation of the word "rebere".

5.3 Error Tolerant Hierarchical Matching

Given the hierarchical graph-based representations proposed in the last section, we now formalize a matching scheme able to exploit its hierarchical structure to speed-up the matching process. Let $g_1, g_2 \in \mathcal{G}$ be two graphs, whose hierarchical graph representation of L abstract levels is defined by $h_1 = c_\varphi(g_1)$ and $h_2 = c_\varphi(g_2)$.

In order to take advantage of the hierarchical graph representation, we propose a coarse-to-fine graph matching approach. The developed framework is independent of the matching methodology, therefore, it can be easily changed. In this work, we have used the assignment edit distance (AED) algorithm proposed by Riesen and Bunke in [181]. It is a graph edit distance approximation that has been previously introduced at Chapter 2, therefore, it computes a distance between two given graphs through a set of edit operations.

The same edit costs as the ones presented in Chapter 3 have been used, the node substitution cost is a weighted sum based in the distance between node positions, their attributes and the local structure of incident edges; the edge substitution cost is computed in terms of the edge attribute, angle and length; finally, predefined costs are used for the node and edge insertion and deletion. Thus, there are 8 parameters to tune, 3 for node substitution, 3 for edge substitution and 2 for insertion and deletion.

Let us denote H^i the graph representation at level $i = 0, \ldots, L$, where $i = L$ is the coarsest level, *i.e.* the level with less number of nodes, and $i = 0$ corresponds to the original graph. The proposed approach iteratively refines the matching computation starting at the coarsest level. The comparison is performed using the before mentioned AED as a graph matching technique taking the graph representation at level i ignoring the hierarchical edges. Starting from the smaller graphs,

if the distance between level i representations is small enough, the matching is performed at the next level $i-1$. The threshold to decide whether to advance in the hierarchy or not is application dependent and it is experimentally set using a validation set. Starting the matching at the abstract level avoids a high number of comparisons at more detailed levels where the graphs are significantly bigger. Ideally, the last level is only used for graphs that are very similar to the input one. The information about the matching level is kept to obtain a rough similarity measure.

Figure 5.3 shows the iterative process to decide whether the graphs match or we can discard the following comparisons in any of the deeper levels of the hierarchy.

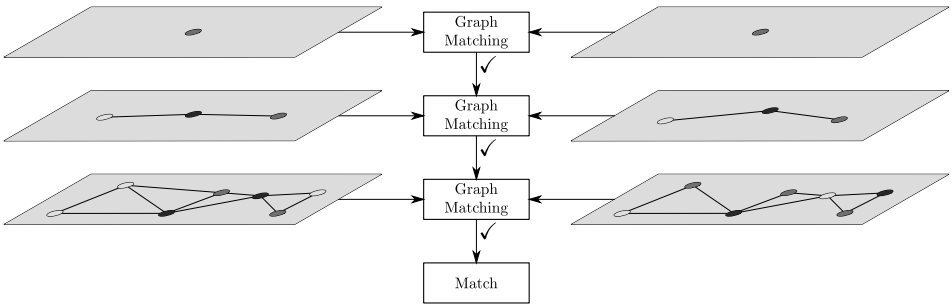


Figure 5.3: Coarse-to-fine graph matching scheme. Each of the hierarchies have three abstract levels and a comparison is performed between these levels independently. This approach may stop the matching comparisons at any of the graph levels.

5.4 Experimental Validation

This section is devoted to validate the performance of the developed approach. To illustrate it, we perform two different tasks. Three real databases have been used to show the potential of the proposed framework. First, the *Columbia Object Image Library* (COIL-100) [160] and the *Object DataBank* (ODBK) [212] have been used to reproduce the experiments proposed by Mousavi *et al.* [158] in an object classification scenario. Second, *The Barcelona Historical Handwritten Marriages Database* (BH2M) [76] database has been used in a graph-based word spotting scenario in handwritten documents where graphs are irregular and suffer from high distortions. Thus, the experiments have been divided into two challenges: *object classification* in the COIL-100 and ODBK datasets and *word spotting* in the BH2M dataset. The thresholds used in this Section have been carefully selected using the validation set.

5.4.1 Object Classification

The COIL-100 database consists of images of 100 different objects taken at 72 equally spaced poses whereas the ODBK database is formed by 209 3D objects with 14 views. For both datasets, graph nodes are extracted using the *Harris corner detector* and the edges are generated using the Delaunay triangulation on these nodes. More details on these datasets are provided in the Appendix A.

Firstly, we have reproduced the experiments proposed in [158] using these two datasets. The objective of this experiment is to classify color images into their corresponding classes using their graph representation. The k-nearest neighbour (k-NN) has been used as a classification framework. For this experiment, the original graphs and the 1st and 2nd abstract levels are evaluated. The selected embedding function φ encodes information of the node position and the Morgan Index of length 1 and 2 of the previous level. Three approaches have been evaluated on how the node information is combined,

- averaging the Morgan Index and node position from all nodes in the graphlet;
- averaging the Morgan Index and selecting the most connected node as the position;
- taking the maximum Morgan Index and the most connected node position.

We only present the results of averaging the Morgan Index and node position from all contracted nodes because other configurations lead to similar performance. In this experiment, the Morgan Index information codifies the local topology or connectivity of the subgraph.

Each level of the hierarchy has been validated alone and combined with the original graph to explore the benefits of the proposed coarse-to-fine matching. All the parameters for the distance computation have been chosen performing a random search in the validation set. Since the graphs are generated using a triangulation method, there are no articulation points, therefore, both contraction functions will lead to the same hierarchical representation.

Table 5.1 shows the performance for both datasets, COIL-100 and ODBK respectively. As far as we know, hierarchical graph representations have not been used until now to prune comparisons and speed-up the matching process. The last 3 rows of the Table correspond to the performance reported by [158] using their hierarchical representation with the same input graphs.

Note that the big loss of performance between the abstract levels is corrected choosing a good trade-off between them. Our approach is able to prune more than the 50% of comparisons at the finest level while achieving good results refining the classification using the original graph. For instance, choosing a conservative threshold, the time reduction is half, losing only 2% of accuracy for the COIL-100 database and 1.5 times faster maintaining the same accuracy for the ODBK

Table 5.1: Performance for Object Classification for COIL-100 (left) and ODBK (right) datasets. Rows are divided in 5 blocks: performance for each level; coarse-to-fine matching using the 1st, 2nd abstract levels and the combination of them; the final row-block correspond the the performance reported by Mousavi *et al.* Columns correspond to the used threshold; accuracy of a k-NN classifier; percentage of avoided comparisons at the base level; time in seconds to perform all the comparisons.

COIL-100 database						
	Thresh.	k-NN(%)			AC[1](%)	t(s)
		1	3	5		
Original	-	100.00	100.00	98.00	-	2010
1st abst.	-	72.67	74.67	72.67	-	167
2nd abst.	-	38.00	39.33	44.67	-	13
1st abst.	0.1982	98.00	97.33	93.33	67.37	977
	0.1680	90.00	89.33	82.67	95.41	289
2nd abst.	0.2153	100.00	99.33	96.67	33.68	1444
	0.1895	97.33	94.67	93.33	58.99	937
1st abst. 2nd abst.	0.1982 0.2153	**98.67**	98.00	92.67	71.63	**893**
Original	-	100.00	97.00	90.00	Mousavi *et al.* [158]	
1st abst.	-	98.17	94.83	88.83		
2nd abst.	-	87.00	81.67	78.17		

ODBK database						
	Thresh.	k-NN(%)			AC[1](%)	t(s)
		1	3	5		
Original	-	79.33	76.00	74.00	-	34959
1st abst.	-	58.67	58.00	54.67	-	1954
2nd abst.	-	42.00	41.33	46.00	-	141
1st abst.	0.2396	79.33	76.00	74.00	48.18	22501
	0.2130	78.67	75.33	72.00	79.10	10496
2nd abst.	0.2973	78.67	74.67	72.67	33.76	26111
	0.2573	76.67	71.33	68.67	68.49	12228
1st abst. 2nd abst.	0.2130 0.2973	**78.00**	74.00	70.67	79.23	**10292**
Original	-	66.67	65.33	63.33	Mousavi *et al.* [158]	
1st abst.	-	66.67	62.67	62.00		
2nd abst.	-	60.00	55.33	53.33		

[1]AC stands for Avoided Comparison

database. However, relaxing this threshold, we are able to achieve a speed-up of 7× with a loss of 10% in accuracy for the COIL-100 database and 3.3× faster losing 1% in accuracy for ODBK. In a large scale scenario, this is an acceptable loss to make an application much faster.

Compared to [158], our methodology does not achieve as good results when taking into account the different abstraction levels alone. One of the main reasons is that our hierarchy is dynamically constructed, not fixing the contraction ratio between levels. Thus, we are generating smaller graphs and generating an abstract representation rather than a change in scale. However, when performing the whole proposed pipeline, our methodology outperforms their abstract levels with a significant speed-up, 98% in the COIL-100 database and 78% in the ODBK database, with a time reduction higher than 50%.

5.4.2 Word Spotting

Continuing with our running experiment on the BH2M database [76] we evaluate our hierarchical representation and matching for the retrieval problem of word spotting on historical documents

Figure 5.4 shows the interpretation of the hierarchical graph representation in the context of this database. Observe how the graphemes are combined at each level to create more complex shapes like letters, bi-grams and finally words.

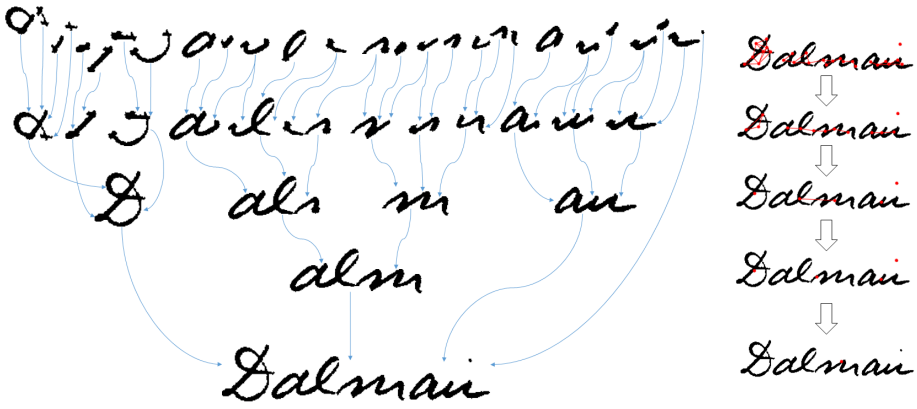


Figure 5.4: Construction of the hierarchical graph representation for the word "Dalmau", convexities extracted from the word vectorization and how are combined following the hierarchical edges (left) and grapheme graph at 5 resolution levels.

For this experiment, the embedding function φ consists of a vector that counts the number of walk of length up to k from any node to a node with label i. The best configuration has been $k = 0$, *i.e.* counting the number of nodes with label i,

which is similar to a bag of words for the nodes on the contracted subgraph. In this experiment, the same parameters proposed in Chapter 3 for the original graphs have been used in order to compute the edit cost operations. Table 5.2 shows the performance at the original level alone or using the two proposed contraction functions. Note that splitting the articulation points leads to a more robust representation of the graph. This is caused by the sequential nature of the handwritten strokes that leads our representation to generate lots of ambiguous situations.

Table 5.2: Comparison of the proposed hierarchical framework changing the contraction function in the BH2M dataset. The original performance is compared with and without splitting the articulation points.

	mAP (%)
Original [177]	69.45
1st abstraction	35.67
+ split	46.37

Figure 5.5 shows qualitative results of the retrieved words at two levels. Note that in the 1st abstract levels, we retrieve some incorrect words. However, in terms of shape, these words are quite similar. In fact, most of them, such as "ferrer", "ferran" and "forner", will fall in the same class in a coarse human classification in terms of shape.

Table 5.3: Comparison of the proposed hierarchical framework against an indexation framework introduced in Chapter 4. The mean average precision corresponds to the evaluation of the word spotting problem; recall (R) and specificity (SPC) are computed on the selected graphs using the hierarchy or the indexation respectively; finally, the time per query is provided.

	mAP (%)	**R (%)**	**SPC (%)**	**Time/query[2] (s)**
Original [177]	69.45	100.00	0.00	19.58
+abst. (t=0.30)	**68.27**	90.91	69.98	**12.46**
+abst. (t=0.25)	**61.71**	67.93	97.91	**3.94**
+ [179] (t=0.20)	66.13	92.54	46.13	16.34
+ [179] (t=0.30)	61.15	83.55	63.04	12.74

A key element in our approach is to define a good threshold that will determine a good trade-off between speed and performance. Figure 5.6 shows the evolution of avoided comparisons at the original level and mAP changing this threshold.

Table 5.3 shows a comparison between the original graphs, the proposed framework with two thresholds, and the graph indexation proposed in the previous chapter. Recall (R) and specificity (SPC) are computed on the selected graphs using the first

[2] 1000 queries selected randomly against 13098 graphs

Figure 5.5: Qualitative results for the query "ferrer" extracted from the BH2M dataset. The second row shows the retrieval using the original graph (ordered by rows). The last row shows the power of using **only** the first abstraction level. The green words are the correctly retrieved whereas, the red ones are incorrectly retrieved.

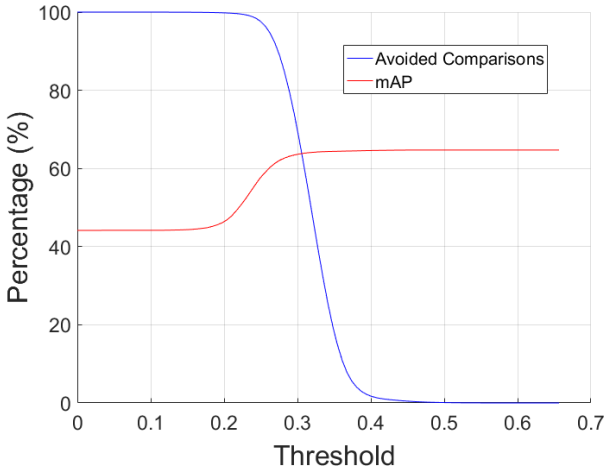


Figure 5.6: Avoided comparisons and mAP evolution changing the threshold to decide whether or not use the second level of the hierarchy. The plot is generated with the validation set to decide the thresholds to use.

abstract level as classifier. The thresholds have been set according to Figure 5.6. Notice that the proposed hierarchical framework achieves high specificity whereas keeping a better trade-off with the recall than the indexation approach. Moreover, only losing 8% of mAP which is acceptable for a large scale retrieval we are able to speed up the process almost 5 times.

5.5 Conclusions

This chapter has presented a construction of a hierarchical graph representation by means of contraction and embedding functions. Contraction uses graph clustering techniques to gather nodes and simplify the graph. Moreover, a modification of the contraction function has been proposed to stabilize the hierarchy in certain graphs. The proposed method is able to significantly reduce the graph size allowing a fast graph comparison through a coarse-to-fine matching approach. This methodology prunes the amount of comparisons in the fine level. The approach has been exhaustively validated using several databases for large-scale graph retrieval. Compared to other related works, the proposed approach dynamically gathers the nodes without predefining the number of clusters, therefore, the ratio of reduction for each sample can change. Furthermore, the graph size is extremely reduced from one level to another.

[2]Computed on 1000 randomly selected queries against 13098 graphs

We conclude that hierarchical graph representations are a powerful tool in the matching process. This representation gives information about the relation of a group of nodes (those that are contracted) instead of the typical pair-wise relations. Moreover, each level of the hierarchy can be enriched following other indexing methodologies such as the one proposed in the previous chapter.

Even though hierarchical representations have a powerful representational power, in the proposed approach, the hierarchical connections are not exploited. Moreover, the proposed coarse-to-fine matching, still relies on the GED which is rather slow (cubic in terms of number of nodes for the AED [181] approximation). Furthermore, the coarse-to-fine matching expects the different hierarchical levels to be aligned. This means that the matching is only computed level by level instead of finding the best correspondence.

6 | Hierarchical Stochastic Graphlet Embedding

The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!" (I found it!) but "That's funny...".

– Isaac Asimov

In this chapter, we consider the hierarchical structure of a graph as a way to mitigate the loss of structural information of graph embeddings. Following the previous chapter, the hierarchical structure is constructed by topologically clustering the graph nodes, and considering each cluster as a node in the upper hierarchical level. Once this hierarchical structure is constructed, we consider several configurations to define the mapping into a vector space given a classical graph embedding, in particular, we propose to make use of the stochastic graphlet embedding (SGE). Broadly speaking, SGE produces a distribution of uniformly sampled low to high order graphlets as a way to embed graphs into the vector space. In what follows, the coarse-to-fine structure of a graph hierarchy and the statistics fetched by the SGE complements each other and includes important structural information with varied contexts. Altogether, these two methodologies substantially cope with the usual information loss involved in graph embedding techniques, obtaining a more robust graph representation. This fact has been corroborated through a detailed experimental evaluation on various benchmark graph datasets, where we outperform the state-of-the-art methods.

6.1 Introduction

As we have seen in previous chapters, graphs have been successfully applied to a huge variety of tasks. However, due to their symbolic and relational nature, graphs have always suffered from some limitations if we compare them with the traditional statistical (vector-based) representations. Some trivial mathematical operations do not have an equivalence in the graph domain. In the literature, a possible way this problem has been addressed is by means of embedding functions. As reviewed in Chapter 2, given a graph space $\mathcal{G}$, an *explicit embedding* function is defined as $\varphi\colon \mathcal{G} \to \mathbb{R}^n$ which maps a given graph to a vector repre-

sentation [35, 93, 147, 193, 201]. However, defining such embedding functions is extremely challenging, when the constraints on time efficiency and preserving the underlying structural information is concerned. The problem becomes even more difficult with the growing size of graphs, as the structural complexity increases the possibility of noise and distortion in structure, and raises risk of loosing information. As we have already introduced, hierarchical representations are often used as a way to deal with noise and distortion [158, 215], which provides a stable delineation for an underlying object. Hierarchical representations allow to incrementally contract the graph, in a space-scale representation, so the salient features (relevant subgraphs) remain in the hierarchy. Thus, top levels become a compact and stable summarization.

Motivated by the successes of the hierarchical models and the efficiency of graph embedding theory, we propose a general hierarchical graph embedding formulation that first creates a hierarchical structure from a given graph, and then utilizes the multi scale structure to explicitly embed a graph in a real vector space by means of local graphlets. Similarly to the previous approach, firstly, we make use of the graph clustering algorithm proposed in [96] to obtain a hierarchical graph representation of a given input graph. Here, each cluster of nodes in a level i is depicted as a single node in the upper hierarchical level $i + 1$, whereas the edges in a level are connected depending on the original topology of the base graph, and the hierarchical edges are created by joining a node representing a cluster to all the nodes in the lower level. Mousavi *et al.* [158] proposed a similar approach that is inspiring our model. Our encoding is richer because our hierarchy not only contains different graph abstractions but also encodes useful hierarchical contractions through the hierarchical edges.

Once the hierarchical structure of a graph is created, we propose a novel use of the *Stochastic Graphlet Embedding* (SGE) [66] to exploit this hierarchical information. On the one hand, we exploit the local configuration in form of graphlets thanks to the SGE design, because graphlets provide information at different neighborhood sizes. On the other hand, the hierarchical connections allow to encode more abstract information and hence to deal with noise present in the data. As a result, the *Hierarchical Stochastic Graphlet Embedding* (HSGE) encodes a global and compact representation of the graph that is embedded in a vector space. The consideration of the entire graph hierarchy for the embedding instead of only the base graph empowers the representation ability and handles the loss of information that usually occurs in graph embedding methods. Moreover, the statistics obtained from the uniformly sampled graphlets of increasing size model the complex interactions among different object parts represented as graph nodes. Here, the hierarchical graph structure and the statistics of increasing sized graphlets fetch important structural information of varied contexts.

As a result, our approach produces robust representations that benefits from the advantages of the two above mentioned strategies: we first take advantage of the embedding ability for mapping symbolic relational representations to n-

dimensional spaces, so machine learning approaches can be used; and second, the ability of hierarchical structures to reduce noise and distortion inherently involved in graph representations of real data, keeping the more stable and relevant substructures in a compact way.

In conclusion, the main contribution of the present chapter is the exploitation of the hierarchical structure of a given graph, rather than only studying the base graph for graph embedding purposes. Assessing the hierarchical information of a graph pyramid allows to extend the representation power of the embedded graph and tolerate the instability caused due to noise and distortion. Our proposal is robust because, on the one hand, it organizes the structural information in the hierarchical abstraction, and on the other hand, it considers the relation between object parts and their complex interactions with the help of uniformly sampled graphlets of unbounded size. Additionally, the proposed method is generic and can adapt any other graph embedding algorithm in the framework. In this sense, we extensively validated our proposed algorithm on many different benchmark graph datasets coming from different application domains.

The rest of this chapter is organized as follows. The generic hierarchical graph representation is presented in Section 6.2. Section 6.3 introduces the Stochastic Graphlet Embedding as the base embedding we will use. Section 6.4 studies the computational complexity of the whole embedding pipeline. Afterwards, Section 6.5 reports the experimental validation and compares the proposed method with available state-of-the-art algorithms. Finally, in Section 6.6 we draw the conclusions and describe future directions of this work.

6.2 Hierarchical Graph Embedding

In the literature, only few embedding approaches exploit the idea of multi-scale or abstraction information. This section is devoted to provide a framework able to include this information given a graph embedding. Some works that have been proposed to exploit the mentioned multi-scale information in the literature [65, 158, 178], discard the hierarchical information provided by the hierarchical edges and focus on abstractions of the original graph. Other works, do not explicitly create a hierarchical structure [126].

As introduced in the previous chapter, the proposed hierarchical representation is constructed in terms of a graph clustering approach and a graph embedding function in charge of summarizing the detected clusters.

6.2.1 Hierarchical Construction

Following the hierarchical construction introduced in Algorithm 5.1. The main difference from the previous chapter is that in this case, the *Girvan-Newman* al-

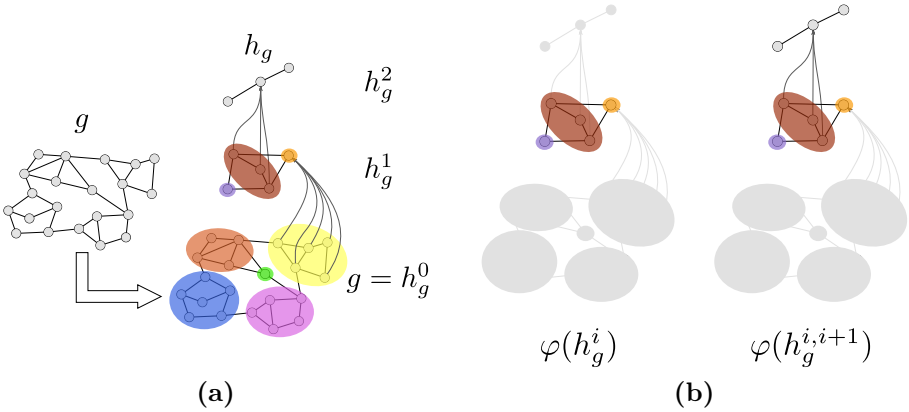


Figure 6.1: Overview of the hierarchical graph notation. (a) Hierarchical graph construction is proposed in Algorithm 5.1. The input graph g is processed to generate a hierarchical graph h_g where each level h_g^i encodes a new abstraction of the original graph. Moreover, hierarchical edges provide the insights of the performed contraction. In this figure, not all the hierarchical edges have been drawn to make it easy to understand, and the node clustering is drawn in color. (b) Following the hierarchical graph construction in (a), the graphs taken into consideration in order to construct the hierarchical embedding are shown. $\varphi(h_g^i)$ takes into account one abstraction level whereas $\varphi(h_g^{i,i+1})$ takes into consideration two of these levels and the hierarchical edges involved.

gorithm is early stopped given the reduction ratio $r \in \mathbb{R}$ described in line 3. Therefore, the number of clusters is forced to be $\lfloor r \cdot |V| \rfloor$.

Following the Definition 2.5.1, let us introduce some notations regarding the hierarchical representation that will be used in the following sections. Given a graph g and a number of levels L, h_g denotes their corresponding hierarchical graph computed from g with L levels. h_g^l, where $l = \{0, \ldots, L\}$ is a graph without hierarchical edges corresponding to the l level of summarization, therefore, $h_g^0 = g$. Moreover, $h_g^{l_1,l_2}$ where $l_i = \{0, \ldots, L\}$ and $l_1 \leq l_2$, corresponds to the hierarchical graph compressed between levels l_1 and l_2. Hence, $h_g = h_g^{0,L}$ and $h_g^l = h_g^{l,l}$. Finally, $h_g^{l_1} \cup h_g^{l_2}$ corresponds to the union of two graphs without hierarchical edges.

Figure 6.1(a) shows the construction of the hierarchy given a graph g. Each level shows an abstraction of the input graph where the nodes have been reduced.

6.2.2 Hierarchical Embedding

This section introduces a novel way to encode hierarchical information of a graph into an embedding. Moreover, the proposed technique is generic in the sense that

it can be used by any graph embedding function.

Given a graph g which should be mapped into a vectorial space and an embedding function $\varphi\colon \mathcal{G} \to \mathbb{R}^n$, we first proceed to obtain a hierarchical representation h_g following the proposed methodology described in Section 6.2.1. Therefore, h_g has enriched the original graph with abstract information considering L levels. Finally, we propose to make use of the hierarchical information to construct a hierarchical embedding. The general form of the proposed embedding takes into account graphs at multiple scales and hierarchical relations. Thus, the embedding function does not only compactly encode the contextual information of nodes at different abstraction levels, but also it encodes the hierarchy contraction. The embedding function is defined as follows:

$$\Phi(h_g) = \left[\varphi(h_g^0), \ldots, \varphi(h_g^K), \phi_1^1(h_g), \ldots, \phi_1^{k_1}(h_g), \phi_2^1(h_g), \ldots, \phi_2^{k_2}(h_g)\right] \tag{6.1}$$

where,

$$\phi_1^k(h_g) = \left[\varphi(h_g^{0,k}), \ldots, \varphi(h_g^{K-k,K})\right] \tag{6.2}$$

$$\phi_2^k(h_g) = \left[\varphi(h_g^0 \cup \cdots \cup h_g^k), \ldots, \varphi(h_g^{K-k} \cup \cdots \cup h_g^K)\right] \tag{6.3}$$

where $K \leq L$ are the considered hierarchical levels and $k_1, k_2 \leq K$ indicate the number of levels taken into account at the same time. Note that when the setting is $K = L$, $k_1 = K$ and $k_2 = K$, the whole hierarchy and possible combinations is taken into account. From this general representation of the proposed embedding, we have evaluated some particular cases.

Baseline embedding: This embedding is the one used as a baseline. In this scenario $K = 0$, $k_1 = 0$ and $k_2 = 0$, therefore $\Phi(h_g) = \varphi(h_g^0)$. No abstract information is taken into consideration, hence, the embedding Φ is defined as:

$$\Phi(h_g) = \varphi(g). \tag{6.4}$$

Pyramidal embedding: This embedding has been previously proposed in the literature [65, 158]. It combines information of the abstract levels of the graph *i.e.* h_g^i not taking into account hierarchical information. Therefore, the hierarchical edges are discarded and no relation between levels is considered, $K \geq 1$, $k_1 = 0$ and $k_2 = 0$. We define the embedding Φ as follows:

$$\Phi(h_g) = \left[\varphi(h_g^0), \ldots, \varphi(h_g^K)\right]. \tag{6.5}$$

Note that each element corresponds to independent levels of the hierarchy without hierarchical edges.

Generalized pyramidal embedding: Following the previous idea, the information of the abstract levels of the graph *i.e.* h_g^i is combined. Now, hierarchical information is taken into account by embedding unions of levels *i.e.* $h_g^{i_1} \cup h_g^{i_2}$ but

discarding hierarchical edges (no clustering information is taken into account). In this scenario $K \geq 1$, $k_1 = 0$ and $k_2 \geq 1$, therefore, we define the embedding Φ as:

$$\begin{aligned}\Phi(h_g) = \big[&\varphi(h_g^0), \ldots, \varphi(h_g^K), \\ &\varphi(h_g^0 \cup h_g^1), \ldots, \varphi(h_g^{K-1} \cup h_g^K), \ldots, \\ &\varphi(h_g^0 \cup \cdots \cup h_g^{k_2}), \ldots, \varphi(h_g^{K-k_2} \cup \cdots \cup h_g^K)\big]\end{aligned} \tag{6.6}$$

Hierarchical embedding: This embedding is computed mixing different levels considering them as a single graph with two types of edges, namely hierarchical and intra-level, $K \geq 1$, $k_1 \geq 1$ and $k_2 = 0$. The idea is to create an embedding able to codify both, graph and clustering information. Depending on the embedding, hierarchical edges can make use of special label to treat them differently. The hierarchical embedding is defined as:

$$\begin{aligned}\Phi(h_g) = \big[&\varphi(h_g^0), \ldots, \varphi(h_g^K), \\ &\varphi(h_g^{0,1}), \ldots, \varphi(h_g^{K-1,K}), \ldots, \\ &\varphi(h_g^{0,k_1}), \ldots, \varphi(h_g^{K-k_1,K})\big]\end{aligned} \tag{6.7}$$

Note that each element corresponds to the subhierarchy compressed between the specified levels.

