University of Insubria
Department of Theoretical and Applied Science (DiSTA)
PhD Thesis in Computer Science
XXIX Cycle of Study

Document Image Classification Combining Textual and Visual Features

Candidate:
LUCIA NOCE

Thesis Advisor:
Prof. IGNAZIO GALLO

December, 2016

To my Father

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Ignazio Gallo and my colleague Alessandro Zamberletti for the continuous help during my research activities, the joy and enthusiasm they have for research was contagious and motivational for me, even during tough times of my Ph.D course.

I gratefully acknowledge Prof. Marco La Cascia (Univeristà degli studi di Parlermo) and Prof. Umberto Castellani (Università degli studi di Verona) for reviewing my thesis giving me precious and priceless comments.

Sincere thanks go to Nicola Lamberti, 7pixel and Gruppo MutuiOnline that made my Ph.D. work possible, providing important cues and ideas as well as data, funds and places for our activities.

I would also like to thank present and past members of the Applied Recognition Technology Laboratory (ArteLab) and to all my co-workers in 7pixel Varese, they all have been a source of support and friendships.

Last but not least, I would express a deep sense of gratitude to my dearest father Sergio, thank you for being my support, my guide and my strength.

Lucia Noce
Varese, 23 November 2016

1

Introduction and Motivations

This thesis addresses to the problem of classifying document images by jointly exploiting both visual and textual features.

Traditionally, transmission and storage of information have been made by paper documents. In the past few decades, documents increasingly originated on the computer, however, in spite of this, it is unclear whether the computer has decreased or increased the amount of paper. Documents are still printed out for reading, dissemination, and markup [1]. The objective of the major of institutions that deals with a large amount of documents is to be able to cope with electronic and paper documents in an efficient and integrated way. Because of this all the paper documents are digitalized, and the result of this step produces the so called document images [1]. The whole process of document image analysis aims to recognize the text and graphics components in images, and to extract the intended information as a human would [2].

Document image classification represents an important step in the document image analysis, classifying and grouping documents into known categories is often a prerequisite step for the subsequent phases in several applications (e.g., document retrieval and information extraction). There are many examples of use and need of document classification [3]. Document classification allows the automatic archiving or distribution of documents, in this way for example business letters can be classified in “offer”, “order” or “inquiry”, and be dispatched to the correct department for processing [4]. In Digital Library, document classification is used to improve the indexing efficiency: classification of documents into table of contents pages or title pages can suggest from which pages to

extract meta-data useful for indexing [5]. In Document image retrieval, classification of image documents based on visual similarity can improve retrieval efficiency and accuracy, helping users to retrieve articles from one specific journal or documents that contains table or graphics for examples [6]. Document classification facilitates higher-level document analysis. In order to obtain high accuracy, most of the available information extraction systems are casting on a specif type of document; a first document classification is necessary to select the set of suitable documents for the appropriate document analysis system [7].

Document classification problem is generally tackled in two ways: a document page can be classified exploiting its content, or exploiting its visual aspects.

In the first case, a document image is transformed into a text document and the classification is performed only relying on text; this kind of classification is called content-based. Generally an Optical Character Recognition (OCR) algorithm is applied on images and the result represents text documents to be classified. Text documents may contain plain text or a some extra meta-data usually represented with tags. This kind of classification is based on textual features (such as word frequency or word histogram) or on structural information known from tags. Although such methods can be efficiently applied on specif simple and well-made artificial documents, they present many limitations. First of all such methodologies are limited to documents where OCR can reach good results, moreover they cannot be used to classify documents that do not contain any textual information, and they ignore important visual aspects that may help in the classification decision.

On the other hand, a single page document can be classified exploiting visual features generally extracted from the document layout. Document structure-based classification approaches are focused on finding features able to describe the visual representation of a document. The choice of