Exhaustive embedding: Finally, in order to take into consideration the whole hierarchy, we make use of the whole embedding Φ as defined in Equation 6.1 where $K \geq 1$, $k_1, k_2 \geq 1$.

Figure 6.1(b) shows the graphs taken into consideration when the hierarchical embeddings are computed.

6.3 Stochastic Graphlet Embedding

As defined in Definition 2.4.1, a graph embedding is a function $\varphi\colon \mathcal{G} \to \mathbb{R}^n$ that explicitly embeds a graph $g \in \mathcal{G}$ to a high dimensional vector space $\mathbb{R}^n$. In this section, we introduce the *Stochastic Graphlet Embedding* (SGE) proposed by Dutta *et al.* [66]. The entire procedure of SGE is described in two stages (see Figure 6.2), where in the first step, the method samples graphlets from g in a stochastic manner and in the second step, it counts the frequency of each isomorphic graphlet from the extracted ones in an approximated but near accurate manner. The entire procedure fetches a precise distribution of connected graphlets with increasing number of edges in g with a controlled complexity, which fetches the relation among information represented as nodes and their complex interaction.

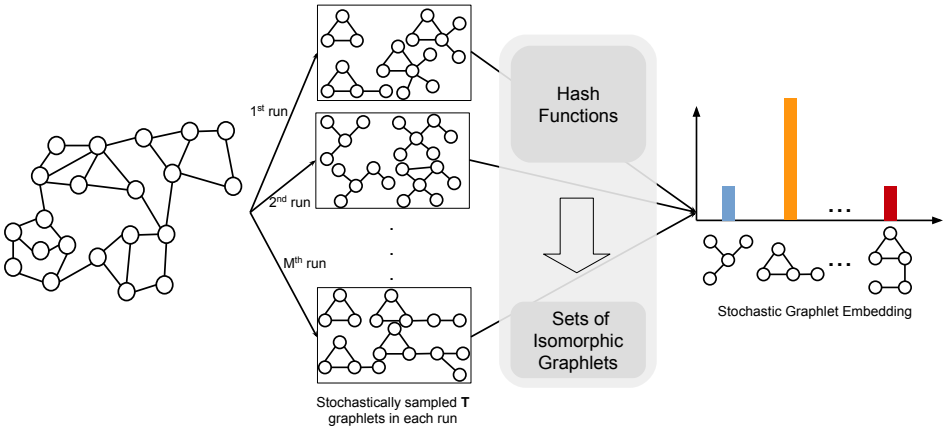


Figure 6.2: Overview of stochastic graphlet embedding (SGE). Given a graph g, the stochastic parsing algorithm is able to uniformly sample graphlets of increasing size. Controlled by two parameters M (number of graphlets to be sampled) and T (maximum size of graphlets in terms of number of edges), the method extracts in total $M \times T$ graphlets. These graphlets are encoded and partitioned into isomorphic graphlets using the set of hash functions with a low probability of collision. A distribution of different graphlets is obtained by counting the number of graphlets in each of these partitions. This procedure results in a vector space representation of the graph g referred to as stochastic graphlet embedding.

6.3.1 Stochastic Graphlets Sampling

Considering a graph $g = (V, E, \mu, \nu)$, the goal of the graphlet extraction procedure is to obtain statistics of stochastic graphlets with increasing number of edges in g. The way of extracting graphlets is stochastic and it uniformly samples graphlets with boundlessly increasing number of edges without constraining their topology or structural properties such as maximum degree, maximum number of nodes, etc. Our graphlet sampling procedure, outlined in Algorithm 6.1, is iterative and the number of runs is controlled by a parameter M that indicates the number of distinct graphlets to be sampled (see line 2 of Algorithm 6.1). Also, each of these M processes are regulated by another parameter T that denotes the maximum number of iterations a single process should have (see line 5). Since each of these iterations adds an edge to the presently constructing graphlet, T indirectly specifies the maximum number of distinct edges each graphlet should contain. Considering U_t and A_t respectively as the aggregated sets of visited nodes and edges till iteration t, they are initialized at the beginning of each step as $A_0 = \emptyset$ and $U_0 = \{u\}$ with a randomly selected node u which is uniformly sampled from V (see line 4). Thereafter, at t-th iteration (with $t \geq 1$), the sampling procedure randomly selects an edge $(u, v) \in E \backslash A_{t-1}$ that is connected from any node $u \in U_{t-1}$ (see line 7). Accordingly, the process updates $U_t \leftarrow U_{t-1} \cup \{v\}$ and $A_t \leftarrow A_{t-1} \cup \{(u, v)\}$ (see

line 8). All these processes within a step are repeated T times to sample a graphlet with maximum T edges. M is set to relatively large values in order to make the graphlet generation statistically meaningful. Theoretically, the values of M are guided by the theorem of *sample complexity* [229], which is widely studied and used in the bioinformatics domain [170, 203]. However, the discussion and proof of that is out of scope of the current dissertation. Intuitively, the graphlet sampling procedure explained in this section follows a random walk process with restart that efficiently parses g and extracts the desired number of connected graphlets with an increasing number of edges. This algorithm allows to sample connected graphlets from a given graph but avoids expensive way of extracting them in an exact manner. Here the hypothesis is that if a sufficient number of graphlets are sampled, then the empirical distribution will be close to the actual distribution of graphlets in the graph. Furthermore, it is important to note that from the above process, one can extract, in total, $M \times T$ graphlets each with number of edges varying from 1 to T.

Algorithm 6.1 Stochastic-Graphlet-Parsing which obtains a set of graphlets $\mathcal{S}$ by traversing g.

Input: $g = (V, E)$, M, T
Output: $\mathcal{S}$
1: $\mathcal{S} \leftarrow \emptyset$
2: **for** $i = 1$ to M **do**
3: $u \leftarrow \text{SELECTRANDOMNODE}(V)$
4: $U_0 \leftarrow u$, $A_0 \leftarrow \emptyset$
5: **for** $t = 1$ to T **do**
6: $u \leftarrow \text{SELECTRANDOMNODE}(U_{t-1})$
7: $v \leftarrow \text{SELECTRANDOMNODE}(V) : (u, v) \in E \setminus A_{t-1}$
8: $U_t \leftarrow U_{t-1} \cup \{v\}$, $A_t \leftarrow A_{t-1} \cup \{(u, v)\}$
9: $\mathcal{S} \leftarrow \mathcal{S} \cup \{(U_t, A_t)\}$
10: **end for**
11: **end for**

6.3.2 Hashed graphlets distribution

For obtaining a distribution of the extracted graphlets from g, it is needed to identify sets of isomorphic graphlets from the sampled ones and then count cardinality of each isomorphic set. A trivial way of doing that certainly involves checking the graph isomorphism for all possible pairs of graphlets for detecting possible partitions that might exist among them. Nevertheless, graph isomorphism is a GI-complete problem [154] for general graphs, so the previously mentioned scheme is extremely costly as the method samples huge number of graphlets with many edges. An alternative, efficient and approximate way of partitioning isomorphic graphlets is *graph hashing*. A graph hash function is defined as a mapping

$H : \mathcal{G} \rightarrow \mathbb{R}^m$ that maps a graph into a hash code (a sequence of real numbers) based on the local as well as holistic topological characteristic of graphs. An ideal graph hash function should map two isomorphic graphs to the same hash code as well as two non-isomorphic graphs to two different hash codes. While it is easy to design hash functions satisfying the condition that two isomorphic graphs should have the same hash code, it is extremely difficult to find hash function that ensures different hash codes for every pair of non-isomorphic graphs. An alternative is to design graph hash functions with low collision probability, *i.e.* mapping any two non-isomorphic graphs to the same hash code with a very low probability. For obtaining a distribution of graphlets, the main aim of graph hashing is to assign extracted graphlets from g to corresponding subsets of isomorphic graphlets (*a.k.a.* partition index or histogram bins) in order to count and quantify their distributions. The proposed mechanism for obtaining the distribution of uniformly sampled graphlets, outlined in Algorithm 6.2, maintains a global hash table $\mathbf{H}$, whose single entry corresponds to a hash code of a graphlet g_i produced by the graph hash function. $\mathbf{H}$ grows incrementally as the algorithm confronts new graph hash codes and maintains all the unique hash codes encountered by the system. It is to be noted that the position of each unique hash code is kept fixed, because each position corresponds to a partition index or histogram bin. Now to allocate a given graphlet g_i to its corresponding histogram bin, its hash code $H(g_i)$ is mapped to the index of the hash table $\mathbf{H}$, whose corresponding graph hash code gives a hit with $H(g_i)$ (see line 8). If $H(g_i)$ does not exist in $\mathbf{H}$ at some instance, it is considered as a new hash code (and hence g_i as a new graphlet) encountered by the system and appended $H(g_i)$ at the end of $\mathbf{H}$ (see line 6).

Algorithm 6.2 Hashed-Graphlets-Statistics which creates a histogram $\mathbf{h}$ of graphlet distribution for a graph g.

Input: g, $\mathbf{H}$
Output: $\mathbf{h}$
1: $\mathcal{S} \leftarrow$ Stochastic-Graphlet-Parsing(g)
2: $\mathbf{h}_i \leftarrow 0,\ i = 1, \ldots, |\mathcal{S}|$
3: **for all** $g_i \in \mathcal{S}$ **do**
4: $\quad H(g_i) \leftarrow$ HashFunction(g_i)
5: $\quad$ **if** $H(g) \notin \mathbf{H}$ **then**
6: $\quad\quad \mathbf{H} \leftarrow \mathbf{H} \cup \{H(g_i)\}$
7: $\quad$ **end if**
8: $\quad i \leftarrow$ GetIndex-In-HashTable($H(g_i)$)
9: $\quad \mathbf{h}_i \leftarrow \mathbf{h}_i + 1$
10: **end for**

Designing hash functions that yield identical indices for two isomorphic graphlets is quite simple, whereas prototyping those providing two distinct hash codes for two non-isomorphic graphs is very challenging. The chance of mapping two non-isomorphic subgraphs to the same hash code is termed as *collision probability*. Indicating H_0 as the set of all pairs of non-isomorphic graphs, the probability of

collision is expressed as the following energy function:

$$E(f) = P\left((g, g') \in H_0 \quad | \quad f(g) = f(g')\right) \tag{6.8}$$

So, in terms of collision probability, the hash functions that produce comparatively lower $E(f)$ values in Equation 6.8 are considered to be more reliable for checking the graph isomorphism. It has been studied that sorted *degree of nodes* has 0 collision probability for all graphs with number of edges less or equal to 4 [66]. Moreover, it is also a well known fact that two graphs with the same *betweenness centrality* (sorted) would indeed be isomorphic with high probability [48, 162]. For example, sorted *betweenness centrality* has collision probabilities equal to $3.2e^{-4}$, $1.9e^{-4}$, $1.1e^{-4}$ respectively for graphlets with 7, 8 and 9 edges. Interested readers are requested to see [66] for further discussions and analysis on various graph hash functions and corresponding elaboration on probability of collision. Considering the above facts, in this work, we consider sorted *degree of nodes* for graphlets with $t \leq 4$ and the *betweenness centrality* for graphlets with $t \geq 5$.

$$\text{Hash function} = \begin{cases} \text{sorted degree of nodes,} & \text{if } t \leq 4 \\ \text{sorted betweenness centrality,} & \text{otherwise} \end{cases} \tag{6.9}$$

It should be observed that the distribution of sampled graphlets obtained by the way mentioned until now, only considers the topological structure of a graph, and ignores the node and edge attributes. However, it is worth mentioning that the stochastic graphlet embedding permits to consider a small set of nodes and edge attributes by creating respective signatures and then appending it to the hash code encoding the topology of the graphlet. In this work, if needed, we first discretize the existing continuous attributes using a combination of clustering algorithm such as *k-means* and pooling technique. Later, the sorted discrete node and edge labels are used as the attribute signatures and combined with the hash code.

6.3.3 Hierarchical Stochastic Graphlet Embedding

We propose to combine the properties of the proposed *Stochastic Graphlet Embedding* with the *Hierarchical Embedding* introduced in the previous section.

On the one hand, SGE provides statistical information about local structures varying the number of edges involved. Therefore, it provides fine-grained insights of the graph which cannot deal with too noisy data. The use of abstractions provided by the graph hierarchy increases the receptive field of each graphlet moving to coarser information that is able to provide insights of the global graph information. Moreover, the use of hierarchical edges during the computation allows to combine information at some levels, *i.e.* combining different levels of detail (see Equation 6.1). For now on, we will denote this embedding as *Hierarchical Stochastic Graphlet Embedding* (HSGE).

6.4 Computational Complexity

This Section is devoted to study the computational complexity of the proposed approach given a graph $v = (V, E, \mu, \nu)$ where $|V| = n$ and $|E| = m$.

6.4.1 Hierarchical Embedding Complexity

Graph clustering algorithms are usually high computational complexity techniques. As it has been stated in Section 6.2.2, the *Girvan-Newman* algorithm has been chosen as a graph clustering technique. The Girvan-Newman algorithm is based on the betweenness centrality of the edges which has a time complexity of $\mathcal{O}(n \cdot m)$ for unweighted graphs and $\mathcal{O}(n \cdot m + n \cdot (n+m)\log(n))$ for weighted graphs. Hence, the *Girvan-Newman* algorithm, which has to remove all the edges, can be computed in $\mathcal{O}(n \cdot m^2)$ for unweighted graphs and $\mathcal{O}(n \cdot m^2 + n \cdot m \cdot (n+m)\log(n))$ for weighted graphs.

Assuming an embedding function φ which has a complexity of $\mathcal{O}(N)$ and assuming that the hierarchical graph construction has a complexity of C_1, then, if we assume L levels, the proposed configurations would become a complexity $\mathcal{O}(C_1 + L \cdot N)$ in the case of the pyramid and $\mathcal{O}(C_1 + L^2 \cdot N)$ for the hierarchy and the exhaustive embeddings.

6.4.2 Stochastic Graphlet Embedding Complexity

The computational complexity of Algorithm 6.1 is $\mathcal{O}(M \cdot T)$ where M is the number of graphlets to be sampled and T is the maximum size of graphlets in terms of the number of edges. Assuming a hash function with a complexity of $\mathcal{O}(C_2)$, Algorithm 6.2 has a time complexity of $\mathcal{O}(M \cdot T \cdot C_2)$ for computing the stochastic graphlet embedding. Here it is worth mentioning that "degree of nodes" and "betweeness centrality", respectively have the time complexity of $\mathcal{O}(n)$ and $\mathcal{O}(n \cdot m)$. From the above explanation, it is clear that the complexity of these two algorithms does not depend on the size of the input graph g, but only on the parameters M, T and the hash functions used.

6.5 Experimental Validation

This section presents the experimental results obtained by our proposed Hierarchical Stochastic Graphlet Embedding method. The main aim of this experimental study is to validate the proposed graph embedding technique for the graph classification task, which demands robust embedding technique for mapping a graph into a vector space. For experimentation, we have considered many different widely used graph datasets with varied characteristics. All these graphs

come from real data generated in the fields of Biology, Chemistry, Graphics and Handwriting recognition. The MATLAB code of our experiment is available at `https://github.com/priba/hierarchicalSGE`.

6.5.1 Experiments on Molecular Graph Datasets

The first set of experiments is conducted on various benchmarks of molecular graphs. Several bioinformatics databases have been used, *viz.* *MUTAG*, *PTC*, *PROTEINS*, *NCI1*, *NCI109*, *D&D* and *MAO*. These datasets have been widely used as benchmark in the pattern recognition literature representing chemical compounds for a binary classification problem. All the datasets are carefully described in Appendix A.

Experimental setup

We have performed two different experiments: the first one does not use the attribute information encoded in the nodes and edges of the graphs, whereas the second experiment does use the available node and edge features. For evaluating the performance of the proposed embedding technique, we have used a C-*Support Vector Machine* (C-SVM) solver [41] as a classifier. Since the datasets considered in this set of experiments do not contain predefined train and test sets, we have used a 10-fold cross validation scheme to obtain accuracies and have reported the mean accuracies respectively in Table 6.1 and Table 6.2 for unlabeled and labeled datasets. We follow a classical graph classification pipeline where, in the first stage, graph embedding is computed by our proposed scheme, whereas in the second step, embedded graphs are classified using a previously trained classifier.

Results and discussion

In Table 6.1, we present the experimental results obtained by our proposed hierarchical embedding techniques together with other existing works on the unlabeled datasets. The previously mentioned three configurations of our hierarchical embedding are respectively denoted as: pyramidal, hierarchical and exhaustive. For unlabeled datasets, we have considered 10 different state-of-the-art methods: (1) random walk kernel (RW) [90], (2) shortest path kernel (SP) [23], (3) graphlet kernel (GK) [203], (4) Weisfeiler-Lehman kernel (WL) [202], (5) deep graph kernel (DGK) [238], (6) multiscale Laplacian graph kernel (MLK) [126], (7) diffusion CNNs (DCNN) [11], (8) strong graph spectrums (SGS) [125], (9) family of graph spectral distances (F_GSD) [219], and (10) stochastic graphlet embedding (SGE) [66].

From the quantitative results shown in Table 6.1, it should be observed that for most datasets, the highest accuracy is achieved by one of the hierarchical configurations proposed by us, which sets a new state-of-the-art results on all the datasets considered. Particularly, the best accuracies are obtained either by the pyramidal or the exhaustive configurations, which indicates the importance of considering hierarchical information for the graph embedding problem. As expected, the pro-

posed hierarchical embeddings have achieved better performance than the SGE which is regarded as the baseline of our proposal. It should be observed that with this experimental setting, particularly the hierarchical configuration has performed quite poorly compared to the other two configurations. This fact might suggest that only hierarchical edges together with the connecting levels do not contain sufficient information for a robust graph representation. Information captured in the multi-scale graphs thought to play a vital role for graph embedding, which is proved by the excellent performance obtained with the pyramidal and exhaustive configurations.

Table 6.1: Classification accuracies on *unlabeled* molecular graph datasets. In the table, RW corresponds to the random walk kernel [90], SP stands for the shortest path kernel [23], GK denotes the graphlet kernel [203], WL indicates the Weisfeiler-Lehman kernel [202], DGK corresponds to the deep graph kernel [238], MLK stands for the multiscale Laplacian graph kernel [126], DCNN indicates the diffusion CNNs [11], SGS denotes the strong graph spectrums [125], F_GSD stands for the family of graph spectral distances [219], SGE corresponds to the stochastic graphlet embedding [66], and HSGE indicates the hierarchical graph embeddings proposed by us. The best results obtained on a dataset is indicated by bold face.

Methods	MUTAG	PTC	PROTEINS	NCI1	NCI109	D&D	MAO
RW [90]	83.50	55.52	68.46	44.84	59.80	–	83.52
SP [23]	87.23	58.72	72.14	68.15	68.30	–	90.35
GK [203]	84.04	60.17	71.78	62.07	62.04	75.05	80.88
WL [202]	87.28	55.61	70.06	77.23	78.43	73.76	89.79
DGK [238]	86.17	59.88	71.69	64.40	67.14	72.75	87.76
MLK [126]	87.23	62.20	71.35	77.57	75.91	77.02	91.17
DCNN [11]	66.51	55.79	65.22	63.10	60.67	OMR	76.10
SGS [125]	88.61	–	–	62.72	62.62	–	–
F_GSD [219]	92.12	62.80	73.42	79.80	78.84	77.10	95.58
SGE [66]	91.11	63.53	71.89	83.23	82.92	74.92	95.71
HSGE (pyr.)	91.11	65.29	75.32	**85.24**	**83.24**	78.73	**100.00**
HSGE (gen. pyr.)	92.22	67.94	75.50	83.36	81.73	**79.32**	**100.00**
HSGE (hier.)	**93.33**	67.06	**76.31**	82.85	81.33	72.03	**100.00**
HSGE (exhaus.)	92.22	**70.88**	76.58	83.84	82.12	73.90	**100.00**

In Table 6.2, we demonstrate the results obtained by the different configurations of our proposed hierarchical embedding applied to the labeled graph datasets. For comparing with other state-of-the-art methods, we have considered two additional techniques: (1) PATCHY-SAN (PSCN) [163] and (2) graphlet spectrum (GS) [127]. Some of the previously considered state-of-the-art techniques do not work with labeled graphs, so they have not been evaluated in this experimentation.

The results presented in Table 6.2 show that, except on the MUTAG dataset, our proposed hierarchical embedding techniques have achieved the best performances on all the other datasets. This demonstrates the usefulness of considering the hierarchical information for embedding graphs to a vector space. Contrary to the previous experiments on unlabeled datasets, in this case, the hierarchical configu-

ration has performed reasonably better. This fact shows that on labeled graphs, the hierarchical edges together with the connecting levels might provide important structural information. Also, it is important to note that the level information also performed consistently on all the datasets.

Table 6.2: Classification accuracy on *labeled* molecular graph datasets. In the table, MLK stands for the Multiscale Laplacian Graph kernel [126], DCNN indicates the Diffusion CNNs [11], PSCN corresponds to the PATCHY-SAN [163], GS denotes the Graphlet Spectrum (GS) [127], SGE corresponds to the Stochastic Graphlet Embedding (SGE) [66], and HSGE indicates the hierarchical graph embeddings proposed by us. The best results obtained on a dataset is specified by bold face.

Methods	MUTAG	PTC	PROTEINS	NCI1	NCI109	D&D	MAO
MLK [126]	87.94	63.26	–	81.75	–	78.18	88.29
DCNN [11]	66.98	56.60	–	62.61	–	OMR	75.14
PSCN [163]	92.63	62.90	–	78.59	–	77.12	–
GS [127]	88.11	–	–	65.00	–	–	–
SGE [66]	88.33	57.94	74.05	83.44	81.89	77.37	94.29
HSGE (pyr.)	91.11	62.06	75.68	**84.79**	82.03	77.46	94.29
HSGE (gen. pyr.)	**92.78**	65.59	**76.58**	81.31	80.24	79.66	**97.14**
HSGE (hier.)	91.11	**67.35**	75.77	82.50	82.88	79.32	94.29
HSGE (exhaust.)	91.67	66.18	76.04	84.42	**84.42**	**80.25**	**97.14**

6.5.2 Experiments on Pattern Recognition Datasets

While the datasets considered in the previous set of experiments were mostly molecular in nature, the set of experiments to be discussed in this section consider graphs from various fields, such as, biology, computer vision, graphics recognition and handwriting recognition. Underneath, we give a brief description of the datasets considered followed by the experimental setup, results and discussions.

In this experiment, we consider four different datasets; three of them *viz. AIDS*, *GREC* and *COIL-DEL* are taken from the *IAM graph database repository*[1] [180]. The last one called HistoGraph dataset[2] [207] consists of graphs representing words from the communicating letters written by the first US president, George Washington. All the datasets are carefully discussed in Appendix A.

Experimental Setup

In this case as well, we have employed a C-SVM solver [41] as a classifier. Since the datasets used in this set of experiments contain well defined train and test sets, we have reported the obtained accuracy on the test set of the respective datasets in Table 6.3.

[1] Available at `http://www.fki.inf.unibe.ch/databases/iam-graph-database`
[2] Available at `http://www.histograph.ch`

Results and Discussion

Similar to the experimental results obtained in the previous section, in this set of experiments our proposed hierarchical embeddings have achieved the best results on most datasets as well. In this set of experiments, the leading scores are mostly obtained by the exhaustive configuration, which shows the effectiveness of combining multi-scale structural information together with the hierarchical connections. For some datasets, our hierarchical embedding does not achieve the best results, but it has performed very competitively. This also proves the robustness of the hierarchical graph representation.

Table 6.3: Results obtained on the AIDS, GREC, COIL-DEL and HistoGraph datasets. In the table, RW corresponds to the Random Walk kernel [90], DE stands for the dissimilarity embedding [35], NAS indicates the node attribute statistics [93], AED denotes to the approximated graph edit distance [181], SGE corresponds to the Stochastic Graphlet Embedding [66], and HSGE indicates our hierarchical graph embeddings. The best results obtained on a dataset is indicated by bold face.

Methods	AIDS	GREC	COIL-DEL	HistoGraph					
				Keypoint	Grid-NNA	Grid-MST	Grid-DEL	Projection	Split
RW [90]	98.50	96.20	94.20	–	–	–	–	–	–
DE [35]	98.10	95.10	96.80	–	–	–	–	–	–
NAS [93]	98.30	99.20	98.10	–	–	–	–	–	–
AED [181]	–	–	–	77.62	65.03	74.13	62.94	**81.82**	80.42
SGE [66]	98.67	**99.62**	98.14	79.02	**72.73**	77.62	74.83	79.72	81.12
HSGE (pyr.)	98.87	99.43	98.79	79.02	**72.73**	77.62	74.83	79.72	81.12
HSGE (gen. pyr.)	98.35	99.43	98.37	77.62	72.03	77.62	74.13	79.72	81.45
HSGE (hier.)	98.33	99.05	**98.99**	79.02	70.63	76.22	**75.52**	80.42	80.42
HSGE (exhaust.)	**99.00**	99.43	98.86	**79.72**	72.03	**78.32**	74.83	**81.82**	**81.82**

6.5.3 Parameters Discussion

Our algorithm is mainly controlled by three different parameters: (1) the *number of levels* L of the graph pyramid, (2) the *reduction ratio* R and (3) the maximum *number of edges* T of a graphlet. For illustrating how these three parameters control the performance of the system, first we plot the classification accuracy by varying the levels of the graph pyramid (see Figure 6.3), reduction ratio (see Figure 6.4) and T (see Figure 6.5). Here it is worth mentioning that for the sake of simplicity, for each level we just consider the maximum accuracy obtained by any configuration mentioned in Section 6.2.2. From Figure 6.3, we observe that for all the datasets, considering a second level together with the base graph increases the classification accuracy. However, the successive inclusion of hierarchical levels does not always increase the performance. It has been observed that for smaller graphs (with less number nodes and edges; *e.g.* the graphs from MUTAG), the further inclusion of hierarchical abstraction decreases the performance of the system; this means that for smaller graphs a higher level abstraction can introduce noise or

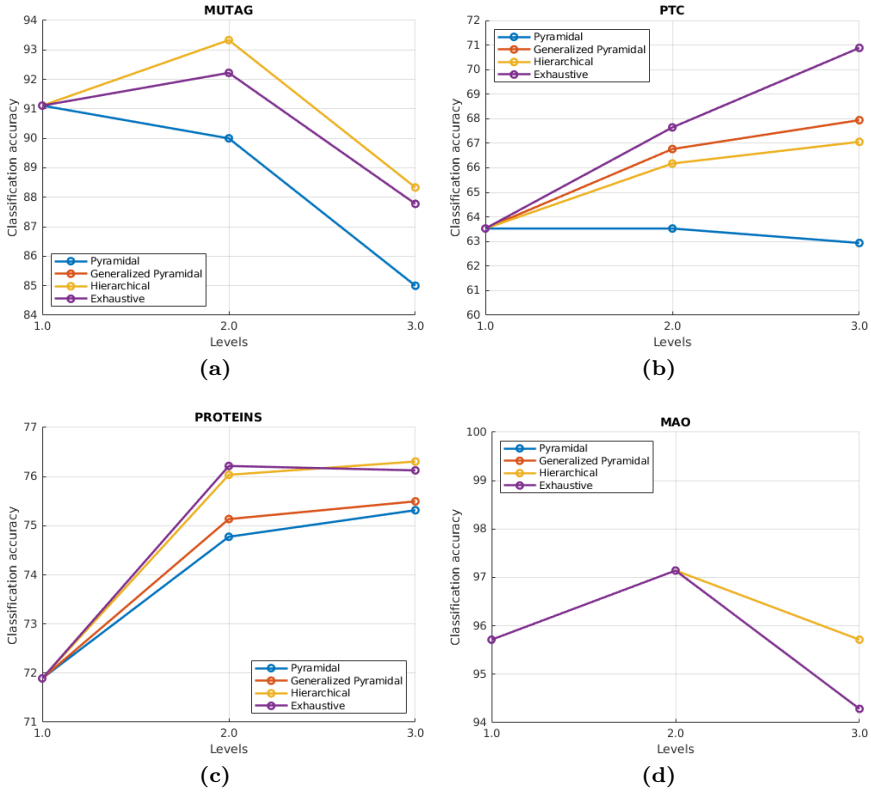


Figure 6.3: Plots showing classification accuracies by varying the levels of pyramidal graph construction on different datasets.

distortion. The reduction ratio R directly decides the number of clusters in a given level, and hence the number of nodes in the next higher level of the hierarchy. For example, $R = 1$ indicates that the number of clusters should remain the same with the number of nodes, while $R = 2$ indicates that the number of clusters should be half the number of nodes in that level. Figure 6.4 shows the behavior of our method with different values of R while we have fixed $L = 2$. From these plots, one must observe that R is completely dependent on the datasets irrespective of the size of graphs they contain. For PTC, PROTEINS, and MAO datasets, the performance mostly increases with the increase of R, while for MUTAG, it improves until $R = 2$, and then it decreases for all hierarchical configurations. For MAO dataset, all the hierarchical configurations behave exactly in the same way with the increase of R, which might be because the smaller sized graphs on which the contribution of different hierarchical configuration is indistinguishable.

In Figure 6.5, we show the performance trend on six datasets (*i.e.* MUTAG,

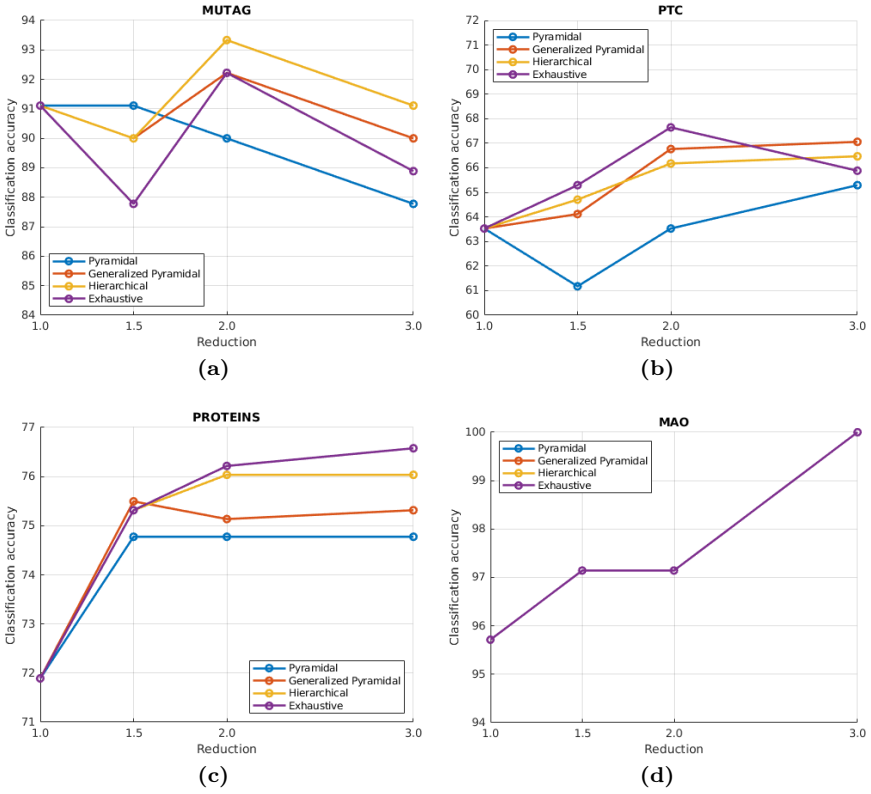


Figure 6.4: Plots showing classification accuracies by varying the reduction ratio of pyramidal graph construction on different datasets.

PTC, PROTEINS, NCI1, and NCI109) only with the SGE algorithm, which is the baseline graph embedding technique that we considered. The hierarchical configurations are not considered in this case because they have different graphlet size in different hierarchical levels, so understanding their behavior would have been complicated. From Figure 6.5, it is clear that increasing T mostly improves the performance of the system on all the datasets. Albeit, there are some exceptions (*e.g.* , for PTC dataset, $T = 6$), which suggests that graphlets with T edges are less informative for that particular graph dataset.

6.5.4 Discussion on the Stochasticity of the Algorithm

It is important to note that our proposed algorithm is stochastic in nature because of the involvement of the stochastic graphlet sampling and the subsequent graph embedding procedure. The graphlet sampling engaged here uniformly samples

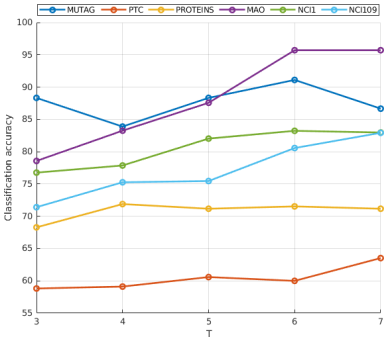


Figure 6.5: Plot showing the classification accuracy obtained by SGE by varying the maximum number of edges from 3 to 7 on different datasets: MUTAG, PTC, PROTEINS, MAO, NCI1, NCI109.

graphlets from a given population of graphs, and by the law of large numbers, this sampling guarantees that the empirical distribution of graphlets is asymptotically close to the actual distribution [170]. For demonstrating the fact that the stochastic behavior of our algorithm does not heavily impact on the experimental results, we repeated the last experiment on all the datasets considered for 10 iterations, and in each iteration, we randomly seeded the sampling algorithm. The mean and standard deviation of the classification accuracy obtained for each dataset is reported in Table 6.4. The mean accuracies reported in the table are quite close to the ones reported in Table 6.3, and the standard deviations are comparatively low (all of them are less than 1.0). This suggests that the proposed graph embedding technique, although employed a stochastic process, is consistent in terms of performance.

Table 6.4: Mean and standard deviation of the accuracies obtained by repeating the classification task on the AIDS, GREC, COIL-DEL and HistoGraph datasets for 10 iterations. Here the mean accuracies consistent with the ones in Table 6.3 and the low standard deviations show that the proposed graph embedding is not sensitive to the stochasticity involved in the algorithm. The best results obtained on a dataset is specified by bold face.

Methods	**AIDS**	**GREC**	**COIL-DEL**	**HistoGraph**					
				Keypoint	**Grid-NNA**	**Grid-MST**	**Grid-DEL**	**Projection**	**Split**
HSGE (pyr.)	**98.74** (±0.13)	99.36 (±0.19)	98.74 (±0.21)	78.98 (±0.32)	**72.71** (±0.10)	77.57 (±0.43)	74.79 (±0.62)	79.72 (±0.99)	81.04 (±0.84)
HSGE (gen. pyr.)	98.12 (±0.27)	99.58 (±0.23)	98.49 (±0.49)	**79.31** (±0.52)	71.28 (±0.58)	78.05 (±0.47)	74.96 (±0.71)	79.94 (±0.18)	80.24 (±0.74)
HSGE (hier.)	98.24 (±0.36)	99.04 (±0.16)	**98.98** (±0.60)	79.03 (±0.20)	70.51 (±0.55)	76.20 (±0.40)	**75.47** (±0.86)	80.39 (±0.17)	80.38 (±0.21)
HSGE (exhaust.)	**98.74** (±0.21)	**99.64** (±0.80)	98.84 (±0.17)	79.01 (±0.70)	71.96 (±0.10)	**78.28** (±0.97)	74.79 (±0.01)	**80.82** (±0.46)	**81.53** (±0.94)

6.6 Conclusions

In this chapter we have proposed to enhance the information encoded in graph embeddings by means of hierarchical representations. We have experimentally validated that the abstract information is able to improve the graph classification performance.

The embedding function is based on a stochastic sampling of graphlets to obtain the graphlet distribution within the graph. Graphlets of different sizes are considered to allow a change on the node context. Moreover, the hashing functions are used to identify graphlets in an efficient way. Event though considering different size graphlets provides robustness in terms of graph distortions, they still provide local information when we consider larger graphs. Therefore, building a graph hierarchy allows to increase the graphlet context without increasing the time needed for identifying the graphlet. In this work, we have carefully validated the performance of our approach in different application scenarios, showing that we outperform the state-of-the-art approaches in the graph classification task using an SVM as a classifier.

Further research on improving the hierarchical graph construction may lead to a better embedding approach. Even though the Girvan-Newman algorithm is able to exploit the desired properties of the graph, generating clusters that allow to create good abstractions, their time complexity is a drawback that should be studied when considering large graphs.

Even though the achieved results are state of the art on the proposed task, we are still dealing with hand-crafted approaches that may not be realistic in a real scenario. With the emergence of deep learning, it has been proved the success of this kind of models against handcrafted statistical methods. Moreover, nowadays, these new approaches are being extended to non-Euclidean data such as graphs or manifolds. These new family of methods is called *Geometric Deep Learning*.

Part II

Geometric Deep Learning

Geometric deep learning has emerged as the generalization of deep neural models to non-Euclidean data, *i.e.* graphs and manifolds. These novel techniques, have unleashed a world of possibilities when dealing with structural representations. However, these approaches were not exploited on DIAR tasks yet. In this dissertation, we explore two models able to provide solutions to several tasks. Moreover, we propose to combine traditional approaches overviewed in the first Part with this new field.

7 | Geometric Deep Learning

We shouldn't be looking for heroes, we should be looking for good ideas.

– Noam Chomsky

This chapter introduces the concept of geometric deep learning. This novel framework has leveraged the benefits of deep learning to structured graph data. Here, we present the relevant literature works on geometric deep learning starting from node embeddings, where the structure is just indirectly taken into account, to graph neural networks, which are able to inject the structure at the same time as the node features. Finally, we review the main applications on computer vision.

7.1 Geometric Deep Learning

For many years, deep learning has been proposed for a boundless number of tasks and applications taking into account different input types. However, none of these tasks where able to generalize to non-Euclidean input data.

- **Convolutional Neural Networks** (CNN). This set of methodologies have become the standard tool when dealing with n-dimensional grids. Some examples of their application are image classification [129], image generation [100] or 3D deep learning [140]. Even though, the adjacency matrix of a given graph is represented as a 2-dimensional grid, the local neighborhood of each entry of the matrix is meaningless and do not encode any spatial or local relationship between rows or columns. Therefore, CNN methodologies, which exploit these local relationships, are not able to deal with graph data.

- **Recurrent Neural Netoworks** (RNN). These approaches are well-defined over sequential data. They have demonstrated to be state-of-the-art for image captioning [235], video frame classification [243] and handwritten text recognition [116]. However, in general, graphs cannot be serialized in a sequence of nodes and their correspondent edges.

In this context, *Geometric Deep Learning*[1] (GDL) has emerged as a generalization of deep learning methods to non-Euclidean domains such as graphs and manifolds [30]. This field has generated much attention in the recent years allowing the developed models to encode structural and relational data. Several fields have benefit from this new paradigm, for instance computer vision [239], quantum chemistry [95] and computer graphics [152] among others.

Following the same paradigms as other machine learning approaches, graph-based learning methods, are trained in a supervised, unsupervised or reinforcement learning manner. As a matter of fact, these training paradigms are both able to process the graphs as a whole, but also interpret its individual compounding parts. Thus, among others, some common tasks of the graph domain are: node classification, relation prediction, community detection and graph classification.

7.2 Node Embeddings

Several strategies have been proposed with the objective of encoding graph nodes into a vectorial representation, *i.e.* numerical features, by means of embedding functions. The main objective is to encode local structural graph information for each node into a feature vector. Node embeddings can be later used for tasks such as node classification in terms of local structural contexts.

The first approaches enforce structural constraints in the embedding space. For instance, Weston *et al.* [230] constrain the learned node embeddings to be close or distant depending on the presence of edges between the corresponding nodes. Thus, during training, the objective function is based on both, a node classification loss and a similarity constraint. Later, Tang *et al.* [211] proposed a model named LINE able to decide the first and second order similarities from the node embeddings.

Alternatively to augment the objective functions with the desired structural properties, Perozzi *et al.* [168] and later Grover *et al.* [102] proposed to learn some structural features based on random walks, namely DeepWalk and Node2Vec respectively. The idea behind these approaches is that given a structural feature representation, the model should be able to easily predict the nodes that surround it. So far, these methods allow a fully unsupervised training as they do not depend on any prior knowledge of the node information. Yang *et al.* [240] proposed an extension called Planetoid which incorporates node labels in their training framework. Basically, they propose to sample positive and negative pairs of nodes based on both random walks and node labels.

Until now, in the reviewed approaches, the graph structure was only injected indirectly taking only into account node information. The structural information, provided by the edges, was still disregarded. In order to take full advantage of the

[1] `http://geometricdeeplearning.com/`

structural information, edges have to be leveraged as well as nodes features.

7.3 Graph Neural Networks

Graph Neural Networks (GNN) were first proposed by Gori and Scarselli [101, 195] as the first attempt to generalize neural networks to graphs. Later, Bruna *et al.* [32] proposed the first formulation of CNNs working on the graph spectral domain. However, the ability to process graph data came with a huge time complexity becoming inappropriate for real cases. Afterwards, the works of Henaff *et al.* [106], Defferrard *et al.* [54] and Kipf *et al.* [123] addressed these computational drawbacks.

In its simplest form, a GNN layer is defined as,

$$h^{(k+1)} = \rho\left(\sum_{B\in\mathcal{A}^{(k)}} Bh^{(k)}\Theta_B^{(k)}\right) \tag{7.1}$$

where $h^{(k)}$ is the node hidden state at the k-th layer, ρ is a non-liniarity such as $\mathrm{ReLU}(\cdot)$, $\mathcal{A}$ is a set of graph intrinsic linear operators that act locally on the graph signal such as the adjacency matrix and Θ_B are learnable parameters. The set of graph intrinsic linear operators are able to handle multi-relational graphs, however, in most of the cases, $\mathcal{A}$ only contains the adjacency matrix. Taking into account the self connections, this equation can be reformulated by considering $\mathcal{A} = \{I, A\}$ where I and A are the identity and adjacency matrix. Thus, separating the self connection information from the neighborhood, it is defined as

$$h^{(k+1)} = \rho\left(Ih^{(k)}\Theta_{\text{self}}^{(k)} + Ah^{(k)}\Theta_{\text{neigh}}^{(k)}\right) = \rho\left((A+I)h^{(k)}\Theta^{(k)}\right). \tag{7.2}$$

Several recent works have proposed different architectures. In general, these learning architectures on graphs are divided in two groups, spatial and spectral domain respectively.

- **Spatial domain** methods extend the idea of CNN at the image domain by moving a filter across nodes and applying a set of operations involving the local neighborhood to compute a new hidden representation [67, 139].
- **Spectral domain** architectures take advantage of the spectral graph theory [204] in order to generalize the convolution operation to arbitrary graphs by means of the graph Laplacian [54, 123].

Recently, Gilmer *et al.* [95] proposed an approach named *Message Passing Neural Networks* (MPNNs) as a general learning framework for graphs. This approach is able to generalize the aforementioned methodologies to have a common pipeline.

Therefore, they redefine the previous spatial and spectral architectures using the MPNN framework proposing two phases: *message passing* and *readout*. This is defined using three arbitrary differentiable functions $M^{(k)}(\cdot)$, $U^{(k)}(\cdot)$ and $R(\cdot)$.

In the message passing phase, the hidden state $h_v^{(k)}$ of each node v is updated by a node update function $U^{(k)}(\cdot)$ which receives a message $m_v^{(k+1)}$ collected from the neighboring nodes by means of the message function $M^{(k)}(\cdot)$. This phase is repeated during K time steps and is defined as

$$m_v^{(k+1)} = \sum_{u \in \mathcal{N}(v)} M^{(k+1)}(h_v^{(k)}, h_u^{(k)}, e_{vu}), \tag{7.3}$$

where $h_u^{(k)}$ and $h_v^{(k)}$ are the hidden states of nodes v and u at iteration k and $\mathcal{N}(v)$ denotes the neighbors of v in the graph g. Afterwards, the hidden state of node v is updated according to the message $m_v^{(k+1)}$.

$$h_v^{(k+1)} = U^{(k+1)}(h_v^{(k)}, m_v^{(k+1)}). \tag{7.4}$$

In the literature, the message passing step is sometimes written as update and aggregation, following the notation introduced in Equation 7.1, message passing updates are formally expressed as,

$$h_u^{(k+1)} = \textsc{Update}^{(k+1)}\left(h_u^{(k)}, \textsc{Aggregate}^{(k+1)}\left(\left\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right), \tag{7.5}$$

where $\textsc{Update}^{(k)}$ and $\textsc{Aggregate}^{(k)}$ are also arbitrary differentiable functions.

The message passing phase gathers structural information from the graph and embeds this information as node labels. Thus, the node features can now be used for node level tasks such as node classification.

Moreover, the readout phase computes a feature vector for the whole graph based on the set of hidden states of the nodes. Basically, it aggregates the information across all nodes for graph classification purposes. Hence, the readout function $R(\cdot)$ must be permutation invariant as the order of the nodes is not important in a graph. It is formally defined as,

$$\hat{y} = R(\{h_v^{(K)} | v \in V\}). \tag{7.6}$$

This final function is also known as global pooling layers.

Among others, some relevant GNN formulations proposed in the literature are,

Graph Convolutional Networks (GCN) [123]. Motivated by the spectral graph theory, Kipf *et al.* proposed to take advantage of a first order approximation of a spectral graph convolution. A GCN layer is formally defined as,

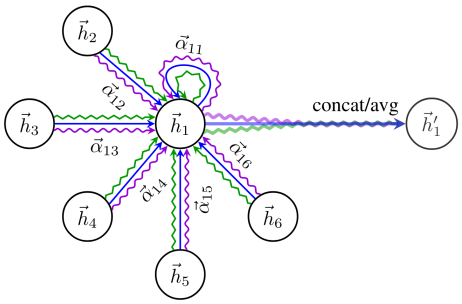


Figure 7.1: Illustration of a 3-head attention for node 1. Each arrow denote an independent attention computation. Reprinted from [217].

$$h_u^{(k+1)} = \rho\left(\left(\sum_{v\in\mathcal{N}(u)} \frac{h_v^{(k)}}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}\right)\Theta^{(k)}\right). \tag{7.7}$$

Note that this formulation applies a neighborhood normalization which gives less weight to these messages coming from nodes with higher degrees. Previously, Defferrard *et al.* [54] proposed a similar approach which is not restricted to the first order approximation but to a given parameter.

Graph Attention Networks (GAT) [217]. Veličković *et al.* proposed to learn attention weights during the message phase. In that sense, the aggregation is able to weight the neighbors according to a learned importance. Thus, a GAT layer is defined by

$$h_u^{(k+1)} = \rho\left(\sum_{v\in\mathcal{N}(u)} \alpha_{uv} h_v^{(k)}\Theta^{(k)}\right), \tag{7.8}$$

where α_{uv} are the attention weights computed following the idea proposed by Bahdanau *et al.* [14],

$$\alpha_{uv} = \frac{\exp(s_{uv})}{\sum_{w\in\mathcal{N}(u)}\exp(s_{uw})}, \tag{7.9}$$

where $s_{uv} = a_\theta\left(h_u^{(k)}, h_v^{(k)}\right)$ and a_θ is a shared attention mechanism with learnable weights θ. Moreover, following the Transformer [216], GAT is able to employ multiple attention heads that are then combined via, for example, concatenation. Figure 7.1 illustrates the multi-head attention defined by GAT networks.

Gated Graph Neural Networks (GG-NN) [139]. Li *et al.* proposed to formulate the update function in terms of *Recurrent Neural Networks* (RNN) models. In this

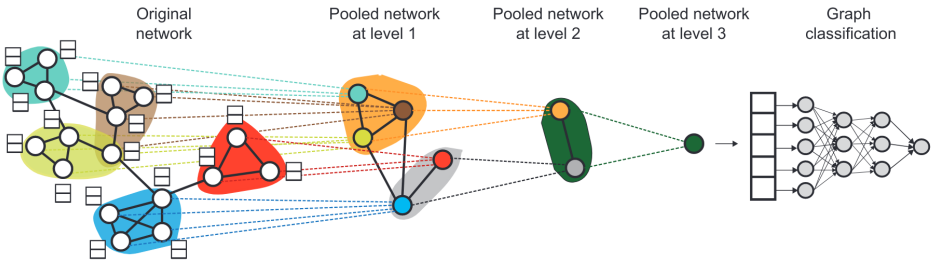


Figure 7.2: Overview of the DiffPool methodology. Reprinted from [242].

setting, the RNN hidden state becomes the node hidden state whereas the RNN input becomes the aggregated message. In the particular case of GG-NN, the RNN model is a Gated Recurrent Unit (GRU) [45].

Two important concepts arise according to the neighbor contributions are whether the models are isotropic or anisotropic. On the one hand, in the isotropic update schemes, each neighbor contributes equally to the update of the hidden state of the central node as it does not consider the edge direction information. Some works introducing isotropic schemes are GCN [123], GraphSAGE [103] and *Graph Isomorphism Networks* (GIN) [236]. On the other hand, anisotropic models exploit the edge directions by weighting the contribution of each neighbor by means of different mechanisms such as attention or gates. Important works on anisotropic models are GAT [217], *Gaussian Mixture Model Networks* (MoNet) [156] and *Gated Graph Convolutional Networks* (GatedGCN) [26].

Finally, some architectures have explored the idea of pooling in the sense it is used in CNNs. Hence, to encode hierarchical information in the forward pass of the GNN. This is done by means of graph clustering techniques. Recently, Ying *et al.* [242] proposed a differentiable pooling. They propose to learn a differentiable soft cluster assignment at each layer allowing the network to obtain a coarsened representation for the next layer. Figure 7.2 shows an overview of this architecture.

Given these set of approaches, now, we can make use of the deep learning tool box for graph data. Thus, elements defined initially for CNN are now available, for instance, Batch normalization layers or skip connections, *i.e.* ResNets, Highway Nets and DenseNets. The literature on graph neural networks and their applications is quite large and continuously evolving. We refer the interested readers to recent survey paper for a comprehensive overview of these methodologies [19, 30, 233, 246]. Very recently, Dwivedi *et al.* [68] presented a reproducible benchmarking framework. They introduce six different datasets to evaluate four different tasks that are usually performed on graph data. Moreover, they present an exhaustive experimental comparison of these methods as well as some very intuitive diagrams illustrating the main difference between the main GNN layers.

7.4 Geometric Deep Learning in Computer Vision

Geometric deep learning has provoked a paradigm shift in several computer vision applications. Among others, here we introduce some applications that have benefit of this novel framework in the context of computer vision.

3D vision. 3D objects are usually represented in terms of meshes which is a graph-based shape representation. Thus, several works have proposed to exploit these representations. For example, Smith [205] proposed a 3D object reconstruction from images generating adaptive meshes.

Scene graph representations. Scene graphs have emerged as an important step for image understanding. In this setting, not only the objects should be detected, but also the relations between these objects are detected. Several works have been proposed making use of these representation, for example image retrieval using scene graphs or scene graph generation [9, 111, 234, 239].

Knowledge graphs. External knowledge in terms of knowledge graphs has also benefited from GNN. For example, Shen *et al.* [200] exploited the semantic relations between labels in order to incorporate external knowledge in the learning process of a zero-shot sketch image hashing. Thus, a GNN is used in order to learn a good binary signature able to incorporate information about the semantic space.

Few-shot learning. Some approaches have been proposed for the task of few-shot learning. These frameworks benefit from the GNN paradigm in order to propagate information among the support set. Thus, the task is formulated in terms of node or edge classification. Moreover, these frameworks are easily extended to semi-supervised or active learning settings [89, 121].

8 | Learning Graph Distances

We know not through our intellect but through our experience.

– Maurice Merleau-Ponty

The emergence of geometric deep learning as a novel framework to deal with graph-based representations has faded away traditional approaches in favor of completely new methodologies. In this chapter, we propose a new framework able to combine the advances on deep metric learning with traditional approximations of the graph edit distance. Hence, we propose an efficient graph distance based on the novel field of geometric deep learning. Our method employs a message passing neural network to capture the graph structure, and thus, leveraging this information for its use on a distance computation. The performance of the proposed graph distance is validated on two different scenarios. On the one hand, in a graph retrieval of handwritten words i.e. *keyword spotting, showing its superior performance when compared with (approximate) graph edit distance benchmarks. On the other hand, demonstrating competitive results for graph similarity learning when compared with the current state-of-the-art on a recent benchmark dataset.*

8.1 Introduction

As it has been introduced in Chapter 2, when dealing with graphs, an important property is the ability to compare them in terms of a similarity or distance. This operation, which is trivial when considering feature vectors defined in $\mathbb{R}^n$, is not properly defined in the graph domain [50, 85]. Due to the inherent graph flexibility, it forces us to adopt some definitions of dissimilarity (similarity) ad hoc to particular purposes. This problem was introduced by Borgwardt [22] and formally defined in Definition 2.2.1.

Lots of efforts have been made in this direction. In the literature, error-tolerant or inexact graph matching algorithms have been proposed and already reviewed in Chapter 2. Let us remind about the graph edit distance (GED) [88] which is one of the most popular error-tolerant graph matching methods. In this case, the graph comparison problem is formulated in terms of finding the minimum transformation cost in such a way that an isomorphism exists between the transformed graph g_1

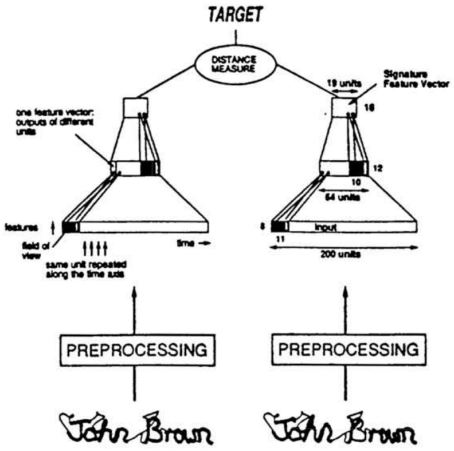


Figure 8.1: First siamese architecture proposed for signature verification. Reprinted from [29].

and the second one g_2.

The main drawback of the techniques to compute the GED is that the time complexity is exponential in terms of the number of nodes of the input graphs. Hence, GED computation is unfeasible in a real scenario, where there may be no constraints in terms of the graph size. Therefore, several algorithms have been proposed to cope with this complexity [82, 181]. However, these approximate algorithms only consider very local node structures in their computation and they do not adapt their costs according to the problem being addressed.

Inspired by this efficient GED approximations, and the powerful framework provided by the new advances in geometric deep learning, reviewed in Chapter 7, we propose to leverage its effectiveness as a learning framework to enhance our graph distance computation. Therefore, we are facing a graph metric learning problem. It can be formulated as a contrastive learning problem that finds contrast between similar and dissimilar objects. A siamese architecture is suitable for this problem. Bromley *et al.* [29] proposed it for signature verification. Siamese networks make use of the same model and weights on two separated branches in order to learn a representation where distances can be computed. Figure 8.1 shows the first siamese architecture introduced for metric learning. Later, several approaches have extended this idea, being triplet loss [228] one of the most successful methods. Recently, novel approaches have focused on extending this concept in order to exploit groups of samples instead of pairs or triplets [70]. Moreover, contrastive learning has been used, not only as a metric learning framework but it has also raised some attention due to its astonishing improvement on unsupervised learning tasks [43].

To the best of our knowledge, our paper [176] was the first work that introduced the idea of learning a graph metric by means of message passing architectures. In this chapter, we propose a triplet learning framework for the graph metric learning problem. In our proposed approach an enriched graph representation is learned by means of a graph neural network. In addition, our proposed distance is based on the Hausdorff Edit Distance introduced by Fischer *et al.* [82] as an efficient approximation of the real graph edit distance. In comparison, our framework has the ability to enrich the initial graph representation by means of message passing stages which learns the edit cost operations. Furthermore, insertion and deletion costs, are dynamically learned according to the node local context. Therefore, we avoid a costly manual process on setting the edit cost operations per each specific problem. The proposed approach is validated using standard graph datasets for keyword spotting and object classification. In this application scenario, the proposed approach based on a graph neural network shows competitive results demonstrating the efficacy of our learning framework.

The rest of this chapter is organized as follows. Section 8.2 introduces the related work on graph neural networks for graph metric learning. Section 8.3 and 8.4 proposes our learning framework and the corresponding learning strategy. Section 8.5 evaluates the proposed approach in two different scenarios. Finally, Section 8.6 draws the conclusions and future work.

8.2 Related Work on Graph Metric Learning

As mentioned in the introduction, learning an appropriate distance or similarity between data points has been tackled by the area of metric learning. Neural networks have been widely used as the learning framework for similarity problems. Promptly, siamese neural networks were adopted as a family of neural networks consisting of two identical networks with shared weights for similarity learning. For example, Baldi *et al.* [17] makes use of a siamese model for fingerprint recognition. In the particular application field of DIAR tasks, Bromley *et al.* [29] presented a siamese architecture for signature verification. Siamese neural networks use pairs of samples to train with positive and negative examples *i.e.* being similar or not. Later, several approaches have extended this idea in order to take always into account a positive and a negative example. These approaches are known as triplet networks [228]. In this case, three identical networks with shared weights are used to bring similar examples together and dissimilar examples to be far apart.

Inspired on these works, several papers appeared extending these ideas to the graph domain. Thus, we proceed to review a handful of approaches facing the graph similarity learning problem. Li *et al.* [138] presented two different models to solve the graph similarity problem, on the one hand, a graph embedding model, and on the other hand, a graph matching network. Both models can be trained with pairs or triplets. Let us briefly review each one of these models.

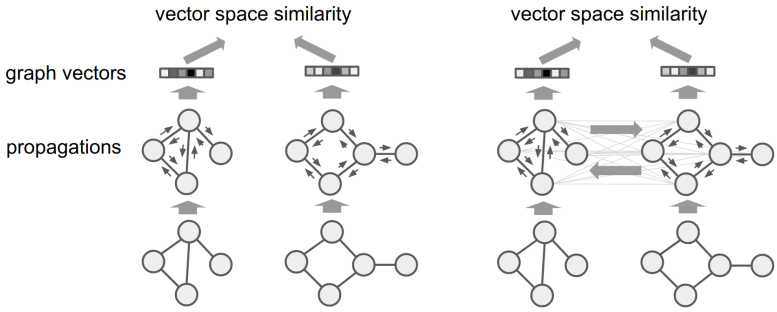


Figure 8.2: Illustration of the two models proposed by Li *et al.* [138]. Reprinted from [138].

- **Graph embedding model**: This model, illustrated in Figure 8.2 (left), takes advantage of siamese GNN's to embed the given graphs into a vectorial space. Then, given the pair of vectorial representations, a similarity metric in the vector space can be computed by means of the Euclidean, cosine or Hamming similarities. The model is trained with a margin-based pairwise loss or a margin-based triplet loss respectively. Very similar approaches have also been presented in other works, for instance Chaudhuri *et al.* [42] who trained a similar approach with contrastive loss or the work introduced by Zhang *et al.* [244] which uses the L_2 loss to mimic the real similarity score.
- **Graph matching networks (GMN)**: Similarly to the previous architecture, two GNNs with shared weights process the input graphs. However, in this case, the authors propose to modify the node update module in order to take into account not only the aggregated messages on the edges of each graph, but a cross-graph message which measures how well the nodes match from one graph to the other. Finally, following the same idea as the previous model, each graph is finally converted into a vectorial representation which is later used in a similarity metric and trained with the same loss. Figure 8.2 (right) presents the proposed architecture. Similarly, Wang *et al.* [225] also proposed a cross-graph convolution for their model, however, to train this model, they propose to use the cross-entropy loss at node-to-node level.

Another interesting approach, namely SimGNN, was proposed by Bai *et al.* [15]. In this work, the authors proposed to combine graph-level embeddings and node-node similarity scores by taking their histogram of features. However, as the histogram function is not differentiable, this methodology still relies on the graph-level embedding for computing the final similarity score. Their model is trained according to the real graph edit distance for small datasets whereas the smallest distance computed by three well-known approximate algorithms is taken to handle large datasets. The authors extended this work by proposing a new model named GraphSim [16]. In this architecture, only three node-node similarities scores are

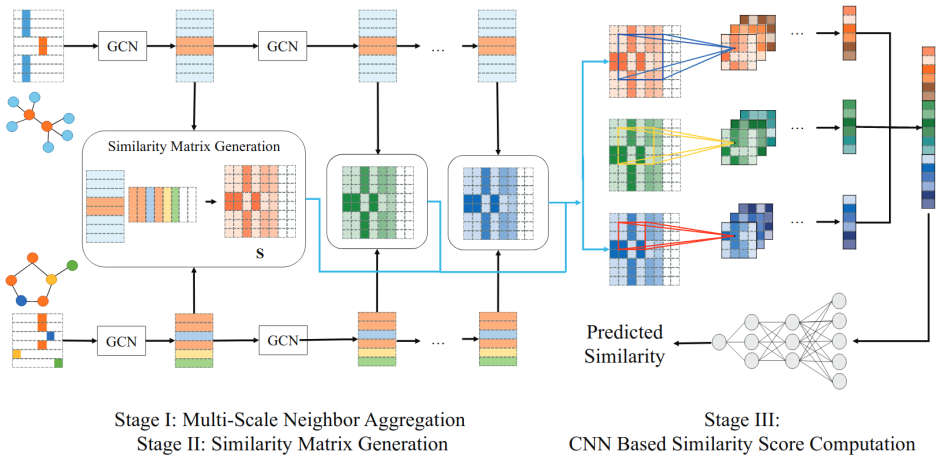


Figure 8.3: Illustration of the GraphSim model proposed by Bai *et al.* [16]. Reprinted from [16].

used corresponding to node embeddings at different scales. After that, the similarity matrices are treated as images and a CNN is used to process them to discover the optimal node matching pattern. However, to deal with the permutation invariant ordering of graph nodes, they propose a BFS ordering. It also allows the use of CNN as they claim that the required spatial locality performs properly. Moreover, the similarity matrices are first padded to $\max(|V_1|, |V_2|)$, where V_1 and V_2 are the node sets of the graphs involved and resized to meet the expected size. Finally, following the SimGNN training, a precomputed similarity score is used to lead the training. Figure 8.3 overviews the proposed GraphSim architecture.

Compared to these works, our model makes use of a node-node distance matrix to obtain a global graph distance metric. Therefore, we are not obtaining a global vectorial representation of our graphs nor applying cross-convolution layers in our graph neural network architecture. This allows us to make use of any differentiable graph or set distance, which preserves the permutation invariance property, while avoiding the computational overhead of the cross-convolution layers. Moreover, we avoid the loss of structural information of other approaches when obtaining a vectorial graph representation by explicitly dealing with the structure in the distance itself.

8.3 The Learned Graph Distance Framework

This section is devoted to present our proposed learning framework for graph distance. The proposed model learns the Hausdorff edit distance, proposed by Fischer *et al.* [82]. As a learning setting, the proposed architecture can be trained

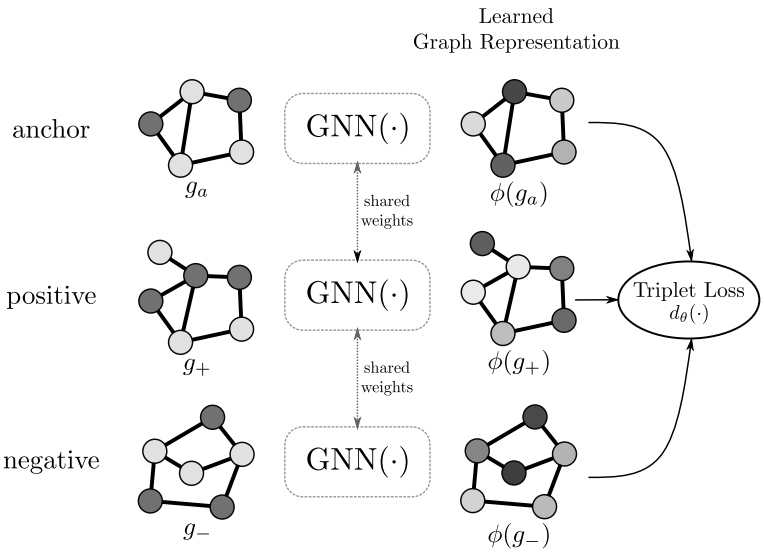


Figure 8.4: Overview of our distance learning framework. Given a triplet of graphs (g_a, g_+, g_-) as the anchor, positive and negative samples respectively, the GNN $\phi(\cdot)$ learns a graph representation per each one $(\phi(g_a), \phi(g_+), \phi(g_-))$, which are matched by means of a learned distance $d_\theta(\cdot)$.

either with pairs or triplets of graphs. Hence, as ground-truth, only the information on whether or not two graphs belong to the same class is required. Note, that in our proposed approach, we do not require the node correspondence information nor the real graph edit distance. Instead, the node assignment is implicitly learned by our system. Moreover, the edit costs for both insertions, deletions and substitutions are also learned by our framework. Therefore, we do not require to manually tune these parameters following the traditional pipeline. Even though our framework can be trained using pairs, in this work we will focus on the triplet setting. Therefore, we will make use of three GNN with shared weights.

Figure 8.4 shows a graphical outline of our proposed approach. Our pipeline is divided in two stages. Firstly, a graph neural network $\phi(\cdot)$ is used to obtain a node-level embedding which codifies the local context information, in terms of structure, for each node. Secondly, a novel graph similarity algorithm based on the Hausdorff edit distance is proposed as a technique to compare two graphs $d_\theta(\cdot)$. Observe that the graph similarity can be replaced by any differentiable graph distance approach.

Each stage is carefully described in the following sections. First, the GNN architecture is explained in detail. Afterwards, the proposed graph similarity is developed.

8.3.1 Learning Node Embeddings

The first stage of our framework is a graph neural network architecture $\phi(\cdot)$ able to learn a new graph representation in terms of node embeddings. Our architecture consists of K propagation layers that map the input graph to an enriched representation. Thus, each propagation layer takes a set of node representations at layer k, $\{h_i^{(k)}\}_{i \in V}$ and maps it to a new node representation $\{h_i^{(k+1)}\}_{i \in V}$ at layer $k+1$. We evaluate the following two different architectures according to two different message passing strategies.

GAT-based model: This model uses graph attention networks (GAT), introduced by Veličković *et al.* [217]. We have already described this model in Section 7.3. It provides an anisotropic update scheme by means of its attention weights. Moreover, a multi-head attention has been used to enrich the model capacity and to stabilize the learning process. In our setting, we use these layers with residual connections, four attention heads and BatchNorm layers [110] with the exception of the last layer. The attention heads are concatenated at the intermediary layers and averaged for the final layer.

GRU-based model: This architecture is based on the gated graph neural networks (GG-NN) proposed by Li *et al.* [139]. Originally, the message function is formulated as

$$M(h_v^{(k)}, h_u^{(k)}, e_{vu}) = A_{e_{vu}} h_u^{(k)},$$

where $A_{e_{vu}}$ is a learned matrix for each possible edge label. Note that we are restricted to a discrete set of labels. In order to overcome this constrain, Gilmer *et al.* [95] proposed to use a modified message function defined as

$$M(h_v^{(k)}, h_u^{(k)}, e_{vu}) = A(e_{vu}) h_u^{(k)},$$

where $A(e_{vu})$ is a neural network which maps the edge vector to a matrix $d \times d$. This modification allows the use of non-discrete information as edge attributes. Finally, the update function is defined as

$$U(h_v^{(k)}, m_v^{(k+1)}) = \text{GRU}(h_v^{(k)}, m_v^{(k)}),$$

where GRU is the Gated Recurrent Unit [45]. In its original formulation, the first node hidden state is padded with zeros to meet the size defined by the GRU, however, we propose to use a fully-connected layer as a first node embedding. Moreover, we propose to incorporate edge features according to the source and destination node according to

$$e_{vu} = \text{MLP}(|h_v^{(1)} - h_u^{(1)}|)$$

as proposed in [89]. There, MLP stands for multi-layer perceptron and the absolute value is used to preserve the symmetry of the edge direction.

8.3.2 Graph Distance or Similarity

Following the idea of the Hausdorff edit distance (HED) defined in Equation 2.2, we defined the set of edit operations in terms of insertions, deletions and substitution. Moreover, we propose to dynamically adapt the insertions and deletion costs according to the application domain. With this aim, we introduce two learnable costs $c(\varepsilon \to v)$ and $c(u \to \varepsilon)$ for the insertion and deletion of nodes. Thus, taking advantage of the computed node embeddings, our nodes are enriched with information aggregated from their local context and, therefore, its importance within the graph. Thus, we propose to take advantage of this in order to define two neural networks $\varphi_i(\cdot)$ and $\varphi_d(\cdot)$, defined as $\varphi_*\colon \mathbb{R}^n \to \mathbb{R}^+$, able to decide the corresponding cost of this operation. In our experiments, $\varphi_i(\cdot)$ and $\varphi_d(\cdot)$ are the same network with shared weights as we consider the insertion and deletion operations to be symmetric. In addition, we take the absolute value as the insertion and deletion costs must be positive.

Therefore, we define the distance between two graphs $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ as

$$d_\theta(g_1, g_2) = \frac{1}{|V_1| + |V_2|} \left(\sum_{u \in V_1 \cup \{\varepsilon\}} \min_{v \in V_2 \cup \{\varepsilon\}} c_\theta(u, v) + \sum_{v \in V_2 \cup \{\varepsilon\}} \min_{u \in V_1 \cup \{\varepsilon\}} c_\theta(u, v) \right), \tag{8.1}$$

where θ are learnable parameters and $c_\theta(\cdot, \cdot)$ is the corresponding learnable cost function defined as

$$c_\theta(u, v) = \begin{cases} \varphi_d(h_u^{(K)}; \theta) & \text{if } (u \to \varepsilon) \text{ is a deletion,} \\ \varphi_i(h_v^{(K)}; \theta) & \text{if } (\varepsilon \to v) \text{ is an insertion,} \\ \frac{d(h_u^{(K)}, h_v^{(K)})}{2} & \text{otherwise.} \end{cases} \tag{8.2}$$

In our scenario, the edges are not taken into account as we consider the local structures to be already encoded during the message passing phase. However, edges can be incorporated to Equation 8.2, considering the adjacent edges as nodes and applying the same distance $d_\theta(\cdot)$ with different learned weights.

Observe that an important aspect of the proposed distance is the fact that the node correspondence might not be symmetric. Figure 8.5 illustrates this issue, moreover, we also show the effect of considering insertions and deletions as a ε node in Figure 8.5(b). Note that not considering ε nodes as proposed in [176] and illustrated in Figure 8.5(a), constrains the learning capabilities of our framework.

A limitation of our approach is that in some scenarios it might lose some node feature information in favor of encoding the local node structure. For this reason,

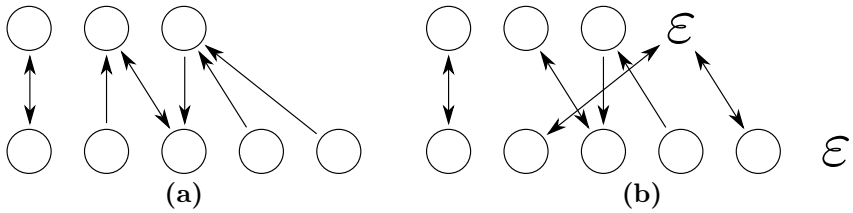


Figure 8.5: Assignment problem according to the proposed distance. (a) only substitutions are considered, (b) insertions and deletions are included as an extra epsilon node.

we optionally combine the original graph information into Equation 8.2, that is redefined as,

$$c_\theta(u,v) = \begin{cases} \tau_d + \varphi_d(h_u^{(K)};\theta) & \text{if } (u \to \varepsilon) \text{ is a deletion,} \\ \tau_i + \varphi_i(h_v^{(K)};\theta) & \text{if } (\varepsilon \to v) \text{ is an insertion,} \\ \frac{d(h_u^{(K)},h_v^{(K)})+d(u,v)}{2} & \text{otherwise,} \end{cases} \tag{8.3}$$

where $\tau_d, \tau_i > 0$ are user-defined parameters to fix the minimum cost for node deletion and insertion respectively. Moreover, $d(u,v)$ corresponds to the distance computed on the original node attributes, for instance, the (x,y)-coordinates. We find this setting helpful to avoid incorrect matchings between nodes that share structurally similar neighborhoods.

8.4 Training Setting and Learning Objective

In this work, we follow the idea of triplet networks to exploit the ranking properties of the desired metric. Thus, we use three GNN models with shared weights following the architecture illustrated in Figure 8.4.

The model is trained in a supervised manner, so we know which pairs of graphs belong to the same class. Compared to other approaches, we do not require node assignments nor a pre-computed similarity score. All models were trained using the *Adam* optimizer [122] with weight decay *i.e.* L_2 regularization. The learning rate of 0.001 is multiplied by 0.95 every 5 epochs to decrease its value, and we applied early stopping to finish our training process.

The objective function to minimize is the triplet loss, also known as margin ranking loss. This learning objective receives three samples in which we already know its ranking, *i.e.* which pair should have a higher similarity score or distance. Let $\{g_a, g_+, g_-\}$ be a triplet training sample where, g_a is the anchor graph, g_+ is a positive graph sample *i.e.* a sample different from g_a but belonging to the same class and g_- is a negative graph example *i.e.* a sample belonging to a different

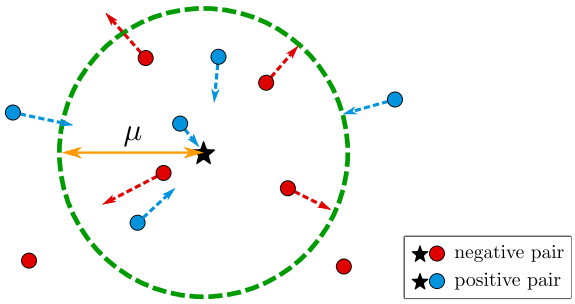


Figure 8.6: Illustration of the triplet learning objective. The anchor graph illustrated as a star should be close to its positive pair, in blue, and farther than a margin μ to its negative counterpart.

Algorithm 8.1 Training algorithm for our proposed model.

Input: Input data $\mathcal{G}$; max training iterations T
Output: Networks parameters $\Theta = \{\Theta_\phi, \theta\}$.

1: **repeat**
2: Sample triplet mini-batches $\{g_a, g_+, g_-\}_{i=1}^{N_B}$
3: $\mathcal{L} \leftarrow$ Eq. 8.5
4: $\Theta \leftarrow \Theta - \Gamma(\nabla_\Theta \mathcal{L})$
5: **until** Convergence or max training iterations T

class. Then, the triplet loss is defined as

$$\mathcal{L}(\delta_+, \delta_-) = \max(0, \mu + \delta_+ - \delta_-), \tag{8.4}$$

where μ is a fixed margin parameter, $\delta_+ = d_\theta(\phi(g_a), \phi(g_+))$ is the distance with respect to the positive sample and $\delta_- = d_\theta(\phi(g_a), \phi(g_-))$ is the distance with respect to the negative sample. Figure 8.6 illustrates how this loss performs. Note that positive pairs are pushed to be close each other whereas negative samples are separated at least by the predefined margin μ.

Moreover, following the idea introduced in [18], we apply an in-triplet hard negative mining which means that the anchor and positive samples can be swapped in case the positive sample is harder than the anchor one. Hence, we define $\delta'_- = d_\theta(\phi(a), \phi(n))$ and $\delta_* = \min(\delta_-, \delta'_-)$. Finally, the new loss with the anchor swap is defined as

$$\mathcal{L}(\delta_+, \delta_*) = \max(0, \mu + \delta_+ - \delta_*). \tag{8.5}$$

Algorithm 8.1 presents our training strategy, where $\Gamma(\cdot)$ denotes the optimizer function.

8.5 Experimental Validation

For validating the proposed approach, a keyword spotting application in historical manuscripts has been considered as our main application scenario. Moreover, a final experiment on a classical mesh graph dataset is conducted. Our empirical evaluation demonstrate that the proposed approach provides competitive results when compared to the state-of-the-art. All the code is available at `github.com/priba/graph_metric.pytorch`.

8.5.1 Historical Keyword Spotting

In Document Image Analysis and recognition, *Keyword Spotting* (KWS), also known as word spotting, has emerged as an alternative to handwritten text recognition for documents in which the transcription performance is not satisfactory. Therefore, KWS is formulated as a content-based image retrieval strategy which relies upon obtaining a robust word image representation and a subsequent retrieval scheme.

Dataset Description

The HistoGraph dataset [207, 208] is a graph database for historical keyword spotting evaluation[1]. It consists of different well known manuscripts.

George Washington (GW) [79]: This database is based on handwritten letters written in English by George Washington and his associates during the American Revolutionary War in 1755[2]. It consists of 20 pages with a total of $4,894$ handwritten words. Even tough several writers were involved, it presents small variations in style and only minor signs of degradation.

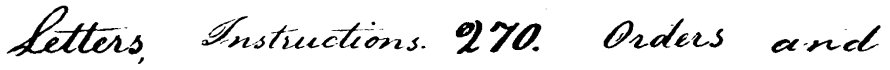
Parzival (PAR) [79]: This collection consists of 45 handwritten pages written by the German poet Wolfgang Von Eschenbach in the 13th century. The manuscript is written in Middle High German with a total of $23,478$ handwritten words. Similarly to GW, the variations caused by the writing style are low, however, there are remarkable variations caused by degradation.

Alvermann Konzilsprotokolle (AK) [169]: It consists of German handwritten minutes of formal meetings held by the central administration of the University of Greifswald in the period of 1794 to 1797. In total $18,000$ pages were used with small variations in style and only minor signs of degradation.

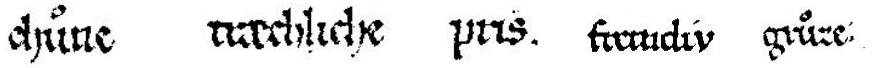
Botany (BOT) [169]: It consists of more than 100 different botanical records

[1] Available at `http://www.histograph.ch/`

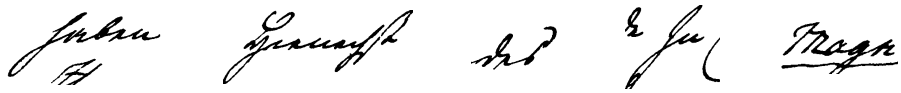
[2] George Washington Papers at the Library of Congress from 1741-1799, Series 2, Letterbook 1, pages 270-279 and 300-309, `https://www.loc.gov/collections/george-washington-papers/about-this-collection/`



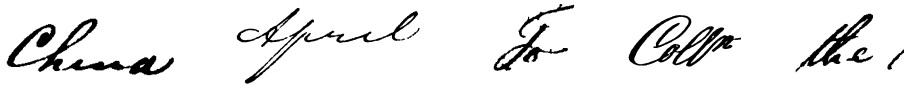
(a) George Washington (GW)



(b) Parzival (PAR)



(c) Alvermann Konzilsprotokolle (AK)



(d) Botany (BOT)

Figure 8.7: Pre-processed word examples of the four datasets.

made by the government in British India during the period of 1800 to 1850. The records are written in English and contain certain signs of degradation and especially fading. The variations in the writing style are noticeable especially with respect to scaling and intra-word variations.

Figure 8.7 provides some examples of pre-processed word images from which the graphs are created. Observe that the word segmentation of AK and BOT datasets is imperfect [169]. Moreover, these two datasets do not provide a validation set. So, some images from the training set have been used for validation. Table 8.1 provides an overview of the dataset in terms of number of words.

Table 8.1: Dataset overview in terms of number of keywords and word images for training, validating and testing respectively

Dataset	Keywords	Train	Validation	Test
GW	105	2,447	1,224	1,224
PAR	1,217	11,468	4,621	6,869
BOT	150	1,684	-	3,380
AK	200	1,849	-	3,734

To obtain a graph for each word in these datasets, the two most promising graph constructions introduced in [208] have been used:

- **Keypoint**: Characteristic points are extracted from the skeletonized word

image. Moreover between the connected characteristic points, equidistant nodes are inserted on top of the word skeleton.

- **Projection**: An adaptive grid is generated according to the vertical and horizontal projection profiles. Then, nodes are inserted in the corresponding center of mass of each grid cell. Moreover, undirected edges are inserted if nodes are directly connected by a stroke.

All the datasets presented in this work only contain spatial and structural information. This means that nodes are labeled with its normalized (x,y)-position on the image and that edges are unlabeled.

Experimental Protocol

The experiments reported in this section use $K = 3$ GNN layers and, for the edit cost operation described in Equation 8.3, we experimentally set the parameters τ_i and τ_d to 0.5. Following the evaluation schemes of previous word spotting methodologies, we performed our experiments on two evaluation protocols.

- **Individual**: Each query image follows the traditional retrieval pipeline. Thus, queries are matched against the elements in the gallery and each ranking is evaluated independently.
- **Combined**: A query consists of a set of graphs $Q = \{q_1, \ldots, q_t\}$ where all the instances $q \in Q$ represent the same keyword. In this case, we consider the minimal distance achieved on all t query graphs. This second evaluation protocol was adopted in some previous graph-based word spotting works [8, 208]. The motivation of this setting is to mitigate the structural bias provided by the query instance, *i.e.* different handwriting styles provide extremely different graphs, so, in terms of graph distances, it is unrealistic to consider them from the same class.

For the evaluation, we use the *mean Average Precision* (mAP), a classic information retrieval metric [190]. First, let us define Average Precision (AP) as

$$\mathrm{AP} = \frac{\sum_{n=1}^{|\,\mathrm{ret}\,|} P@n \times r(n)}{|\,\mathrm{rel}\,|}, \tag{8.6}$$

where $P@n$ is the precision at n and $r(n)$ is a binary function on the relevance of the n-th item in the returned ranked list. Then, the mAP is defined as:

$$\mathrm{mAP} = \frac{\sum_{q=1}^{Q} \mathrm{AP}(q)}{Q}, \tag{8.7}$$

where Q is the number of queries.

Ablation Study

We first empirically investigate the influence of the margin parameter μ to each model, as well as the importance of the GNN layers choice. Table 8.2 presents a comparison of the different settings, providing the averaged mAP of 4 runs and its corresponding standard deviation. This evaluation has been done for all the datasets and for both graph representations *viz.* Keypoint and Projection. The evaluation protocol in this experiment is Individual as we believe that it is the natural experimental setting for this problems.

From these results, we observe that GRU-based models are slightly better, allowing a higher degree of deformations between words from the same class. Note that both datasets AK and BOT do contain imperfect word segmentations. In addition, these datasets contain samples written with a more artistic calligraphic style as shown in Figure 8.7. Thus, the artistic strokes are drivers of a higher degree of complexity. Observe that the performance drop in BOT dataset is also explained by the artistic nature of the dataset.

Additionally, the Keypoint representation performs the best on the GW dataset, since the strokes are simpler and its structure in terms of the binary image skeleton is more relevant for a proper retrieval. Finally, we observe that a higher margin μ is more adequate for the GAT-based models whereas it's harmful for the GRU-based one.

Table 8.2: Study on the GNN model and margin parameter of the proposed model. Mean average precision (mAP) and standard deviation (average on four runs) for graph-based KWS system on George Washington (GW), Parzival (PAR) Alvermann Konzilsprotokolle (AK) and Botany (BOT) datasets.

	Model	μ	GW		PAR		AK		BOT	
			mAP	±	mAP	±	mAP	±	mAP	±
Keypoint	GAT	1	72.49	1.169	66.46	3.162	62.90	1.325	39.86	0.396
		10	**76.92**	2.309	**73.14**	0.973	62.72	1.783	**41.52**	0.782
	GRU	1	72.86	3.331	67.27	1.281	**64.42**	1.003	39.69	0.532
		10	68.45	2.715	48.59	9.571	60.84	1.127	38.22	0.778
Projection	GAT	1	67.86	2.379	70.77	1.906	63.44	1.233	39.12	2.037
		10	**70.25**	3.431	**75.19**	0.755	62.72	1.518	38.83	2.801
	GRU	1	68.09	1.234	71.07	1.933	**65.04**	1.226	**42.83**	0.568
		10	63.39	4.222	52.32	1.298	60.51	1.451	37.59	0.778

Results and Discussion

Table 8.3 compares with graph-based methodologies. In this setting we follow the Combined evaluation protocol reported by [8, 208]. For each dataset and graph representation, we use the best model reported in the previous section. Observe that, for a fair comparison, that is, using the same graph representation, we outperform both AED [181] and HED [82] on all the datasets but AK, where we obtain very similar results. However, the ensemble methods reported in [208] are able to obtain a better performance on the datasets with more variability. Note that these ensembles combine, in different ways, the graph distances computed on different graph representations of the same images. Therefore, we do not consider it a fair comparison but a remarkable fact that the performance of our system is able to outperform them in two of the datasets while obtaining competitive results on the other two.

Table 8.3: Comparison against state-of-the-art on graph-based KWS techniques. Mean average precision (mAP) for graph-based KWS system on GW, PAR, AK and BOT datasets.

Distance	Representation	GW	PAR	AK	BOT
AED [181]	Keypoint [8]	68.42	55.03	77.24	50.94
	Projection [8]	60.83	63.35	76.02	50.49
Ensemble [208]	min	70.56	67.90	82.75	65.19
	max	62.58	67.57	82.09	67.57
	mean	69.16	79.38	84.25	**68.88**
	sum_α	68.44	74.51	**84.77**	68.77
	sum_{map}	70.20	76.80	84.25	**68.88**
HED [82]	Keypoint [8]	69.28	69.23	79.72	51.74
	Projection [8]	66.71	72.82	**81.06**	51.69
Our	Keypoint	**78.48**	79.29	78.64	51.90
	Projection	73.03	**79.95**	79.55	**52.83**

Table 8.4 shows a comparison with non-graph based approaches. In particular, we compare against three state-of-the-art learning-based reference systems of the ICFHR2016 competition [169]. In this case, the evaluation of these learning-based frameworks follows the protocol described in the competition. Thus, queries of the same query keyword are considered to be independent. Note that, due to this query protocol, the learning-based frameworks are not directly comparable to the state-of-the-art graph-based KWS results that we have reported above. In this table, we observe the superiority of learning methods working directly on the image domain. In particular, PHOCNet [209] leads to stunning accuracies for this task. Their method estimates the *Pyramidal Histogram of Characters* (PHOC) representation of word images. However, the proposed graph-based approach, is able to provide a

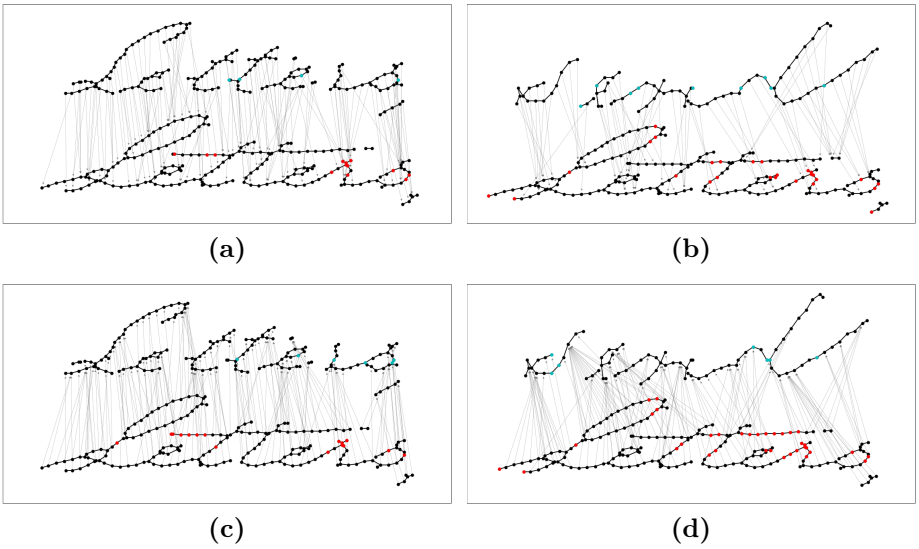


Figure 8.8: Visualization of the learned node correspondance. First row, shows the node matching from top to bottom; the second row of the figure shows de opposite. (a)–(d) "Letters"–"Letters"; (b)-(d) "send"–"Letters"

new step towards closing the gap between structural and statistical methodologies on this kind of tasks. In addition, the proposed approach is able to deal with the noise introduced by the graph construction.

Table 8.4: Comparison against non-graph learning based systems. Mean average precision (mAP) for graph-based KWS system on AK and BOT datasets.

Method	Representation	AK	BOT
CVCDAG [6]	-	77.91	75.77
PHOCNet [209]	-	96.05	89.69
QTOB [231]	-	82.15	54.95
Our	Keypoint	64.42	41.52
	Projection	65.04	42.83

Finally, Figure 8.8 provides qualitative examples of our matching framework for a positive and negative sample. The first row provides the top to bottom matching whereas the second row shows the opposite, from the bottom to the top. Notice that in the positive example, both directions are much more consistent than in the negative case.

8.5.2 Experimental Comparison to GMN

Among the graph metric learning approaches in the literature, the *Graph Matching Networks* (GMN) work [138] is the most prominent one. In this section, we propose an extra experiment to compare with their work.

Dataset Description

The IAM Graph Database Repository [180] provides several graph datasets covering a wide spectrum of different applications. In particular, we focused on the COIL-DEL dataset. The COIL-100 [160] consists of 100 object images at different poses. In order to construct the COIL-DEL dataset, these images were converted into mesh graphs by means of the Harris corner detection algorithm followed by a Delaunay triangulation. COIL-DEL is divided in 2,400, 500 and 1,000 graphs for training, validation and test respectively. In average these graphs have 21.5 nodes and 54.2 edges. Thus, they are rather small graphs.

Experimental Protocol

In these experiments, we followed the same experimental protocol introduced by Li *et al.* [138], so we evaluated our method on two different metrics:

- **pair AUC**: The area under the ROC curve for classifying pairs of graphs as similar or not on a fixed set of 1,000 pairs.
- **triplet accuracy**: The accuracy of correctly assigning a higher similarity to the positive pair in a triplet than a negative pair on a fixed set of 1,000 triplets.

Note that the fixed pairs and triplets are not the same from the original paper. We performed a random selection of these pairs and triplets while trying to balance the number of examples per class.

Results and Discussion

Table 8.5 presents a comparison to the state-of-the-art techniques on graph metric learning. In particular, we compare with the different architectures proposed in [138].

It is not surprising that their GMN technique outperforms our proposed model as they do incorporate cross-graph connections following an attention paradigm. Therefore, the correspondence is learned end-to-end in a much robust way. However, this is only feasible in rather small datasets as it incorporates a huge computational overhead. Notice also, that their GNN and Siamese-GNN models just

obtain a slightly better performance than our proposed approach. However, when dealing with such small graphs it is hard to compare against embedding based approaches as they are able to encode the graph characteristic features without a huge loss of information.

In this extra experiment, we have also evaluated the effect on the choice of GNN models, the number of layers and the margin parameter μ. From this table, we conclude that the GRU-based models are drivers of a better performance on these experiments. Moreover, we find important to set our margin parameter to 1. The number of layers have not proven to bring a boost on performance on this particular dataset. Overall, our best model is able to obtain comparable results against GMN in this small dataset.

Table 8.5: Performance comparison on the COIL-DEL dataset against the methodologies introduced in [138]. We studied the effect of the proposed GAT and GRU models, as well as, the number of layers and margin parameter μ.

Model		# Layers (K)	μ	Pair AUC	Triplet Acc
GCN [138]		-	-	94.80	94.95
Siamese-GCN [138]		-	-	95.90	96.10
GNN [138]		-	-	98.58	98.70
Siamese-GNN [138]		-	-	98.76	98.55
GMN [138]		-	-	98.97	98.80
Our	GAT	3	1	97.82	96.74
			10	96.22	96.94
		5	1	97.85	96.94
			10	97.70	97.54
	GRU	3	1	98.08	97.50
			10	96.25	95.69
		5	1	97.56	97.60
			10	95.36	96.07

8.6 Conclusions

In this chapter, we have proposed a triplet GNN architecture for learning graph distances. Our architecture is able to learn node embeddings based on structural information of nodes local contexts. These learned features lead to an enriched graph representation which is later used in the distance computation. Moreover, we extended the graph edit distance approximation *viz.* Hausdorff edit distance, to the new learning framework in order to learn its operation costs within an end-to-end fashion. We have validated our proposed architecture on a graph retrieval

scenario, in particular, we faced a keyword spotting task for handwritten words. Finally, we demonstrated competitive results against state-of-the-art, learning-based methods for graph distance learning.

Several future research lines emerge taking as starting point the proposed framework. Firstly, the learned distance does not represent the exact graph edit distance but a metric between graphs that achieves the desired ranking properties, *i.e.* it distinguishes graphs belonging to the same class. In that sense, the obtained distance depends on our graph classification rather than being an unsupervised approach such as the traditional definition of graph edit distance. Hence, future research could focus on obtaining the real graph edit distance or at least, obtain a better approximation making use of lower and upper bounds. Secondly, the proposed framework do not exploit the edge structure at matching time as we considered it implicitly encoded as node features. However, leveraging the edges information at matching time might lead to better matching results. Another promising line of research relates to the use of different graph pooling layers for reducing the size of large graphs before computing the learned Hausdorff edit distance.

9 | Table Detection in Invoice Documents

Bloody paperwork [...]
You can't move without a form.

– Harry Tuttle, Brazil

Tabular structures in documents offer a complementary dimension to the raw textual data, representing logical or quantitative relationships among pieces of information. In digital mail room applications, where a large amount of administrative documents must be processed with reasonable accuracy, the detection and interpretation of tables is crucial. Table recognition has gained interest in document image analysis, in particular in unconstrained formats such as absence of rule lines and unknown information of rows and columns. With the recent research advances in information extraction, automatic processing of administrative documents has gained popularity. However, these documents usually contain sensitive contents limiting the amount of public benchmarking datasets. In this chapter, we propose a graph-based approach for detecting tables in document images which do not require the raw content of the document. Therefore, the sensitive content is previously anonymized and, instead of using the raw image or textual content, we make use of a purely structural approach to keep sensitive data anonymous. Our framework makes use of graph neural networks in order to describe the local repetitive structures that constitute a table. In particular, our main application domain are invoice documents. We have carefully validated our approach in two invoice datasets and a modern document benchmark. Our experiments demonstrate that table structures can be detected by purely structural approaches. Additionally, due to the scarcity of benchmark datasets for this task, we have contributed to the community a novel dataset derived from the RVL-CDIP invoice data.

9.1 Introduction

This chapter reports on the research developed in the scope of an industrial collaboration with *omni:us*[1], a German-based company leader on the automatic processing of administrative documents, and in particular invoices. Thus, in this chapter, we present a research that has been motivated by the real needs of the industry.

[1] `https://omnius.com/`

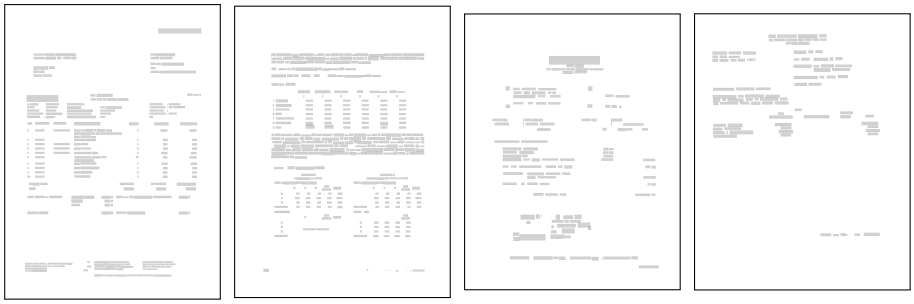
Moreover, all the knowledge generated in this chapter has been transferred to the before-mentioned company.

Extracting information from administrative documents in digital mailroom applications is a common task in various domains including finance, insurance, manufacturing, and trading. The manual extraction of the relevant information is often a tedious and time consuming process. The ultimate goal of automatic information extraction methods is to reduce the manual effort and speed up the overall process. For forms and structured documents with simple named entities such as names, dates, and prices, the existing extraction methods can already achieve a high accuracy. However, for semi-structured or unstructured documents with challenging contents like addresses, tables or some specific details, the automatic extraction is not yet good enough and still requires human assistance and validation.

Tabular layouts have been, over centuries, one of the main instruments to communicate ideas through documents. We refer to tabular layouts in the broadest sense, *i.e.* information terms organized in a two-dimensional arrangement with some perceptual organization rules, in terms of horizontal and vertical alignments. Such organization of information offers a complementary dimension to the plain document contents, showing logical or quantitative relationships among pieces of information. Tables are graphical structures that visually show relationships among named entities, giving a rich semantic message beyond the basic literality of the constituent terms. Table structured layouts are present in documents of different types and time periods. Parishes have been registering over centuries birth, marriage and death events in manuscripts organized in tabular arrangements. In 1869 the Russian chemist Dmitri Mendeleev presented in a manuscript a system to arrange the chemical elements. The periodic table has become an icon of science and culture[2]. The public administration, due to socio-political models based on the rule of law introduced in the 19th century, had to develop mechanisms for collecting and quantifying the composition of the population and sources of wealth. Civil and notarial documents like census, tax, election records among many others, all of them mainly in tabular form, proliferated as information collection documents. In the specific case of administrative documents, they are very often semi-structured, in other words, without a fixed layout but sharing a common set of components being header, footer, sender and recipient some typical examples. This spatial arrangement can be roughly perceived as a tabular layout. A key observation is that humans, when reading, perceive tables because of the observation of repetitive patterns. The Gestalt principles of visual perception [146] can be applied, table items have a regular arrangement, with a continuity in horizontal and vertical directions.

Moreover, administrative documents, in particular, invoices, usually contain sensitive information (*e.g.* names, addresses, bank details, health information) that should be protected. Therefore, privacy policies may impose limitations with re-

[2]The United Nations has declared 2019 as the International Year of the Periodic Table, `https://iypt2019.org/`.



(a) **(b)** **(c)** **(d)**

Figure 9.1: Example of several anonymized administrative documents. (a) Invoice taken from the CON-ANONYM dataset; (b) modern document belonging to the cTDaR table detection competition; (c-d) invoices obtained from the RVL-CDIP dataset.

gards to the document handling that may prevent the document to be sent to a cloud service. For instance, some works have studied how to share models keeping the training data private [247]. Taking this into consideration, we constrained our problem not to rely on the document contents, but pre-processed anonymized data. Figure 9.1 shows some examples of documents containing one or more tables without any visual cues. Note that this is the **only** information which is provided to our system. Later in this dissertation, we will analyze these documents in more detail. Before further reading, we suggest the reader to try to spot the table regions on these documents as we will later see which are the limitations of our approach when restricted to this setting.

Recently, table analysis and recognition has received a lot of attention in the DIAR community. For instance, in the last *International Conference on Document Analysis and Recognition* (ICDAR)[3] held in Sydney, which is the flagship conference of this community, a competition was proposed related to this topic[4] [87]. Moreover, one oral session was exclusively devoted to table analysis besides several works presented as posters.

In this chapter, we propose a graph message passing approach for the detection and interpretation of tabular structures. Due to their representational power, graphs are suitable models for tabular layouts. On the one hand, graph nodes represent segmented entities, *i.e.* isolated words, groups of words or symbols. On the other hand, graph edges are inferred in terms of visibility relations. Figure 9.2 illustrates a possible representation scheme of a document in terms of a visibility graph. When a table is present in the document, due to the repetitivity principle, its corresponding graph can be decomposed into a set of repeated and connected

[3] `http://icdar2019.org/`
[4] Available at `http://sac.founderit.com/index.html`

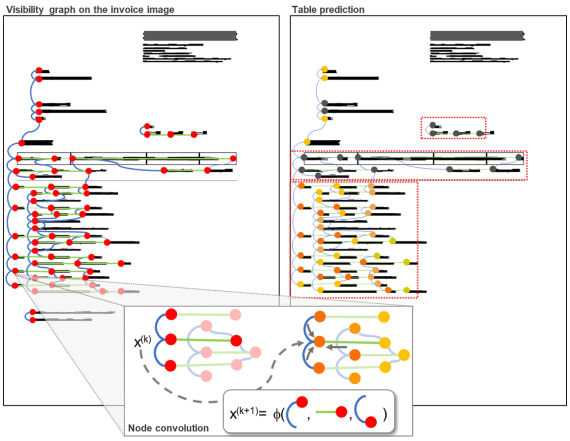


Figure 9.2: Graph representation of an invoice (left). Graph convolution idea (middle). Similar embeddings in table nodes after convolutions (right).

graphlets, *i.e.* small induced subgraphs. Thus, the detection of tables is formulated in terms of a frequent graphlet discovery algorithm. Since these local structures have a high variability, we propose to make use of a supervised learning framework. Therefore, the proposed system is able to train with a priori knowledge about the graphlets that correspond to table parts. In particular, graph neural networks offer a solid foundation to achieve this objective. These models are able to learn a graph node embedding which encodes the context of the node as it is illustrated in Figure 9.2. A GNN layer embeds into a node a combination of the information of the neighboring nodes and edges. Then, after each layer, structurally similar nodes tend to receive a "similar" encoding. In addition, our method exploits the belief propagation algorithms to add consensus at the GNN predictions for nodes and edges. Finally, we define the table detection task as a node clustering problem.

In summary, this chapter has several contributions. Firstly, we introduce a GNN model for tabular layout detection in administrative documents based on the classification of graph nodes and edges which is not constrained to a rigid tabular layout in terms of single rows, columns or presence of rule lines. Secondly, belief propagation has been proposed as a post-process to the GNN prediction in order to enforce consensus between node and edge prediction. Moreover, the proposed model is language independent, *i.e.* although an OCR is used as pre-processing, only character type attributes are considered, but not the transcription of the OCR. This has the side advantage that privacy is preserved if required. As far as we know, this is the first approach not considering the content information for table detection.

Even though processing administrative documents is of key importance for the industry, publicly available data is scarce due to data privacy issues. Therefore, a

dataset[5] consisting of 518 invoice pages from RVL-CDIP [104] dataset augmented with ground truth for table detection and layout analysis has been created and publicly released. Note that this dataset is provided for the use case we are dealing in this work, therefore, we provide the bounding box information of the OCR entities.

The rest of the chapter is organized as follows. Section 9.2 reviews the state-of-the-art in table detection and recognition. Section 9.3 describes the system architecture. Afterwards, Section 9.4 evaluates our approach on several datasets. Finally, section 9.5 draws conclusions and outlines future research directions.

9.2 Related Work on Table Detection and Recognition

Table detection is the first step towards a table analysis and recognition methodology. We can classify these approaches according to the type of the input documents, *i.e.* spreadsheets or textual documents. Here we focus on the latter which is more challenging due to the lack of explicit row and column information in most of the cases.

Table detection and recognition in unconstrained documents is considered a challenging task and has recently received significant attention within the community [47, 94, 118, 124, 144, 172, 197]. Available OCR systems only provide textual information without considering the actual tabular structures that exist in a document. However, recognizing tabular structures is crucial for getting a contextual meaning of the recognized textual information, which acted as the main motivation behind this research line. Early works on this topic were mostly bottom-up in nature [40, 119, 120, 226], and they often start by detecting words or parallel lines following some heuristic to group homogeneous elements to detect tabular components (*i.e.* rows and columns), and hence, tables in a document. As a consequence, most of the methods do not work well on multi-column document images due to the simplifying assumptions [198]. Later, Ghanmi and Belaïd [92] proposed to use a *Conditional Random Field* (CRF) to localize tabular components in unconstrained handwritten documents.

The recent developments on table detection are focused on the current advances of deep learning techniques. Among them, Gilani *et al.* [94] proposed a variant of region proposal network where they feed pre-processed document images for detecting tables. A similar approach based on a region proposal network is also proposed in DeepDeSRT [197] for detecting tables, they further extended it to rows and column detection. In [172], Rashid *et al.* used a pre-trained neural network model to distinguish whether a word belongs to a table or not, and depending on the outcome applied some post-processing techniques for detecting tables. A

[5]Available at `https://zenodo.org/record/3257319`

saliency based fully connected neural network performing multi-scale reasoning on visual cues followed by a fully connected CRF for localizing tables and charts is proposed by Kavasdis *et al.* [118]. In [47], Clinchant *et al.* developed two graph-based methods and compared them for the table detection task, where the first method relies on graph Conditional Random Fields (gCRF) [131], while the second method is based on Graph Convolutional Networks (GCN) [123]. At the same time, Koci *et al.* [124] proposed another graph-based method, where instead of a GCN, they used a remove and conquer algorithm for detecting tables. Later, Lohani *et al.* [144] used a similar GCN-based technique for recognizing different fields in an invoice.

Very recently, Paliwal *et al.* [165] proposed a CNN architecture for learning features able to perform table detection and column prediction simultaneously. Also, some very recent works have focused on the decomposition of the table structure in its grid patterns and cells [171, 213].

9.3 Table Detection Framework

Tables are complex document entities composed of different elements (headers, rows, columns, etc.). These elements are distributed on document pages following repetitive structures. When dealing with structured data, we propose to use the high representation power of graphs to discover these repetitive patterns characterizing the tabular structure. However, this repetitive structures are not consistent among different documents and require some a priori knowledge to be able to identify them properly as tables. With the aim of obtaining these knowledge from examples, data driven techniques such as deep learning provide an efficient framework to learn the key substructures to deal with these complex documents.

Considering a document graph whose nodes are entities such as words or symbols and the edges represent their spatial relationships, in this work, we propose a GNN architecture, which is trained in a supervised manner. Therefore, we know which is the corresponding class of each node, in terms of its semantic region (header, address, table, etc.), as well as if an edge is connecting two entities of the same region, *i.e.* the two connected nodes belong to the same semantic region. Thus, we formulate the problem of table detection as a classification problem by learning how entities are related. In this setting, cross entropy is the proposed objective function which is optimized by the Adam optimizer [122] with weight decay for parameter estimation. Once both nodes and edges have been classified, a grouping algorithm is applied. With this aim, we implemented the *belief propagation* algorithm [241] as a non-learnable approach which combines nodes and edges predictions for the sake of avoiding inconsistencies from both sources of information. Our final goal is to decide which edges connect different semantic regions and, therefore, can be removed from the graph in order to isolate table regions (connected subgraphs). Hence, once the corresponding edges have been removed, the

table detection problem, is reformulated in terms of finding connected components whose nodes are classified, in average, as tables. Figure 9.3 shows the outline of the whole table detection method.

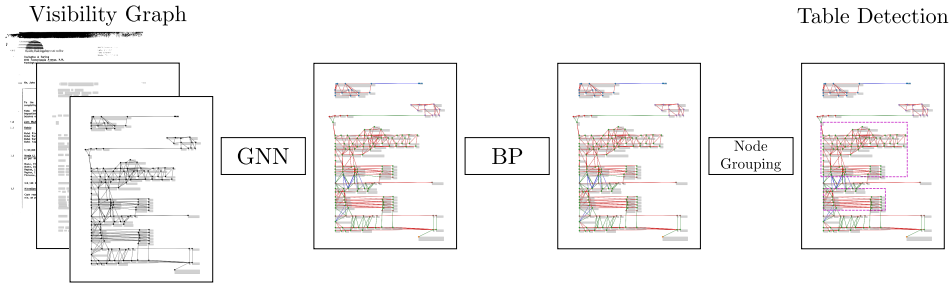
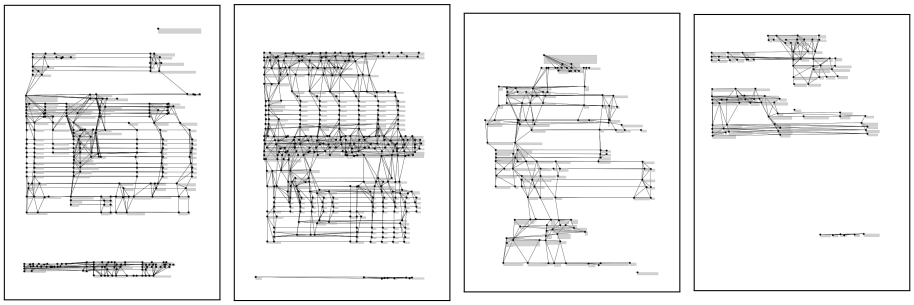


Figure 9.3: Outline of the proposed table detection method. (i) the visibility graph is constructed from the original invoice image. Afterwards, the successive modules are applied to the obtained graph: (ii) a GNN-based node and edge classifier; (iii) the belief propagation (BP) algorithm to add consensus between the corresponding node and edge predictions and; (iv) a final node grouping to determine the bounding box of the predicted table.

9.3.1 Graph-based Representation of Invoice Documents

In this subsection, we describe how we obtain a graph-based representation given an administrative document image. It is during this stage that the document image containing possible sensitive contents is anonymized. Given an administrative document image, we firstly segment the physical layout detecting graphical and textual entities. This is done by applying an off-the-shelf OCR. On the one hand, bounding boxes of segmented textual and graphical entities are obtained, on the other hand, textual attributes (numeric, alphabet or symbol) are associated to the entities. At this point, we must remark that even though the OCR provides the recognized text, this is not used in our setting in order to guarantee the anonymity of the data from this point on. Therefore, from an industrial perspective, the document is treated in terms of its content in the client internal network. Afterwards, the table detection service, allocated in the cloud, will not require that sensitive information to perform the detection.

Thus, given a document, we represent each detected entity (words, symbols or numbers) with the available information whereas keeping the anonymity of the document. In this case, each entity corresponds to a 7-dimensional vector containing the bounding box position in terms of (x,y)-coordinates, width and height, and a histogram which counts the number of numeric, alphabet or symbol elements. This encoded information is the one that will be used for the table detection, and the only one required for our system. Then, from this set of entities, we generate



(a) **(b)** **(c)** **(d)**

Figure 9.4: Example of several graph representations for administrative documents. (a) Invoice taken from the CON-ANONYM dataset; (b) modern document belonging to the cTDaR table detection competition; (c-d) invoices obtained from the RVL-CDIP dataset.

a visibility graph, sometimes referred in the literature as line-of-sight graph [148], in order to represent the structural information of the document.

Let $g = (V, E, \mu, \nu)$ be such a *visibility graph*. The set of nodes V corresponds to the detected entities of the document. The set of edges E represents visibility relations between nodes. Two entities are connected with an edge if and only if the bounding boxes are vertically or horizontally visible, *i.e.* a straight horizontal or vertical line can be traced between the bounding box of two entities without crossing any other. It is enough to take these two directions to check the visibility since it follows the way which tables are usually organized in documents. Finally, long edges covering more than a quarter of the page height are discarded. μ and ν are the labeling functions for nodes and edges respectively. As explained, node labels are a 7-dimensional vector and the edge labels are the length and angle between the center of the bounding boxes. The angle has been encoded according to its sine and cosine. Thus, the angles makes our visibility graph directed. Figure 9.4 draws the visibility graphs of the documents introduced in Figure 9.1.

9.3.2 The GNN Architecture

Given the generated graph, we propose to make use of a graph neural network architecture $\varphi(\cdot)$ as learning paradigm. We propose to solve the table detection problem as a graph clustering framework where the network should be able to find the group of nodes that compound a table. Figure 9.5 illustrates the training setting of the proposed GNN architecture, once the document has been processed and the graph has been constructed.

In our setting, both the embedding layers as well as the final multi-layer perceptron (MLP) have been set to single fully-connected layers.

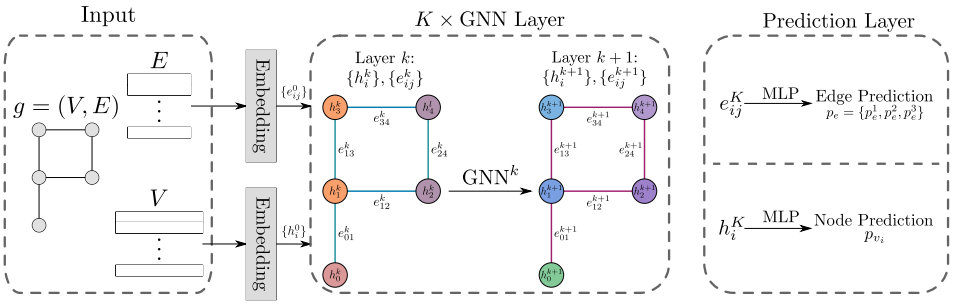


Figure 9.5: Overview of the proposed GNN architecture for table detection. **Input**: visibility graph of document entities. **Output**: each entity classified as well as its pairwise relationship with other entities defining whether they belong or not to the same semantic region. Our framework is composed of an embedding layer for both nodes and edge features, K graph neural network layers and 2 classifiers for nodes and edges respectively.

Similarly to the previous chapter, we have evaluated two different GNN layers as the core of the proposed architecture.

GAT-based layer: This layer uses the Graph Attention Networks (GAT) [217] as they provide an anisotropic update scheme by means of its attention weights. A detailed explanation of this layer can be found in Section 7.3.

GRU-based layer: It relies on the scheme proposed by Gilmer *et al.* [95] as an improvement of the Gated Graph Neural Networks (GG-NN) introduced by Li *et al.* [139]. GRU-based update functions, have been mentioned in some works to report better results [138]. The main difference regarding the model introduced in the previous chapter is that now, we make use of edge update layers in order to learn the pairwise information between the entities across our node updates. These layers take advantage of the edge hidden state and the corresponding nodes information. The proposed edge update layer is formally defined as

$$e_{vw}^{(k+1)} = \text{MLP}([e_{vw}^{(k)}, h_v^{(k)}, h_w^{(k)}]), \tag{9.1}$$

where $[\cdot, \cdot]$ stands for the concatenation operation and $h_v^{(k)}$ and $h_w^{(k)}$ are the hidden state of the source and destination nodes respectively. Observe that this definition depends on the edge direction. In our case, initially we already have a bidirected graph as we consider the angle between entities as a relative positional encoding. Thus, we propose to update the edges hidden state at each message passing layer. This is motivated by the fact that we use the edge information explicitly in the update function.

In both GNN architectures, the edge layer defined in Equation 9.1 is used to compute the final edge features $e_{vw}^{(K)}$ considering $h_v^{(K)}$ and $h_w^{(K)}$. Therefore, in the GAT-based model, the structural information is encoded at node level.

We denote the final node and edge prediction as p_v and p_e respectively. For the edges these predictions are defined as $p_e = \{p_e^0, p_e^1, p_e^2\}$ where p_e^0 is the probability that the edge is contained inside a semantic region; p_e^1 identifies edges relating two different semantic regions and classes; and p_e^2 identifies edges connecting two different semantic regions but representing the same class, for instance, connecting two distinct tables. Note that $p_e^0 + p_e^2$ is the probability of an edge connecting different semantic regions.

9.3.3 Learning Objectives

We propose to combine two complementary learning objectives according to both outputs depicted at the prediction layer in Figure 9.5. On the one hand, we use a node classification loss $\mathcal{L}_{\text{nodes}}$ which is able to exploit the local graph regularities to identify to which semantic region belongs a node. On the other hand, we use the pairwise information $\mathcal{L}_{\text{edges}}$, where edges are classified as connecting nodes belonging to the same semantic region or not. To help in the learning process, we have additionally divided the second case in order to distinguish two nodes, which are part of different region but belong to the same class. This division gives more importance to properly classify these edges that otherwise, are drivers of segmentation errors in the following steps.

Observe that, in this setting, nodes are trained as a multi-class classification problem. Therefore, different semantic regions besides the table category are considered. However, with the current information, which intentionally neglects the textual content for privacy regulations of the service provider, we aim at providing an extra contextual information for table detection rather than properly detecting these unstructured regions. Thus, our model benefits of learning some region-wise relations. For instance, a table usually appears below the address and above the footer for the page.

We propose a cross entropy loss for both, nodes and edges. Therefore, our objective function for nodes is described as

$$\mathcal{L}_{\text{nodes}}(x, \text{class}) = -\log\left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}\right), \tag{9.2}$$

where x are the the scores for each class. Similarly, the loss for the edges $\mathcal{L}_{\text{edges}}$ is defined by the same equation.

Additionally, we have weighted the loss function to make each class to contribute equally to the final loss. Notice that the weighting factor is required to balance the provided information, specially for the graph edges where the problem becomes extremely unbalanced. Thus, the final loss is defined as

$$\mathcal{L} = \mathcal{L}_{\text{nodes}} + \mathcal{L}_{\text{edges}}. \tag{9.3}$$

9.3.4 Table Detection

Taking advantage of the belief propagation (BP) algorithm, we propagate the nodes prediction in order to marginalize the probability of cutting a particular edge. Therefore, the edge and node update stages of the belief propagation are considered to add consensus between node and edge classification. Let us consider the set states defined by $Y = \{0, 1\}$, where 0 and 1 define whether an edge should be cut or not. Therefore, considering a particular edge $e = (v_{\text{src}}, v_{\text{dst}})$ and its source v_{src} and destination v_{dst} nodes we predict the cut probability $p_y = [P(Y = 0|e, v_{\text{src}}, v_{\text{dst}}), P(Y = 1|e, v_{\text{src}}, v_{\text{dst}})]^t$. Remember that edges are classified between three different classes, $p_e = \{p_e^0, p_e^1, p_e^2\}$ In this setting, we define the BP factor as $\phi = [\phi_e^0, \phi_e^1] \in M^{N \times N \times 2}$ where, N is the number of classes at each dataset, and

$$\phi_e^0 = \begin{pmatrix} p_e^0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_e^0 \end{pmatrix}, \quad \phi_e^1 = \begin{pmatrix} p_e^2 & \cdots & p_e^1 \\ \vdots & \ddots & \vdots \\ p_e^1 & \cdots & p_e^2 \end{pmatrix}. \tag{9.4}$$

Thus, at each iteration of the belief propagation algorithm updates the edge states probabilities p_y according to

$$\begin{aligned} P(Y = 0|e, v_{\text{src}}, v_{\text{dst}}) &= p_{v_{\text{src}}}^t \phi_e^0 p_{v_{\text{dst}}}, \\ P(Y = 1|e, v_{\text{src}}, v_{\text{dst}}) &= p_{v_{\text{src}}}^t \phi_e^1 p_{v_{\text{dst}}}. \end{aligned} \tag{9.5}$$

Similarly, the node predictions are also updated according to the edge state probability p_y. In this case, each neighboring node of v_{dst} sends a message formally defined as

$$m_{v_{\text{src}} \to v_{\text{dst}}} = p_{v_{\text{src}}}^t \phi_e^t p_y, \tag{9.6}$$

which, following the belief propagation algorithm, is later aggregated as the product of messages followed by a normalization. Note that the factor ϕ is transposed to meet the size of the edge state probabilities p_y. Finally, given these probabilities, we are able to cut these connections that connect entities belonging to different semantic regions. Algorithm 9.1 provides an overview of the belief propagation algorithm.

Once the edges have been removed, each independent connected component is studied. For all of them, its node prediction are averaged in order to obtain the predicted class of the semantic region.

Algorithm 9.1 Belief propagation algorithm to add consensus between node and edge predictions.

Input: Input data $g \in \mathcal{G}$; trained GNN φ, belief propagation iterations T.
Output: Edge state probabilities $p_y = \{p_y(e)\}_{e \in E}$.

$g' = (V', E') \leftarrow \varphi(g)$, where $V' = \{p_v\}_{v \in V}$ and $E' = \{p_e\}_{e \in E}$
$\Phi = \{\phi_e\}_{e \in E} \leftarrow$ Eq. 9.4
repeat
    Update probabilities p_y (Eq. 9.5)
    Normalize p_y
    Update V' according to $\prod m_{v_{\text{src}} \to v_{\text{dst}}}$ (Eq. 9.6)
    Normalize V'
until Max belief propagation iterations T

9.4 Experimental Validation

This section carefully evaluates the proposed model on three different databases containing documents from different typologies. Moreover, we present a study regarding the limitation of a completely structural approach on anonymized data.

9.4.1 Datasets

Three datasets have been used for the evaluation of the proposed table detection framework. Even though the evaluated task is the same, the typology of the documents presents an important variability. Therefore, they are useful to evaluate our method in several use cases.

CON-ANONYM: This is a particular dataset of 960 documents which has been used as part of an industrial collaboration. Therefore, it is only available for the internal use in the **omni:us** company and it cannot be made publicly available as it contains sensitive contents. The documents are annotated with the following 8 region labels *claim*, *car*, *cost*, *supplier*, *insured*, *mixed*, *other* and *table*.

RVL-CDIP: The original dataset proposed by Harley *et al.* [104] was designed for the evaluation of CNNs for the task of document image classification and retrieval. It contains 400,000 grayscale images, which are divided in 16 classes with 25,000 images per class. We have made use of a subset of these documents for the evaluation of our table detection method. We selected 518 images from the *invoice* class, which have been annotated with the following 6 regions labels *supplier*, *other*, *table*, *receiver*, *invoice_info* and *total*.

cTDaR: Table detection and recognition has become a fundamental step for any information retrieval technique working on structured and semi-structured documents. The *ICDAR 2019 Competition on Table Detection and Recognition* (cT-

DaR)[6] [87] is one of the many research events that have recently raised attention on the table analysis problem. This competition is divided into two tracks according to table detection (Track A) or table recognition (Track B) where the cell structure of the table should be provided. Moreover, they provide two datasets according to modern and archival datasets.

In this chapter, we have utilized the modern dataset as it contains data closer to the proposed use case of invoices. Moreover, due to the anonymization step, we only compare our method for the task defined for the track A as the lack of visual cues makes the recognition step extremely challenging.

For the sake of keeping anonymity, we apply the ABBYY OCR[7] on the three datasets for extracting the text and encode it into a sequence of attribute types. For example, 'NNS' would encode '24$' and 'AAAAAA' would encode 'Google', where 'A', 'N' and 'S' respectively denote if the type is alphabetical, numeric or symbol. In addition to the attribute type, we also keep the bounding boxes of each word that appears in these documents. The ABBYY OCR has been selected as it is the framework used by our industrial partner. Note that the OCR parameters have not been tuned for this specific data. Table 9.1 shows a comparison of the datasets. Note that in CON-ANONYM and RVL-CDIP there are documents containing pages without tables.

Table 9.1: Summary of the datasets statistics as well as the proposed division in train, validation and test sets.

	CON-ANONYM	RVL-CDIP	cTDaR
Total # documents	950	518	840
(tr, va, te)	(665, 95, 195)	(362, 52, 104)	(540, 60, 240)
Total # pages	1252	518	840
Total # tables	1202	485	1423
Total # classes	8	6	2
Avg. # nodes/page	245.31	123.80	402.17
Avg. # edges/page	755.43	327.42	1481.76

More details on these datasets are provided in Appendix A.

9.4.2 Experimental Protocol

Our proposed table detection method strongly relies on the proxy tasks of node and edge classification. Therefore, we adopted the *accuracy* measure to validate these tasks.

Following the evaluation schema introduced for object detection, in this work we

[6] Available at `http://sac.founderit.com/`
[7] `https://www.abbyy.com`

have considered that a table is properly detected if its *Intersection over Union* (IoU) is over a threshold. IoU is formally defined as

$$\text{IoU} = \frac{\text{area}(\text{GT}_{\text{bb}} \cap \text{DT}_{\text{bb}})}{\text{area}(\text{GT}_{\text{bb}} \cup \text{DT}_{\text{bb}})}, \tag{9.7}$$

where GT_{bb} and DT_{bb} are the table bounding boxes from the ground-truth and detection respectively. We denote IoU@t to specify the used threshold t. If it is not specified we will consider a table to be properly detected in case of $t = 0.5$, *i.e.* IoU@0.5.

Based on that, we have evaluate the table detection by means of the traditional measures used in information retrieval, the *precision* and *recall*. On the one hand, the precision evaluates the fraction of tables that have been actually detected. On the other hand, the recall is used to evaluate the fraction of detections that were actually tables. These are formally defined as

$$\text{Precision} = \frac{|\text{ret} \cap \text{rel}|}{|\text{ret}|},$$

$$\text{Recall} = \frac{|\,\text{ret} \cap \text{rel}|}{|\text{rel}|},$$

where *ret* are the set of predicted tables and *rel* stands for the set of relevant elements *i.e.* ground-truth tables. Moreover, we use the F_1 score to asses the overall performance of our system. This is formally defined as the harmonic mean of recall and precision values, thus

$$F_1 = 2\,\frac{\text{PrecisionRecall}}{\text{Precision} + \text{Recall}}.$$

F_1 score was also adopted by the cTDaR competition [87] to guide their final ranking.

9.4.3 Ablation Study

As introduced in Section 9.3 the proposed architecture depends on a set of parameters that we evaluated in the following lines. Let us first consider only one belief propagation iteration. We consider the GNN layer used, in this case GAT-based and GRU-based models. In addition, the number of layers and hidden size are also taken into account. Table 9.2 provides this study in terms of the average F_1 score of three different trains of our system. For these experiments, a table is considered properly detected with an IoU of 0.5 (IoU@0.5). Note that for RVL-CDIP and CON-ANONYM datasets, for different runs we randomly change the train, validation and test partition. In the case of cTDaR, the test partition remains the same for all the cases.

Table 9.2: Study on the GNN model and parameters. We provide the mean and standard deviation of 5 trains of the different models, number of layers (K) and hidden size (HS). In bold, the best performance per each model in %.

Model	K	HS	RVL-CDIP		CON-ANONYM		cTDaR	
			F_1 score	±	F_1 score	±	F_1 score	±
GAT-based	3	32	**55.90**	1.557	85.16	3.997	59.64	3.422
		64	49.38	1.840	86.14	2.231	61.52	2.007
		128	51.04	4.913	86.76	1.471	60.26	1.343
	4	32	52.96	3.688	86.14	2.187	59.20	1.560
		64	52.82	4.752	84.92	1.475	**62.14**	1.374
		128	54.56	5.261	87.54	3.173	60.08	2.315
	5	32	52.98	4.418	85.66	2.101	61.00	2.255
		64	55.44	2.411	**88.86**	1.874	61.22	1.731
		128	51.98	4.619	88.22	1.946	60.62	2.397
GRU-based	3	32	50.96	5.743	81.74	2.557	64.38	2.329
		64	51.68	6.149	84.12	1.988	65.84	1.767
		128	51.96	7.062	84.24	2.746	65.10	2.277
	4	32	52.40	4.840	86.86	2.147	69.02	1.988
		64	**57.22**	3.431	85.84	2.367	65.54	2.798
		128	51.00	4.740	85.80	2.570	64.92	3.220
	5	32	54.80	5.095	84.44	4.482	**70.10**	0.860
		64	55.34	4.786	**88.14**	1.626	66.98	3.558
		128	56.32	3.636	83.60	2.266	67.72	0.789

This table demonstrates that the GRU-based architecture outperforms the GAT-based version across the three datasets. This is likely to happen because of the use of the edge information along the different layers of the proposed model in the case of the GRU model. Additionally, increasing the number of layers helps in the detection of tables as it entails an increase of the receptive field allowing a better contextual information. Thus, nodes and edges are able to be better classified. However, this increase in terms of the number of layers should be carefully done reminding the typical problem of over-smoothing in GNNs. Regarding the hidden size, we conclude that increasing it too much is prejudicial to the network as it leads to over-fitting.

From the datasets perspective, CON-ANONYM is the one that better fits into the specification of the industrial use case and the one obtaining better performance. Moreover, RVL-CDIP contains noisier data which makes the detection task more challenging. Finally, the cTDaR dataset shows the importance of the visual cues. For example, some documents contain tables organized in blocks that at the same

Table 9.3: Table detection evaluation for the best models from Table 9.2 compared to our previous GNN model.

Task	RVL-CDIP			CON-ANONYM			cTDaR		
	F_1 score	Precision	Recall	F_1 score	Precision	Recall	F_1 score	Precision	Recall
GNN [175]	30.80	25.20	39.60	73.70	78.40	69.50	-	-	-
GAT-based	55.90	55.00	56.92	88.86	90.84	86.98	62.14	65.04	59.56
GRU-based	57.22	57.12	57.50	88.14	90.24	86.18	70.10	71.10	69.14

time can be considered as tabular arrangements. Section 9.4.4 discusses this problems in more detail.

From now on, we will consider the best models for each dataset according to their F_1 score which are marked in bold in the table mentioned above.

Table 9.3 provides a comparison in terms of F_1 score, precision and recall against our previous work [175]. With the improvement introduced in this chapter, we are able to improve that seminal work in both datasets, RVL-CDIP and CON-ANONYM, by a big margin. As we have already explained, our methodology strongly relies on the ability to classify nodes and edges to its corresponding class.

Table 9.4 provides an evaluation on these proxy tasks in terms of node classification accuracy, recall on table nodes and edge classification accuracy. Remember that for RVL-CDIP and CON-ANONYM several classes are taken into account for node classification. Moreover, the edge accuracy of our previous model [175] is not directly comparable as it does not consider the same three different classes.

Table 9.4: Node and edge classification performance as well as table node recall for the best models from Table 9.2.

Task	RVL-CDIP			CON-ANONYM			cTDaR		
	Node Acc.	Edge Acc.	Recall	Node Acc.	Edge Acc.	Recall	Node Acc.	Edge Acc.	Recall
GNN [175]	62.30	84.00	-	84.50	93.40	-	-	-	-
GAT-based	68.06	85.62	75.40	83.42	89.22	93.90	89.58	89.50	86.24
GRU-based	67.06	83.02	76.74	84.74	89.42	93.06	92.20	91.60	87.74

In comparison, the accuracies at node level are rather similar, however, thanks to the new edge information and the consensus step inspired on the belief propagation, we are able to improve our final detection score.

Finally, we have evaluated several iterations of the belief propagation algorithm as our consensus layer to perform the final node grouping. Figure 9.6 shows the evolution of the F_1 score for our best model in cTDaR dataset while increasing the number of belief propagation iterations. Observe that 0 corresponds to the direct prediction of the GNN. We have experimentally observed that increasing the number of layers is able to slightly improve the performance in these pages

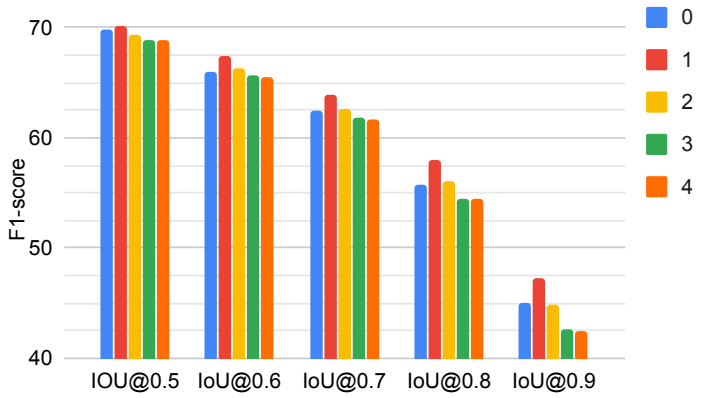


Figure 9.6: Evolution of the F_1 score using several iterations of the belief propagation and different IoU thresholds.

containing just one table. However, for two or more tables the performance decreases. From our experiments we conclude that the best option is achieved by incorporating just one iteration of the belief propagation algorithm. Even though the performance at IoU@0.5 is very similar, the belief propagation layer obtains more precise results. This is observed while increasing the IoU threshold.

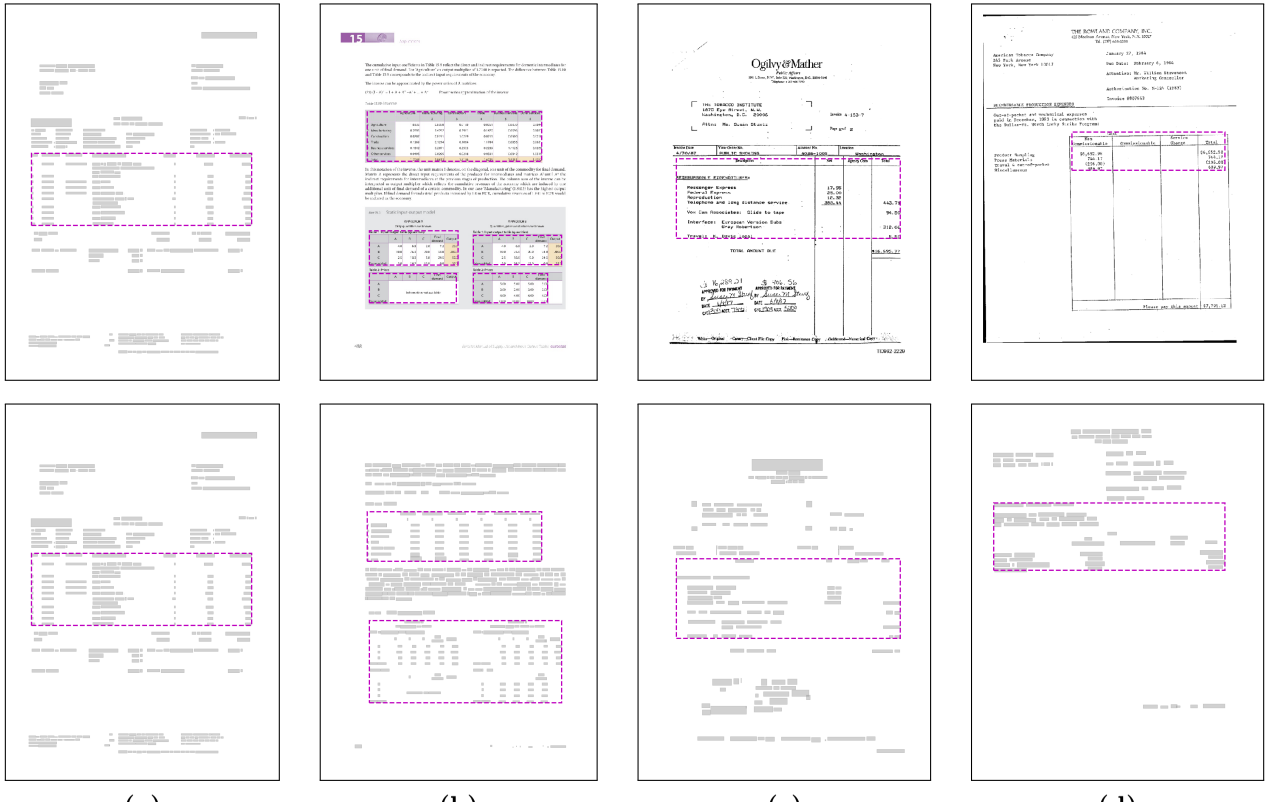
9.4.4 Structural Constraints

As previously stated in the introduction, table detection on undisclosed documents is a challenging task, not only because in administrative documents the classic table structure is sometimes lost, but also, the visual cues exploited by the state-of-the-art table detection algorithms are completely lost. These cues include attributes such as rules, colors, or text emphasis styles (bold, italics, underline, etc.).

Figure 9.7 shows the table detection ground-truth of the same documents as Figure 9.1. Observe that this is not a straightforward task even for humans. For instance, Figure 9.7(b) presents several tables that at a high level might be perceived as a unique table or at least a set of tables in a tabular arrangement. In this particular case, tables are easily spotted using the rule lines and distinct colors.

Moreover, if we compare our results with the ones obtained by the participants in cTDaR competition[8] we get an idea of the importance of the visual cues in this task. Note that these results are not directly comparable to our proposed approach as we do not aim for an accurate segmentation rather than an adjusted bounding box to the detected nodes. Indeed, since we are not using the visual information, it is not possible to include elements such as the table lines into our

[8]Available at `http://sac.founderit.com/results.html`



(a) (b) (c) (d)

Figure 9.7: Example of ground-truth table regions on the original documents (first row) and the predicted ones on the anonymized documents (second row).

Table 9.5: Comparison against the top-3 approaches from the cTDaR competition in terms of F_1 score at different IoU thresholds.

Team	IoU@0.5	IoU@0.6	IoU@0.7	IoU@0.8	IoU@0.9
TableRadar	-	97.16	96.41	95.27	91.12
NLPR-PAL	-	96.24	95.49	93.61	89.47
Lenovo Ocean	-	94.07	93.69	91.40	86.04
GRU-based	70.10	67.40	63.84	58.02	47.24

final detection. Table 9.5 shows a comparison against the top-3 approaches from the cTDaR competition in terms of F_1 score. This table provides the results at different thresholds moving from a coarser segmentation to a fine grained one.

9.5 Conclusions

This chapter has presented, as far as we are concerned, the first table detection method based purely on structural information without making use of the raw content of the text. After modeling the underlying structure of the document as a graph, the table detection is treated as a node classification problem where local node configurations characterize the table structure. Graph neural networks provide an adequate framework to discover the local structures and to classify these nodes belonging to tables. Additionally, we have contributed to the community a novel dataset derived from the RVL-CDIP invoice data.

From the industrial point of view, an important advantage is that our approach is able to deal with anonymized data, as it does not use the raw textual contents of the documents. While most existing works on table detection do not consider anonymization, which is a big concern for companies when dealing with sensitive content as in the case of invoices, our method has demonstrated to overcome this limitation.

Despite demonstrating a good performance for table detection, especially if we take into account the origin of the documents and the drop of information due to the anonymization stage, there are still some open challenges that might be studied in the future. Firstly, the proposed framework relies on a commercial OCR which is used in the production pipeline of our partner company. Therefore, it might not be able to recover text blobs in really small cells. Finally, our pipeline strongly depends on node and edge classification rather than directly train on region detection which is our final goal.

10 | Conclusions

Now it's closing time, the music's fading out
Last call for drinks, I'll have another stout.

– Tom Waits

In this chapter, we summarize the contributions of this dissertation to the pattern recognition and computer vision fields and, in particular, its applications to document image analysis and recognition problems. We also highlight the main achievements and limitations of the proposed approaches. Finally, we lead the reader towards possible new research lines and natural extensions of the proposed methodologies.

10.1 Summary of the Contributions

In this thesis we have introduced a study on how graphs can be used for several pattern recognition and computer vision tasks and, in particular, for document analysis and recognition applications. The mass digitization of document collections has been promoted worldwide in order to foster their preservation in digital libraries. Although it has been specially applied to historical collections, digitization has become an important part for companies which require to rapidly process a huge amount of administrative documentation. For example, in the era of the digital revolution, digital-born documents have become the main communication channel. However, there are still lots of documents shared in paper, either printed, completely handwritten or combined, *i.e.* signed documents or filled forms.

Our starting hypothesis was that document collections have an important structural component that can be exploited in several methodologies. The main idea, as introduced in Chapter 1, was to exploit this structural information in document collections. Research-wise, we decided to divide this thesis in two main parts following two separated trends on graph-based representation algorithms. On the one hand, traditional approaches on graph matching and embeddings have been presented. On the other hand, the new advances on geometric deep learning had been taken into account to incorporate deep learning features on two distinct applications.

The contributions presented in this work are enumerated in six points, three of them corresponding to the first part of this thesis and the other three corresponding to the second part. Moreover, even though the focus of this thesis is the development of DIAR methodologies, some of the contributions are generic algorithms for graph data. Let us briefly summarize these five contributions:

- **Construction of a graph representation**: Utilizing graphs as a representational scheme for DIAR algorithms involves an expensive handcrafted process to design a model for describing the structure of images. In Chapter 3, we have demonstrated that a proper graph representation is able to robustly depict handwritten word images. In particular, we have shown competitive results when comparing to traditional statistical approaches in the context of the word spotting task. Therefore, we have validated these representations as a valid alternative.

- **Efficient graph retrieval**: Efficiency has traditionally been one of the main drawbacks when dealing with these graph representations. Specially, in large-scale scenarios, graphs have been considered to be unrealistic schemes due to the fact that most of the algorithms to process graph data have, at least, a quadratic computational complexity. In this dissertation, Chapter 4 and Chapter 5 have proposed two different methodologies to overcome this limitation. First, an indexation scheme based on the local context of the nodes has been introduced. Therefore, graph matching has been applied only to these regions likely to contain the desired information. Second, a hierarchical graph has been built in order to perform a matching stage on much smaller graphs. In comparison to the indexation scheme, this second methodology kept a global consistency of the graph. Both techniques have been carefully validated on top of a graph edit distance approach.

- **Hierarchical graph embedding**: As introduced in our review about graph theory in Chapter 2, graph embeddings have been widely used to encode structural representations into the Euclidean domain, *i.e.* one dimensional data. However, obtaining this vectorial representation entails a loss on the structural information. In Chapter 6 we have proposed a novel graph embedding technique which exploits the hierarchical graph structure defined in its previous chapter. This embedding emphasizes the structural nature of the original graph making use of a stochastic graphlet sampling. Moreover, we have shown the superiority of our approach not only on document image analysis and recognition datasets, but we extended our validation by means of classical pattern recognition data such as molecular graphs.

- **Graph metric learning**: The new advances on deep learning frameworks dealing with non-Euclidean data, and in particular graphs, have faded away previous schemes such as the graph edit distance. In Chapter 8, we have proposed to take advantage of these traditional approaches, in our case the Hausdorff edit distance, to guide our learning process. In this chapter, graph

neural networks were used to learn, on the one hand, an enriched graph representation and, on the other hand, the edit cost operations of the proposed GED.

- **Automatic processing of administrative documents**: Nowadays, one of the main research topics of the DIAR area focuses on the automatic processing of administrative documents. In this thesis, we focused on the problem of table detection as tables are the drivers of the essential information condensed in a structured text. The main problem is the sensitive data involved in these documents. Thus, public databases are rather scarce and old fashioned as real documents are confidential and cannot be published. Therefore, these documentation is usually treated internally in each corporation. Hence, we put our efforts on a model which does not rely on the textual content. In addition, we released the generated data.

- **Applications in a real use case scenario**: A side contribution of this thesis has been a collaboration with a company dealing with the automatic processing of administrative documents. Therefore, on the one hand, we have faced real problems according to the needs of the industry and, on the other hand, we transferred the generated knowledge to the company.

10.2 Discussion

This thesis has made several contributions in image classification, recognition and retrieval fields using graphs. Although the application domain that has driven our research is document image analysis and recognition, most of our contributions are transversal and can be formulated in terms of graph classification, recognition and retrieval. For example, Chapter 6, introduces a graph embedding which has proved to be state-of-the-art not only for graph-based image classification but also in molecular graph datasets.

Even though during this thesis graph-based representations have been used in a wide range of applications, there are particular cases where they are not the adequate representational framework. For example, starting from Chapter 3, word spotting has been one of the problems tackled during this research. In this case, graphs provide a flexible and powerful representation able to depict the structure of the hand-drawn strokes while preserving the deformations from different writing styles. Therefore, we are able to formulate this problem in a completely learning free approach, which, in addition, provides the node correspondence between words. However, with the advances on deep learning frameworks, CNNs can easily overcome our performance given a carefully annotated set of training words. Although in the particular case of historical manuscripts it is not always feasible to have this requirement, in modern documents we consider CNNs a better solution to this problem.

An important drawback we had to face in this work is the time complexity involved in any graph based approach. Chapters 4 and 5 have proposed solutions to scale graph retrieval techniques to large collections of images. In general, traditional statistical approaches are able to work on reasonably large datasets without any indexing methodology. Nonetheless, researchers have proposed hashing techniques to allow these set of approaches to work at real time, a setting where graph-based representations are unlikely to be used. Thus, in Chapter 6 we tried to bridge the gap between statistical and graph-based approaches by defining a multi-scale graph embedding function.

The second part of this dissertation consists of two disjoint approaches which share the geometric deep learning methodology. We prove how geometric deep learning is a game changer for the structural pattern recognition community. For instance, Chapter 8 demonstrates that traditional graph matching approaches can be combined with graph neural networks in order to obtain an improved matching methodology. In comparison to other works, we argue that it is not necessary to learn a graph embedding which will cause a loss on the structural information, specially, in big graphs. In addition, even though the use cross-graph connections, as proposed in the GMN [138] approach, increases the performance of the learned metric, it brings about the time complexity.

Finally, Chapter 9 has presented a work able to detect tables in an anonymized administrative documents. Although our proposed framework is capable of performing such task, we hypothesize that a full layout analysis system might not be possible. In the particular case of non-structured document regions, the proposed approach is not able to exploit neither the textual contents nor the visual cues required for such cases. Hence, in these scenarios, CNN-based approaches, which are able to make use of the visual information or natural language processing frameworks given the textual content obtained by an OCR, might be a better choice.

10.3 Open Challenges

Along the thesis we have already stressed some open questions worth considering as unexplored lines. Moreover, taking into account the novelty of the geometric deep learning frameworks, we are convinced that there is still a wide variety of opportunities for improving and advancing our work. Also note that the new methodologies derived from the geometric deep learning field has opened several research lines that were not covered in this dissertation.

Traditionally, the generation of graph-based representations from a given image has been performed using handcrafted algorithms, where expert researchers carefully designed the whole process. With the emergence of geometric deep learning, as reviewed in Chapter 3, scene graphs have been proposed as a way to construct a graph-based representation from a given image. However, in the literature,

scene graphs are trained from a manually annotated ground-truth based on object detection and relationship connections. We speculate that by taking advantage of these techniques, a graph can be constructed in an unsupervised way for such a task that strongly requires structural information. Therefore, a first open challenge is to design pure data-driven techniques able to exploit the structure required to solve a particular task.

Even though in Chapter 8 we have introduced a graph metric learning framework able to learn, on the one hand, similarities between graphs and, on the other hand, its corresponding node assignments in terms of substitutions, insertions and deletions, our method has still some limitations. The main one is the lack of a global graph coherence at the node to node assignment. Following classical approximations of the graph edit distance, in order to avoid a huge time complexity, specially in big graphs, we have only considered the local context of the nodes at matching time. Therefore, an interesting future research direction is to study how to extend the proposed methodology to incorporate a global consistency on the node to node assignments.

With regard to Chapter 9, which has presented a work able to detect tables in anonymized administrative documents, the method's main limitation is the loss of information during the data anonymization stage. At that time, several information regarding visual cues, *i.e.* font, color, rule lines, etc. can be extremely helpful in combination with our structural methodology. As a matter of fact, an open challenge is to develop techniques able to combine structural as well as visual information on layout analysis techniques.

Although it is not directly related to the main concern of this thesis, which is to automatic process images from the structural perspective, a topic worth exploring is to incorporate the use of knowledge graphs. Note that knowledge graphs can be treated using similar techniques as the ones introduced in this dissertation, even though its encoded information is intrinsically different.

Finally, deep learning is experiencing an evolution from the point of view of the learning strategies. The huge amount of data required for the supervision of new models causes a huge bottleneck dealing with new problems. Therefore, self-supervision and reinforcement learning strategies are gaining popularity among the machine learning community. We hypothesize that similar approaches will be able to deal properly with graph data. As a matter of fact, we performed some tests on the reinforcement learning setting in order to combine, using its pairwise relationships, several blocks or shapes. Reinforcement learning has proved to lead to meaningful results taking into account a very restricted set of shapes, however, moving to a large scale setting has proved to be very unstable.

Appendix

Along the chapters of this dissertation, we have conducted several experiments to evaluate the performance of our proposed approaches. Therefore, several evaluation metrics and benchmarks have been used. Here, we present an appendix detailing carefully the benchmarks that have been used to asses this performance.

A | Datasets

It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

– Sir Arthur Conan Doyle

Along this thesis, several databases have been used to evaluate the performance of the proposed frameworks. Although this thesis focuses on DIAR problems, in order to test the generality of the proposed approaches, we have also experimented in other datasets developed in Pattern Recognition. In particular, apart from the DIAR datasets, we make use of computer vision datasets for image classification and molecular graph datasets for graph classification. Moreover, the selected datasets are able to evaluate different settings such as classification, retrieval or detection.

A.1 Barcelona Historical Handwritten Marriages Database

The *Barcelona Historical Handwritten Marriages Database* (BH2M) presented in [76] consists of 174 pages from the 17th century. The pages are extracted from one volume of the Marriage Register Books from the Archive of the Barcelona Cathedral. This collection is composed of 244 books with information of approximately 600,000 unions celebrated in 250 parishes between 1451 and 1905. Figure A.1 shows two pages from different volumes of the Marriage Register Books. Note that Figure A.1(a) belongs to the volume used to create the dataset BH2M.

The BH2M dataset contains ground truth at several levels. Thus, it provides a challenging set of collections for several DIAR problems:

- **Layout structure**: The layout information is provided at text block, paragraph, line and word levels. Therefore, this dataset is appropriate for evaluating layout analysis methodologies.

(a) **(b)**

Figure A.1: Examples of pages form different volumes from the Marriage Register Books from the Archive of the Barcelona Cathedral.

- **Content analysis**: All the handwritten text has been carefully transcribed. This dataset has been widely used for word spotting and handwritten text recognition (at both word or line level) tasks.
- **Semantic tasks**: In addition, thanks to the nature of this collection, semantic information has been provided such as names, dates or places.

In this thesis, this dataset has been used for the particular task of word spotting. The use of word spotting in this collection allows to search names, places, occupations, etc. The dataset is divided in 3 sets, train, validation and test with 100, 34 and 40 page images respectively. The ground truth consists of the bounding box and transcription of each one of the words. Together with the dataset, the authors provide a total of 5,170 query words, cropped from these manuscripts. Figure A.2 illustrates two word examples from this dataset.

A.2 IAM Graph Database Repository

The IAM graph database repository[1] [180] provides several graphs generated from different types of data. Thus, it is a multidisciplinary dataset which allows re-

[1] Available at `http://www.fki.inf.unibe.ch/databases/iam-graph-database`

(a) Berga **(b)** Casanovas

Figure A.2: Examples of cropped words from the BH2M dataset with their transcription.

searchers to evaluate their performance independently of the application. Therefore, the dataset is divided in nine datasets named, Letter (3 versions), Digit, GREC, Fingerprint, COIL (2 versions), Webpages, AIDS, Mutagenicity and Protein.

Among these datasets, in this dissertation we considered three of them *viz. AIDS*, *GREC* and *COIL-DEL*.

- **AIDS graphs**: It consists of 2000 graphs representing molecular compounds which are constructed from the AIDS Antiviral Screen Database of Active Compounds[2]. This dataset consists of two classes, *viz.* , active (400 elements) and inactive (1600 elements), which respectively represent molecules with possible activity against HIV.
- **GREC graphs**: It consists of 1100 graphs representing 22 different classes (characterizing architectural and electronic symbols) with 50 instances per class; these instances have different noise levels.
- **COIL-DEL graphs**: It includes 3900 graphs belonging to 100 different classes with 39 instances per class. It has been created as the graph representation of a subset of the *Columbia Object Image Library* [160] reviewed in the following section.

Table A.1: Details of the AIDS, GREC, COIL-DEL and HistoGraph datasets.

Datasets	Subsets	# Graphs	# Classes	Avg. $\|V\|$	Avg. $\|E\|$	Node labels
AIDS	–	2000 (250, 250, 1500)	2	15.7	16.2	Chemical symbol
GREC	–	1100 (286, 286, 528)	22 (50 each)	11.5	11.9	Type, (x,y) position
COIL-DEL	–	3900 (2400, 500, 1000)	100	21.5	54.2	(x,y) position

The IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning [180] presents ten graph sets with different characteristics. Figure A.3 shows two examples from two classes. These sets come from real problems that can be faced through graph-based representations.

[2]See at `http://dtp.nci.nih.gov/docs/aids/aids_data.html`

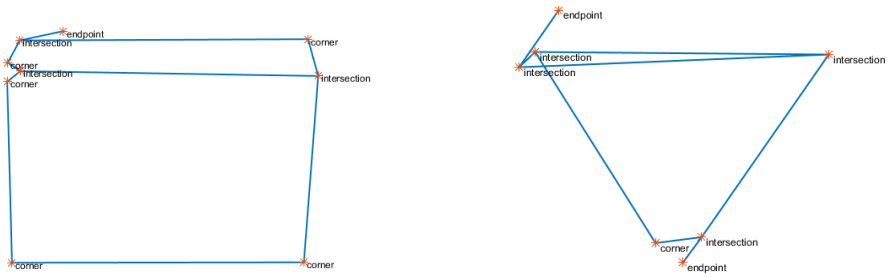
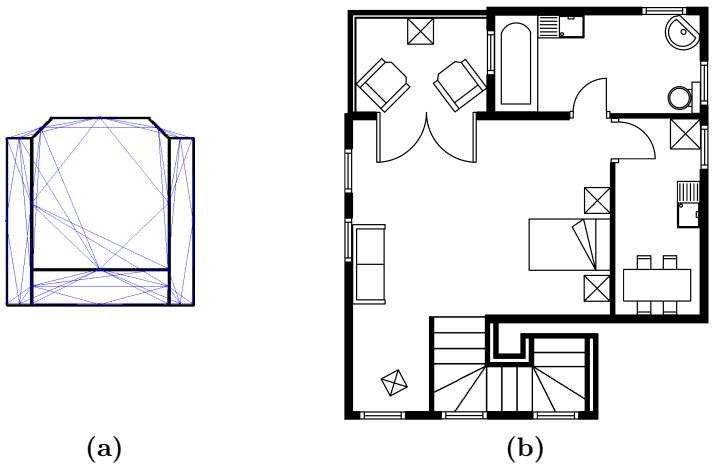


Figure A.3: Examples of graphs from two classes of the dataset.



(a) **(b)**

Figure A.4: Examples of the elements in our database. (a) graph representation of a query; (b) floor plan image where the query should be spotted.

A.3 SESYD Floorplans

The *Systems Evaluation SYnthetic Documents*[3] (SESYD) dataset, created by Delalandre *et al.* [55] is a database of synthetical documents organized in several categories. In particular, for the purposes of this dissertation, we consider the floorplans collection which contains 10 different subsets and 16 query symbols. Figure A.4 shows two example images from the SESYD dataset.

Each one of the subsets contains 100 synthetically generated floorplans created from the same floorplan template. Then, different model symbols, 16 in total, are placed in randomly realistic locations, orientations and scales.

In this thesis, we built up a new subset by combining the original subsets. Thus,

[3] `http://mathieu.delalandre.free.fr/projects/sesyd/`

apart from the symbol distortions, we are also taking into account variations on the floorplan. Our new subset consists of 20 floorplans taken from each subset and all the query symbols.

This dataset was previously processed by Dutta *et al.* [63]. They propose a dual graph representation of both, symbols and floorplans. Following this representation, dual graphs nodes are labeled with the vector of Hu moments invariants. Dual nodes represent a path in the vectorized image.

A.4 Object classification datasets

Two datasets designed for object classification have been used during this thesis,

- The *Columbia Object Image Library* (COIL-100) [160] consists of 100 images of different objects against a black background. For each object, 72 images at equally spaced poses, corresponding to different rotations of the object have been taken.
- *Object DataBank* (ODBK) [212] is formed by 209 3D objects taken into account images of 14 different views per each.

For both datasets, graph nodes are extracted using the *Harris corner detector* [105] and the edges are generated using the Delaunay triangulation on these nodes. In both datasets, the final graphs node attributes correspond to each coordinate in the image whereas the edges are not weighted. Following the experiments reported by Mousavi *et al.* [158], 15 and 50 classes, with maximum average number of nodes, are used. The graphs are divided into three sets, training, validation and test of 360, 75 and 150 images for COIL-100 dataset and 300, 150 and 150 images for ODBK dataset. Figure A.5 shows some example images from these databases.

A.5 Molecular Graph Datasets

Several bioinformatics datasets have been used in this dissertation, *viz. MUTAG*, *PTC*, *PROTEINS*, *NCI1*, *NCI109*, *D&D* and *MAO*. These datasets have been widely used as benchmark in the literature.

- The *MUTAG* [53] dataset contains graph representations of 188 chemical compounds which are either mutagenic aromatic or heteroaromatic nitro compounds where nodes can have 7 discrete labels.
- The *PTC* [214] or Predictive Toxicology Challenge dataset consists of 344 chemical compounds known to cause or not cause cancer in rats and mice. It has 19 discrete node labels.

(a) **(b)**

Figure A.5: Example of objects from (a) the COIL-100; and (b) ODBK databases.

- The *PROTEINS* [24] dataset contains relations between secondary structure elements (SSEs) represented by nodes and neighborhood in the amino-acid sequence or in 3D space by edges. It has 3 discrete labels *viz. helix*, *sheet* or *turn*.
- The *NCI1* and *NCI109* [222] come from the National Cancer Institute (NCI) and are two balanced subsets of chemical compounds screened for their ability to suppress or inhibit the growth of a panel of human tumor cell lines, having 37 and 38 discrete node labels respectively.
- The *D&D* [59] dataset consists of enzymes and non-enzymes proteins structures, in which their nodes are amino acids.
- The *Monoamine Oxydase* (MAO) database, taken from GREYC Chemistry graph dataset collection[4], is composed of 68 graphs representing molecules that either inhibit or not the monoamine oxidase, which is an antidepressant drug.

Some more details on the proposed bioinformatics molecular datasets are provided in Table A.2.

A.6 HistoGraph Database

The HistoGraph dataset [207, 208] is a graph database for historical keyword spotting evaluation[5]. It consists of different well known manuscripts.

George Washington (GW) [79]: This database is based on handwritten letters written in English by George Washington and his associates during the American

[4]Available at `https://brunl01.users.greyc.fr/CHEMISTRY/`
[5]Available at `http://www.histograph.ch/`

Table A.2: Details of the molecular graph datasets.

Datasets	# Graphs	# Classes	Avg. $\|V\|$	Avg. $\|E\|$	Node labels
MUTAG	188	2 (125 vs. 63)	17.9	39.6	7
PTC	344	2 (192 vs. 152)	25.6	51.9	19
PROTEINS	1113	2 (663 vs. 450)	39.1	145.63	3
NCI1	4110	2 (2057 vs. 2053)	29.9	64.6	37
NCI109	4127	2 (2079 vs. 2048)	29.7	64.3	38
D&D	1178	2 (691 vs 487)	284.3	1431.3	82
MAO	68	2 (30 vs. 38)	18.4	19.6	3

Revolutionary War in 1755[6]. It consists of 20 pages with a total of 4,894 handwritten words. Even tough several writers were involved, it presents small variations in style and only minor signs of degradation.

Parzival (PAR) [79]: This collection consists of 45 handwritten pages written by the German poet Wolfgang Von Eschenbach in the 13th century. The manuscript is written in Middle High German with a total of 23,478 handwritten words. Similarly to GW, the variations caused by the writing style are low, however, there are remarkable variations caused by degradation.

Alvermann Konzilsprotokolle (AK) [169]: It consists of German handwritten minutes of formal meetings held by the central administration of the University of Greifswald in the period of 1794 to 1797. In total 18,000 pages were used with small variations in style and only minor signs of degradation.

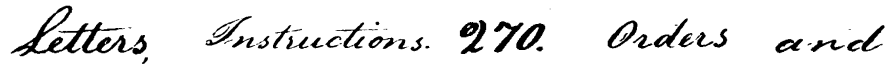
Botany (BOT) [169]: It consists of more than 100 different botanical records made by the government in British India during the period of 1800 to 1850. The records are written in English and contain certain signs of degradation and especially fading. The variations in the writing style are noticeable especially with respect to scaling and intra-word variations.

Figure A.6 provides some examples of pre-processed word images from which the graphs are created. Observe that the word segmentation of AK and BOT datasets is imperfect [169]. Moreover, these two datasets do not provide a validation set. Table A.3 provides an overview of the dataset in terms of number of words.

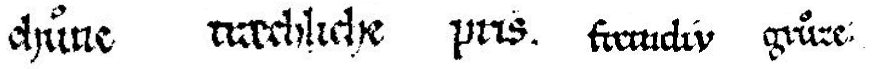
A.6.1 Graph Construction

One of the objectives of this dataset was to evaluate the performance of different graph representations generated from the same input data. Therefore, the authors developed 6 different graph representation paradigms for delineating a single word

[6]George Washington Papers at the Library of Congress from 1741-1799, Series 2, Letterbook 1, pages 270-279 and 300-309, `https://www.loc.gov/collections/george-washington-papers/about-this-collection/`

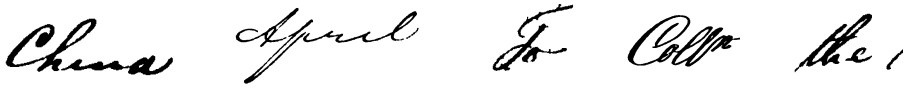


(a) George Washington (GW)



(b) Parzival (PAR)

(c) Alvermann Konzilsprotokolle (AK)



(d) Botany (BOT)

Figure A.6: Pre-processed word examples of the four datasets.

Table A.3: Dataset overview in terms of number of keywords and word images for training, validating and testing respectively

Dataset	Keywords	Train	Validation	Test
GW	105	2,447	1,224	1,224
PAR	1,217	11,468	4,621	6,869
BOT	150	1,684	-	3,380
AK	200	1,849	-	3,734

into a graph. This results in 6 different versions of this dataset. In this section, we briefly explain how these graphs have been constructed as explained in [207].

- **Keypoint**: This methodology extracts characteristic points from the skeletonized word images. The set of keypoints are based on start, end, and junction points. Thus, in the final graph representation these points are represented as nodes and labeled with their corresponding (x, y)-coordinates. Moreover, between connected points in terms of the skeleton, intermediate points at equidistant intervals are also converted into nodes. Finally, edges are created between these nodes connected by a stroke.
- **Grid**: Firstly, a grid-based segmentation is performed to the binarized word images. Each cell which contains foreground pixels, is converted into a node with the corresponding (x, y)-coordinates of the center of mass as a label.

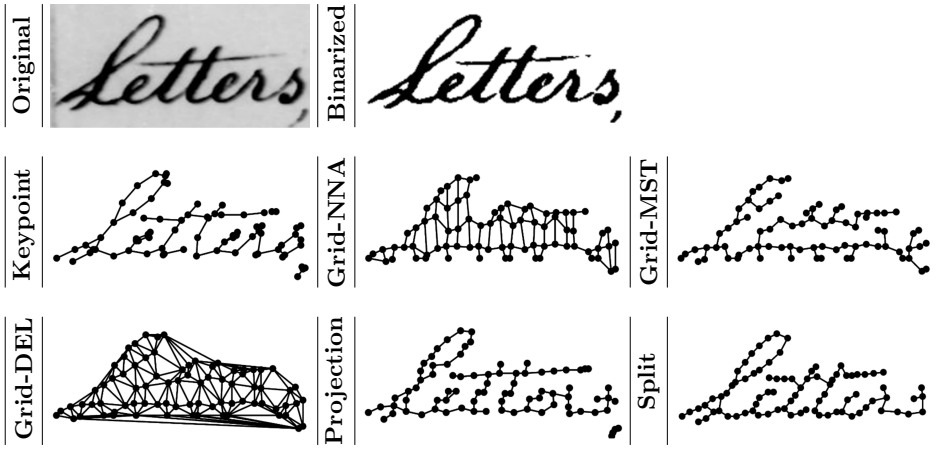


Figure A.7: Overview of the graph representations proposed by Stauffer *et al.* . Reprinted from [207].

Moreover, undirected edges are inserted according to one of three edge insertion algorithms, *viz.* Node Neighborhood Analysis (Grid-NNA), Minimal Spanning Tree (Grid-MST), or Delaunay Triangulation (Grid-DEL). Note that only Grid-MST is used for the whole dataset whereas Grid-NNA and Grid-DEL are only used for the evaluation of the subset for graph classification introduced in Section A.6.2.

- **Projection**: This methodology, rather than a static grid, is based on an adaptive and threshold-based segmentation of binarized word images. It is based on the horizontal and vertical projection profiles. Thus, the image is split at middle of white spaces of the projection profile and then it is further divided in equidistant intervals. A node is inserted into the graph labeled with the (x, y)-coordinates of the skeleton point closest to the center of mass of the cell. Finally, an edge is created if a pair of nodes is connected by a stroke.

- **Split**: This methodology iterative segments the binarized word image until the width and height of all segments are below a certain threshold. Then, a node is inserted into the graph labeled with the (x, y)-coordinates of the skeleton point closest to the center of mass of the cell. Finally, following the same idea as projection, an edge is created if a pair of nodes is connected in the skeleton.

Figure A.7 illustrate the different graph representations they proposed. For further details, we point the interested reader to the original work by Stauffer *et al.* [207].

A.6.2 Subset for Graph Classification

The HistoGraph dataset provides a subset of the George Washington data for the evaluation of the word graph classification problem. It consists of 293 graphs generated from 30 distinct words. Therefore, given a word, the task of the classifier is to predict its class which should be among the 30 words. Nodes are only labeled with their position in the image. The entire dataset is divided into 90, 60 and 143 graphs respectively for training, validation and test purposes. Table A.4 provides a summary of this subset.

Table A.4: Details of the HistoGraph dataset for graph classification.

Subsets	# Graphs	# Classes	Avg. $\|V\|$	Avg. $\|E\|$	Node labels
Keypoint	293 $(90, 60, 143)$	30	101.8	94.8	(x,y) position
Grid-NNA			56.4	81.4	
Grid-MST			66.1	64.4	
Grid-DEL			74.1	205.1	
Projection			63.1	58.8	
Split			73.3	69.8	

A.7 Table Detection Datasets

Three datasets have been used for the evaluation of the proposed table detection framework. Even though the evaluated task is the same, the typology of the documents presents a huge variability. Therefore, they are useful to evaluate our framework in different use cases.

- **CON-ANONYM**: This is a particular dataset of 960 documents which has been used as part of an industrial collaboration. Therefore, it is only available for the internal use in **omni:us** and in cannot be made publicly available as it contains sensitive contents. The documents are annotated with 8 region labels including the following *claim*, *car*, *cost*, *supplier*, *insured*, *mixed*, *other* and *table*.
- **RVL-CDIP**: The original dataset proposed by Harley *et al.* [104] was designed for the evaluation of CNNs for the task of document image classification and retrieval. It contains $400,000$ grayscale images, which are divided in 16 classes with $25,000$ images per class. During the development of this dissertation, we made use of a subset of the documents for the evaluation of our table detection framework. We selected 518 images from the *invoice* class, which have been annotated with 6 regions labels including *supplier*, *other*, *table*, *reciever*, *invoice_info* and *total*.

- **cTDaR**: Table detection and recognition has become a fundamental step for any information retrieval technique working on structured and semi-structured documents. The *ICDAR 2019 Competition on Table Detection and Recognition* (cTDaR)[7] [87] is one of the many research events that have recently rised attention to the table analysis problem. This competition is divided into two tracks and two data typologies per track.

  - **Track A**: Table detection in document images containing one or several tables. The ground truth is provided as the bounding box of the tables per document.
  - **Track B**: Table recognition in the same document images as the first track. Now, the cell structure of the table should be provided, *i.e.* cell bounding boxes and the corresponding starting and ending row and column index. However, table recognition is divided in two subtracks,
    - *Track B.1*: In this subtrack, the document image is provided jointly with the table bounding box. Therefore, it evaluates only the performance on parsing the table structure.
    - *Track B.2*: In the second subtrack, the document image is provided alone. Therefore, the table should be first detected and, afterwards, the cell structure is analyzed.

  Along with these two tracks, two document typologies are provided.

  - **Modern dataset**: 840 document images, generated from born-digital formats such as PDF. These documents contain Chinese and English documents collected from several sources such as scientific journals, forms or financial statements.
  - **Archival dataset**: The provided documents were gathered from more than 23 institutions around the world. The documents do not follow a unique structure but they have a great diversity. For example, images where collected from hand-drawn accounting books, stock exchange lists, train timetables, prisoner lists and many more. In total 840 document images were collected.

  In this work, we have utilized the modern dataset as it contains data closer to the proposed use case of invoices.

Moreover, due to the anonymization step, we only aim to compare ourselves on the task defined for the track A. Table A.5 shows a comparison of the datasets. Note that in CON-ANONYM and RVL-CDIP are datasets containing pages without tables.

[7] Available at `http://sac.founderit.com/`

Table A.5: Summary of the datasets statistics as well as the proposed division into training, validation and test sets.

	CON-ANONYM	RVL-CDIP	cTDaR
Total # documents	950	518	840
(tr, va, te)	(665, 95, 195)	(362, 52, 104)	(540, 60, 240)
Total # pages	1252	518	840
Total # tables	1202	485	1423
Total # classes	8	6	2

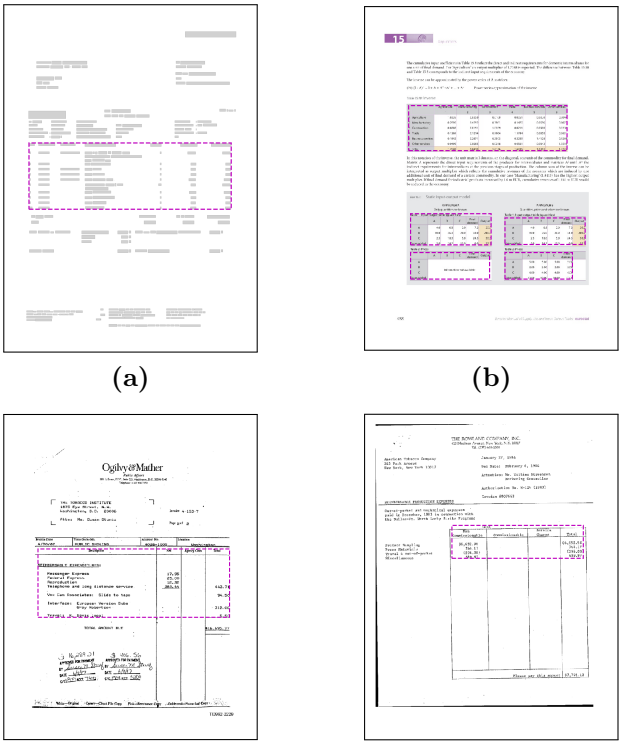


Figure A.8: Overview of the table detection datasets. The provided examples depict the table regions as provided in the ground-truth. The images corresponds to (a) CON-ANONYM dataset; (b) cTDaR dataset; (c-d) RVL-CDIP dataset

List of Contributions

For a moment, nothing happened.
Then, after a second or so, nothing continued to happen.

– Douglas Adams

---

*During the development of this dissertation, several contributions to the community have been developed. In this chapter, the Scientific Communications and leaded projects are listed. Authors marked with * indicate equal contribution on that work.*

Topics

The main topic of this dissertation, is the development of graph-based frameworks in the field of document image analysis and recognition. However, this thesis has also generated other side contributions in other topics that had raised our attention on the same field.

- **Handwritten Text Recognition** (HTR): It is the task of transforming images of handwritten text into machine readable format. The proposed system should deal with the huge variability of handwritten text from different writers.
- **Optical Music Recognition** (OMR): It is the task of transcribing a music score into a machine readable format. The formulation should not only, similarly to HTR, recognize the symbols appearing on the music score, which indicate the rhythm, but also their position in the staff indicating the pitch.
- **Sketch-based Image Retrieval** (SBIR): This topic investigates the problem of sketch-based image retrieval, where human sketches are used as queries to conduct retrieval of photos. In particular, we focused on the problem from the zero-shot point of view, where the evaluated sketches came from unseen categories.

Journals

1. **Pau Riba**, Josep Lladós, Alicia Fornés and Anjan Dutta. Large-scale Graph Indexing using Binary Embeddings of Node Contexts for Information Spotting in Document Image Databases. In *Pattern Recognition Letters*, Vol.87, pp.203–211, 2016, (Q2)

2. Arnau Baró, **Pau Riba**, Jorge Calvo-Zaragoza and Alicia Fornés. From Optical Music Recognition to Handwritten Music Recognition: a Baseline. In *Pattern Recognition Letters*, Vol.123, pp.1–8, 2019, (Q2)

3. **Pau Riba**, Josep Lladós and Alicia Fornés. Hierarchical graphs for coarse-to-fine error tolerant matching. In *Pattern Recognition Letters*, Vol.134, pp.116–124, 2020, (Q2).

4. Anjan Dutta*, **Pau Riba***, Josep Lladós and Alicia Fornés. Hierarchical Stochastic Graphlet Embedding for Graph-based Pattern Recognition. In *Neural Computing and Applications*, 2020, (Q1)

5. Lei Kang, **Pau Riba**, Mauricio Villegas, Alicia Fornés and Marçal Rusiñol. Candidate Fusion: Integrating Language Modelling into a Sequence-to-Sequence Handwritten Word Recognition Architecture. In *Pattern Recognition* 2020, (Q1; *Submitted*)

International Conferences

1. **Pau Riba**, Jon Almazán, Alicia Fornés, David Fernández-Mota, Ernest Valveny and Josep Lladós. e-Crowds: a mobile platform for browsing and searching in historical demography-related manuscripts. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp.228–233, 2014.

2. David Fernández-Mota, **Pau Riba**, Alicia Fornés and Josep Lladós. On the Influence of Key Point Encoding for Handwritten Word Spotting. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp.476–481, 2014.

3. Olivier Lefebvre, **Pau Riba**, Jules Gagnon-Marchand, Charles Fournier, Alicia Fornés, Josep Lladós and Réjean Plamondon. Monitoring Neuromotricity Online: a Cloud Computing Approach. In *Proceedings of the Biennial Conference of the International Graphonomics Society*, pp.63–66, 2015.

4. **Pau Riba**, Josep Lladós, Alicia Fornés and Anjan Dutta. Large-scale graph indexing using binary embeddings of node contexts. In *Graph-Based Representations in Pattern Recognition*, Vol.9069, pp.208–217, 2015.

5. **Pau Riba**, Josep Lladós and Alicia Fornés. Handwritten Word Spotting by Inexact Matching of Grapheme Graphs. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp.781–785, 2015.

6. Arnau Baró, **Pau Riba** and Alicia Fornés. Towards the recognition of compound music notes in handwritten music scores. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp.465–470, 2016.

7. **Pau Riba**, Alicia Fornés and Josep Lladós. Towards the alignment of handwritten music scores. In *Graphic Recognition. Current Trends and Challenges*, Vol.9657, pp.103–116, 2017.

8. **Pau Riba**, Alicia Fornés and Josep Lladós. Error-Tolerant Coarse-to-Fine Matching Model for Hierarchical Graphs. In *Graph-Based Representations in Pattern Recognition*, Vol.10310, pp.107–117, 2017.

9. Anjan Dutta, **Pau Riba**, Josep Lladós and Alicia Fornés. Pyramidal Stochastic Graphlet Embedding for Document Pattern Classification. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp.33–38, 2017.

10. **Pau Riba**, Anjan Dutta, Sounak Dey, Josep Lladós and Alicia Fornés. Improving Information Retrieval in Multiwriter Scenario by Exploiting the Similarity Graph of Document Terms. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp.475–480, 2017.

11. Jialuo Chen, **Pau Riba**, Alicia Fornés, Joan Mas, Josep Lladós and Joana Maria Pujadas-Mora. Word-Hunter: A Gamesourcing Experience to Validate the Transcription of Historical Manuscripts. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp.528–533, 2018.

12. **Pau Riba**, Andreas Fischer, Josep Lladós and Alicia Fornés. Learning Graph Distances with Message Passing Neural Networks. In *Proceedings of the International Conference on Pattern Recognition*, pp.2239-2244, 2018. (**Best Scientific Paper Award**)

13. Lei Kang, J.Ignacio Toledo, **Pau Riba**, Mauricio Villegas, Alicia Fornés and Marçal Rusiñol. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence model for Handwritten Word Recognition. In *Proceedings of the German Conference on Pattern Recognition*, pp.459–472, 2018.

14. Arnau Baró, **Pau Riba**, Jorge Calvo-Zaragoza and Alicia Fornés. Optical Music Recognition by Long Short-Term Memory Networks. In *Graphic Recognition. Current Trends and Evolutions*, Vol.11009, pp.81–95, 2018.

15. Sounak Dey*, **Pau Riba***, Anjan Dutta, Josep Lladós and Yi-Zhe Song. Doodle to Search: Practical Zero-Shot Sketch-Based Image Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.2179–2188, 2019.

16. **Pau Riba**, Anjan Dutta, Lutz Goldmann, Alicia Fornés, Oriol Ramos and Josep Lladós. Table Detection in Invoice Documents by Graph Neural Networks. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp.122–127, 2019.

17. Lei Kang, Marçal Rusiñol, Alicia Fornés, **Pau Riba** and Mauricio Villegas. Unsupervised Adaptation for Synthetic-to-Real Handwritten Word Recognition. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pp.3502–3511, 2020.

18. Lei Kang, **Pau Riba**, Marçal Rusiñol, Alicia Fornés and Mauricio Villegas. Distilling Content from Style for Handwritten Word Recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2020 (*Accepted*).

19. Lei Kang, **Pau Riba**, Yaxing Wang, Marçal Rusiñol, Alicia Fornés and Mauricio Villegas. GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. In *Proceedings of the European Conference on Computer Vision*, 2020 (*Accepted*).

20. Lei Kang, **Pau Riba**, Marçal Rusiñol, Alicia Fornés and Mauricio Villegas. Pay Attention to What You Read: Non-recurrent Handwritten Text Recognition. In *arXiv preprint arXiv:2005.13044*, 2020.

Tutorials

1. GMPRDIA: Graph-based Methods in Pattern Recognition & Document Image Analysis. In the *International Conference on Document Analysis and Recognition*, on September 21, 2019.

Leaded R&D Projects

1. Digitus II: Automatic Indexation Service for Handwritten Archives. Funded by *AGAUR Llavor*, from 21/07/2017 to 21/01/2018. co-PI.

Contributed open source

1. **Kornia**: Open Source Differentiable Computer Vision Library for PyTorch. Contributted in the Data Augmentation module. Code: `https://github.com/arraiyopensource/kornia`.

Bibliography

[1] Edward H. Adelson, Charles H. Anderson, James R. Bergen, Peter J. Burt, and Joan M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160:781–793, 2004.

[3] Narendra Ahuja and Sinisa Todorovic. From region based image representation to object discovery and recognition. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 6218, pages 1–19, 2010.

[4] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 510–517, 2012.

[5] Jon Almazán, Alicia Fornés, and Ernest Valveny. Deformable hog-based shape descriptor. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1022–1026, 2013.

[6] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, 2014.

[7] Hélio Almeida, Dorgival Guedes, Wagner Meira, and Mohammed J. Zaki. Is there a best quality metric for graph clusters? In *Joint European conference on machine learning and knowledge discovery in databases*, pages 44–59, 2011.

[8] Mohammad Reza Ameri, Michael Stauffer, Kaspar Riesen, Tien D Bui, and Andreas Fischer. Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance. *Pattern Recognition Letters*, 121:61–67, 2019.

[9] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569, 2019.

[10] Bengt Aspvall and Richard E Stone. Khachiyan's linear programming algorithm. *Journal of algorithms*, 1(1):1–13, 1980.

[11] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[12] Furqan Aziz, Richard C. Wilson, and Edwin R. Hancock. Backtrackless walks on a graph. *IEEE transactions on neural networks and learning systems*, 24(6):977–989, 2013.

[13] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015.

[15] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. SimGNN: A neural network approach to fast graph similarity computation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019.

[16] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[17] Pierre Baldi and Yves Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418, 1993.

[18] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, pages 119.1–119.11, 2016.

[19] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[20] Serge Belongie, Charless Fowlkes, Fan Chung, and Jitendra Malik. Spectral partitioning with indefinite kernels using the nyström extension. In *Proceedings of the European Conference on Computer Vision*, pages 531–542, 2002.

[21] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[22] Karsten M. Borgwardt. *Graph Kernels*. PhD thesis, Ludwig-Maximilians-Universität München, 2007.

[23] Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the IEEE International Conference on Data Mining*, pages 74–81, 2005.

[24] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21:i47–i56, 2005.

[25] Ehsan Zare Borzeshi, Massimo Piccardi, Kaspar Riesen, and Horst Bunke. Discriminative prototype selection methods for graph embedding. *Pattern Recognition*, 46(6):1648–1657, 2013.

[26] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[27] Klaus Broelemann, Anjan Dutta, Xiaoyi Jiang, and Josep Lladós. Hierarchical graph representation for symbol spotting in graphical document images. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 529–538, 2012.

[28] Klaus Broelemann, Anjan Dutta, Xiaoyi Jiang, and Josep Lladós. Hierarchical plausibility-graphs for symbol spotting in graphical documents. In *Graphics Recognition. Current Trends and Challenges*, pages 25–37, 2014.

[29] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[30] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[31] Luc Brun and Walter Kropatsch. Contains and inside relationships within combinatorial pyramids. *Pattern Recognition*, 39(4):515–526, 2006.

[32] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *Proceedings of the International Conference on Learning Representations*, 2014.

[33] Quang Anh Bui, Muriel Visani, and Remy Mullot. Unsupervised word spotting using a graph representation based on invariants. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 616–620, 2015.

[34] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[35] Horst Bunke and Kaspar Riesen. Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*, 44(9):1928–1940, 2010.

[36] Horst Bunke and Kaspar Riesen. Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057–1067, 2011.

[37] Terrence Caelli and Serhiy Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004.

[38] Hongyun Cai, Vincent W. Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[39] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision*, volume 6314, pages 778–792, 2010.

[40] Francesca Cesarini, Simone Marinai, L. Sarti, and Giovanni Soda. Trainable table location in document images. In *Proceedings of the International Conference on Pattern Recognition*, pages 236–240, 2002.

[41] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology*, 2(3):27, 2011.

[42] Ushasi Chaudhuri, Biplab Banerjee, and Avik Bhattacharya. Siamese graph convolutional network for content based remote sensing image retrieval. *Computer Vision and Image Understanding*, 184:22–30, 2019.

[43] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[44] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. FG-index: Towards verification-free query processing on graph databases. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 857–872, 2007.

[45] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[46] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[47] Stéphane Clinchant, Hervé Déjean, Jean-Luc Meunier, Eva Maria Lang, and Florian Kleber. Comparing machine learning approaches for table recognition in historical register books. In *International Workshop on Document Analysis Systems*, 2018.

[48] Francesc Comellas and Juan Paz-Sánchez. Reconstruction of networks from their betweenness centrality. In *Proceedings of the Workshops on Applications of Evolutionary Computation*, pages 31–37, 2008.

[49] Donatello Conte, Pasquale Foggia, Jean-Michel Jolion, and Mario Vento. A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions. *Pattern Recognition*, 39(4):562–572, 2006.

[50] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[51] Hani Daher, Djamel Gaceb, Véronique Eglin, Stéphane Bres, and Nicole Vincent. Ancient handwritings decomposition into graphemes and codebook generation based on graph coloring. In *International Conference on Frontiers in Handwriting Recognition*, pages 119–124, 2010.

[52] Nicholas Dahm, Horst Bunke, Terry Caelli, and Yongsheng Gao. A unified framework for strengthening topological node features and its application to subgraph isomorphism detection. In *International Workshop on Graph-Based Representation in Pattern Recognition*, volume 7877, pages 11–20, 2013.

[53] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

[54] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[55] Mathieu Delalandre, Tony Pridmore, Ernest Valveny, Hervé Locteau, and Eric Trupin. Building synthetic graphical documents for performance evaluation. In *Graphics Recognition. Recent Advances and New Opportunities*, pages 288–298, 2008.

[56] Aline Deruyver, Yann Hodé, Eric Leammer, and Jean-Michel Jolion. Adaptive pyramid and semantic graph: Knowledge driven segmentation. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 213–222, 2005.

[57] Sounak Dey, Anguelos Nicolaou, Josep Lladós, and Umapada Pal. Evaluation of word spotting under improper segmentation scenario. *International Journal on Document Analysis and Recognition*, 22(4):361–374, 2019.

[58] Sounak Dey, Pau Riba, Anjan Dutta, Josep Lladós, and Yi-Zhe Song. Doodle to search: Practical zero-shot sketch-based image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2179–2188, 2019.

[59] Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[60] François-Xavier Dupé and Luc Brun. Edition within a graph kernel framework for shape recognition. In *International Workshop on Graph-Based Representation in Pattern Recognition*, volume 5534, pages 11–20, 2009.

[61] François-Xavier Dupé and Luc Brun. Hierarchical bag of paths for kernel based shape classification. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 227–236, 2010.

[62] Anjan Dutta. *Inexact Subgraph Matching Applied to Symbol Spotting in Graphical Documents*. PhD thesis, Universitat Autònoma de Barcelona,, 2014.

[63] Anjan Dutta, Josep Lladós, Horst Bunke, and Umapada Pal. A product graph based method for dual subgraph matching applied to symbol spotting. In *Graphics Recognition. Current Trends and Challenges*, pages 11–24, 2014.

[64] Anjan Dutta, Josep Llados, and Umapada Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, 46(3):752–768, 2013.

[65] Anjan Dutta, Pau Riba, Josep Lladós, and Alicia Fornés. Pyramidal stochastic graphlet embedding for document pattern classification. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 33–38, 2017.

[66] Anjan Dutta and Hichem Sahbi. Stochastic graphlet embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 30(8):2369–2382, 2019.

[67] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[68] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[69] David W. Eggert, Kevin W. Bowyer, Charles R. Dyer, Henrik I. Christensen, and Dmitry B. Goldgof. The scale space aspect graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1114–1130, 1993.

[70] Ismail Elezi, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, and Laura Leal-Taixe. The group loss for deep metric learning. *arXiv preprint arXiv:1912.00385*, 2019.

[71] Sergio Escalera, Alicia Fornés, Oriol Pujol, Petia Radeva, Gemma Sánchez, and Josep Lladós. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30(15):1424–1433, 2009.

[72] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

[73] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[74] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 524–531, 2005.

[75] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

[76] David Fernández-Mota, Jon Almazán, Núria Cirera, Alicia Fornés, and Josep Lladós. BH2M: The barcelona historical, handwritten marriages database. In *Proceedings of the International Conference on Pattern Recognition*, pages 256–261, 2014.

[77] David Fernández-Mota, Pau Riba, Alicia Fornés, and Josep Lladós. On the influence of key point encoding for handwritten word spotting. In *International Conference on Frontiers in Handwriting Recognition*, pages 476–481, 2014.

[78] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. HMM-based word spotting in handwritten documents using subword models. In *Proceedings of the International Conference on Pattern Recognition*, pages 3416–3419, 2010.

[79] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7):934–942, 2012.

[80] Andreas Fischer, Kaspar Riesen, and Horst Bunke. Graph similarity features for HMM-based handwriting recognition in historical documents. In *International Conference on Frontiers in Handwriting Recognition*, pages 253–258, 2010.

[81] Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. A fast matching algorithm for graph-based handwriting recognition. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 194–203, 2013.

[82] Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. Approximation of graph edit distance based on Hausdorff matching. *Pattern Recognition*, 48(2):331–343, 2015.

[83] Andreas Fischer, Seiichi Uchida, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. Improving Hausdorff edit distance using structural node context. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 148–157, 2015.

[84] Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[85] Pasquale Foggia, Gennaro Percannella, and Mario Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(1):1–40, 2014.

[86] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[87] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. ICDAR 2019 competition on table detection and recognition (cTDaR). In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1510–1515, 2019.

[88] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.

[89] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[90] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[91] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. On context-dependent clustering of bandits. In *Proceedings of the International Conference on Machine Learning*, pages 1253–1262, 2017.

[92] Nabil Ghanmi and Abdel Belaid. Table detection in handwritten chemistry documents using conditional random fields. In *International Conference on Frontiers in Handwriting Recognition*, pages 146–151, 2014.

[93] Jaume Gibert, Ernest Valveny, and Horst Bunke. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, 45(9):3072–3083, 2012.

[94] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. Table detection using deep learning. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 1, pages 771–776, 2017.

[95] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning*, pages 1263–1272, 2017.

[96] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[97] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[98] Romain Goffe, Luc Brun, and Guillaume Damiand. Tiled top-down pyramids and segmentation of large histological images. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 255–264, 2011.

[99] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[100] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[101] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 729–734, 2005.

[102] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[103] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[104] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 991–995, 2015.

[105] Christopher G. Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, volume 15, pages 147–5244, 1988.

[106] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[107] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[108] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167, 2004.

[109] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the ACM symposium on Theory of computing*, pages 604–613, 1998.

[110] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456, 2015.

[111] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3668–3678, 2015.

[112] Jean-Michel Jolion. Stochastic pyramid revisited. *Pattern Recognition Letters*, 24(8):1035–1042, 2003.

[113] Jean-Michel Jolion and Azriel Rosenfeld. *A pyramid framework for early vision: multiresolutional computer vision*, volume 251. Springer Science & Business Media, 2012.

[114] Salim Jouili and Salvatore Tabbone. Graph embedding using constant shift embedding. In *Proceedings of the International Conference on Pattern Recognition*, pages 83–92, 2010.

[115] Lei Kang, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture. *Pattern Recognition*, 2020. Submitted.

[116] Lei Kang, J. Ignacio Toledo, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. In *Proceedings of the German Conference on Pattern Recognition*, pages 459–472, 2018.

[117] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Kernels for graphs. *Kernel methods in computational biology*, 39(1):101–113, 2004.

[118] Isaak Kavasidis, Sergio Palazzo, Concetto Spampinato, Carmelo Pino, Daniela Giordano, Danilo Giuffrida, and Paolo Messina. A saliency-based convolutional neural network for table and chart detection in digitized documents. *arXiv preprint arXiv:1804.06236*, 2018.

[119] Thomas Kieninger and Andreas Dengel. Table recognition and labeling using intrinsic layout features. In *International Conference on Advances in Pattern Recognition*, pages 307–316, 1999.

[120] Thomas Kieninger and Andreas Dengel. Applying the T-Recs table recognition system to the business letter domain. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 518–522, 2001.

[121] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[122] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[123] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representations*, 2017.

[124] Elvis Koci, Maik Thiele, Wolfgang Lehner, and Romero Oscar. Table recognition in spreadsheets via a graph representation. In *International Workshop on Document Analysis Systems*, pages 139–144, 2018.

[125] Risi Kondor and Karsten M. Borgwardt. The skew spectrum of graphs. In *Proceedings of the International Conference on Machine Learning*, pages 496–503, 2008.

[126] Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. In *Advances in Neural Information Processing Systems*, pages 2982–2990, 2016.

[127] Risi Kondor, Nino Shervashidze, and Karsten M. Borgwardt. The graphlet spectrum. In *Proceedings of the International Conference on Machine Learning*, pages 529–536, 2009.

[128] Nathan Korda, Balázs Szörényi, and Shuai Li. Distributed clustering of linear bandits in peer to peer networks. In *Proceedings of the International Conference on Machine Learning*, volume 48, pages 1301–1309, 2016.

[129] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[130] John Lafferty and Guy Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

[131] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

[132] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[133] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam. Graph edit distance: a new binary linear programming formulation. *arXiv preprint arXiv:1505.05740*, 2015.

[134] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2548–2555, 2011.

[135] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, 2006.

[136] Shuai Li, Wei Chen, and Kwong-Sak Leung. Improved algorithm on online clustering of bandits. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[137] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 539–548, 2016.

[138] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Proceedings of the International Conference on Machine Learning*, pages 3835–3845, 2019.

[139] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations*, 2016.

[140] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3D deep learning. In *Advances in Neural Information Processing Systems*, pages 963–973, 2019.

[141] Mario Livio. *The golden ratio: The story of phi, the world's most astonishing number*. Broadway Books, 2008.

[142] Josep Lladós, Marçal Rusiñol, Alicia Fornés, David Fernández, and Anjan Dutta. On the influence of word representations for handwritten word spotting in historical documents. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(05), 2012.

[143] Arnaud Lods, Eric Anquetil, and Sébastien Macé. Fuzzy visibility graph for structural analysis of online handwritten mathematical expressions. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 641–646, 2019.

[144] D. Lohani, A. Belaïd, and Y. Belaïd. An invoice reading system using a graph convolutional network. In *Proceedings of the Asian Conference on Computer Vision Workshops*, 2018.

[145] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[146] David G. Lowe. *Perceptual Organization and Visual Recognition*, volume 5. Springer Science & Business Media, 2012.

[147] Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, and Thierry Brouard. Fuzzy multilevel graph embedding. *Pattern Recognition*, 46(2):551–565, 2013.

[148] Mahshad Mahdavi, Michael Condon, Kenny Davila, and Richard Zanibbi. LPGA: Line-of-sight parsing with graph-based attention for math formula recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 647–654, 2019.

[149] Raghavan Manmatha, Chengfeng Ha, and Edward M. Riseman. Word spotting: a new approach to indexing handwriting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 631–637, 1996.

[150] Rebeca Marfil, Luis Molina-Tanco, Antonio Bandera, Juan Antonio Rodríguez, and Francisco Sandoval. Pyramid segmentation algorithms revisited. *Pattern Recognition*, 39(8):1430–1451, 2006.

[151] Rebeca Marfil, Luis Molina-Tanco, Antonio Bandera, and Francisco Sandoval. The construction of bounded irregular pyramids with a union-find decimation process. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 307–318, 2007.

[152] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics*, 36(4):71, 2017.

[153] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[154] Kurt Mehlhorn. *Graph algorithms and NP-completeness*. Springer-Verlag New York, Inc., 1984.

[155] Bruno T. Messmer and Horst Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, 1999.

[156] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[157] Harry L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[158] Seyedeh Fatemeh Mousavi, Mehran Safayani, Abdolreza Mirzaei, and Hoda Bahonar. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recognition*, 61:245–254, 2017.

[159] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[160] Sameer A. Nene, Shree K. Nayar, Hiroshi Murase, et al. Columbia object image library (COIL-100). Technical report, Department of Computer Science, Columbia University, 1996.

[161] Michel Neuhaus and Horst Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific, 2007.

[162] Mark E. J. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[163] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the International Conference on Machine Learning*, pages 2014–2023, 2016.

[164] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[165] Shubham Singh Paliwal, D. Vishwanath, Rohit Rahul, Monika Sharma, and Lovekesh Vig. TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 128–133, 2019.

[166] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[167] Elzbieta Pekalska and Robert P. W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations And Applications*. World Scientific, 2005.

[168] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[169] Ioannis Pratikakis, Konstantinos Zagoris, Basilis Gatos, Joan Puigcerver, Alejandro H. Toselli, and Enrique Vidal. ICFHR2016 handwritten keyword spotting competition (H-KWS 2016). In *International Conference on Frontiers in Handwriting Recognition*, pages 613–618, 2016.

[170] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[171] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait. Rethinking table recognition using graph neural networks. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 142–147, 2019.

[172] Sheikh Faisal Rashid, Abdullah Akmal, Muhammad Adnan, Ali Adnan Aslam, and Andreas Dengel. Table recognition in heterogeneous documents using machine learning. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 1, pages 777–782, 2017.

[173] Toni M. Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–II, 2003.

[174] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[175] Pau Riba, Anjan Dutta, Lutz Goldmann, Alicia Fornés, Oriol Ramos, and Josep Lladós. Table detection in invoice documents by graph neural networks. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 122–127, 2019.

[176] Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning graph distances with message passing neural networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 2239–2244, 2018.

[177] Pau Riba, Alicia Fornés, and Josep Lladós. Handwritten word spotting by inexact matching of grapheme graphs. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 781–785, 2015.

[178] Pau Riba, Josep Lladós, and Alicia Fornés. Error-tolerant coarse-to-fine matching model for hierarchical graphs. In *International Workshop on Graph-Based Representation in Pattern Recognition*, pages 107–117, 2017.

[179] Pau Riba, Josep Lladós, Alicia Fornés, and Anjan Dutta. Large-scale graph indexing using binary embeddings of node contexts for information spotting in document image databases. *Pattern Recognition Letters*, 87:203–211, 2017.

[180] Kaspar Riesen and Horst Bunke. IAM graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, 2008.

[181] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

[182] Kaspar Riesen and Horst Bunke. Graph classification by means of lipschitz embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1472–1483, 2009.

[183] Kaspar Riesen and Horst Bunke. *Graph classification and clustering based on vector space embedding*, volume 77. World Scientific, 2010.

[184] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. Bipartite graph matching for computing the edit distance of graphs. In *International Workshop on Graph-Based Representation in Pattern Recognition*, volume 4538, pages 1–12, 2007.

[185] Antonio Robles-Kelly and Edwin R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):365–378, 2005.

[186] Antonio Robles-Kelly and Edwin R. Hancock. A Riemannian approach to graph embedding. *Pattern Recognition*, 40(3):1042–1056, 2007.

[187] Paul L. Rosin and Geoff A. W. West. Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114, 1989.

[188] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011.

[189] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545–555, 2015.

[190] Marçal Rusiñol and Josep Lladós. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *International Journal on Document Analysis and Recognition*, 12(2):83–96, 2009.

[191] Marçal Rusiñol, Josep Lladós, and Gemma Sánchez. Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Analysis and Applications*, 13(3):321–331, 2010.

[192] A. Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

[193] Eric Saund. A graph lattice approach to maintaining and learning dense collections of subgraphs as image features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2323–2339, 2013.

[194] Kenneth M. Sayre. Machine recognition of handwritten words: A project report. *Pattern Recognition*, 5(3):213–228, 1973.

[195] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[196] Lambert Schomaker and Marius Bulacu. Automatic writer identification using connected-component contours and edge-based features of uppercase western script. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):787–798, 2004.

[197] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. DeepDeSRT: Deep learning for detection and structure recognition of tables in document images. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[198] Faisal Shafait and Ray Smith. Table detection in heterogeneous documents. In *International Workshop on Document Analysis Systems*, pages 65–72, 2010.

[199] Dennis Shasha, Jason T. L. Wang, and Rosalba Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the Symposium on Principles of Database Systems*, pages 39–52, 2002.

[200] Yuming Shen, Li Liu, Fumin Shen, and Ling Shao. Zero-shot sketch-image hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3598–3607, 2018.

[201] Nino Shervashidze and Karsten M. Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems*, pages 1660–1668, 2009.

[202] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[203] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 488–495, 2009.

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

[205] Edward Smith, Scott Fujimoto, Adriana Romero, and David Meger. Geometrics: Exploiting geometric structure for graph-encoded objects. In *Proceedings of the International Conference on Machine Learning*, pages 5866–5876, 2019.

[206] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158, 2003.

[207] Michael Stauffer, Andreas Fischer, and Kaspar Riesen. A novel graph database for handwritten word images. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 553–563, 2016.

[208] Michael Stauffer, Andreas Fischer, and Kaspar Riesen. Keyword spotting in historical handwritten documents based on graph matching. *Pattern Recognition*, 81:240–253, 2018.

[209] Sebastian Sudholt and Gernot A. Fink. PHOCNet: A deep convolutional neural network for word spotting in handwritten documents. In *International Conference on Frontiers in Handwriting Recognition*, pages 277–282, 2016.

[210] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[211] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*, pages 1067–1077, 2015.

[212] Michael J. Tarr. The object databank, 2011.

[213] Chris Tensmeyer, Vlad I. Morariu, Brian Price, Scott Cohen, and Tony Martinez. Deep splitting and merging for table structure decomposition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 114–121, 2019.

[214] Hannu Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, 19(10):1183–1193, 2003.

[215] Markus Ulrich, Christian Wiedemann, and Carsten Steger. Combining scale-space and similarity-based aspect graphs for fast 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1902–1914, 2012.

[216] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[217] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *Proceedings of the International Conference on Learning Representations*, 2017.

[218] Mario Vento. A long trip in the charming world of graphs for pattern recognition. *Pattern Recognition*, 48(2):291–301, 2015.

[219] Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pages 87–97, 2017.

[220] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I, 2001.

[221] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[222] Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[223] Peng Wang, Veronique Eglin, Christophe Garcia, Christine Largeron, Josep Lladós, and Alicia Fornes. A novel learning-free word spotting approach based on graph representation. In *International Workshop on Document Analysis Systems*, pages 207–211, 2014.

[224] Peng Wang, Véronique Eglin, Cristophe Garcia, Christine Largeron, Josep Lladós, and Alicia Fornés. A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance. In *Proceedings of the International Conference on Pattern Recognition*, pages 3074–3079, 2014.

[225] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3056–3065, 2019.

[226] Yalin Wang, Ihsin T. Phillipst, and Robert Haralick. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 528–532, 2001.

[227] Chris Watkins. Kernels from matching operations. Technical report, Department of Computer Science, university of London, 1999.

[228] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[229] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J. Weinberger. Inequalities for the l1 deviation of the empirical distribution. Technical report, HP Labs, Palo Alto, 2003.

[230] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pages 639–655. Springer, 2012.

[231] Tomas Wilkinson and Anders Brun. Semantic and verbatim word spotting using deep neural networks. In *International Conference on Frontiers in Handwriting Recognition*, pages 307–312, 2016.

[232] Richard C. Wilson, Edwin R. Hancock, and Bin Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1112–1124, 2005.

[233] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[234] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5419, 2017.

[235] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pages 2048–2057, 2015.

[236] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

[237] Xifeng Yan, Philip S. Yu, and Jiawei Han. Graph indexing: a frequent structure-based approach. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 335–346, 2004.

[238] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[239] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for scene graph generation. In *Proceedings of the European Conference on Computer Vision*, pages 670–685, 2018.

[240] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the International Conference on Machine Learning*, pages 40–48, 2016.

[241] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

[242] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[243] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[244] Jiawei Zhang. Graph neural distance metric learning with graph-bert. *arXiv preprint arXiv:2002.03427*, 2020.

[245] Feng Zhou and Fernando de la Torre. Factorized graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1774–1789, 2015.

[246] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[247] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pages 14747–14756, 2019.

This work has been partially supported by the FPU fellowship FPU15 / 06264 from the Spanish Ministerio de Educación, Cultura y Deporte. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.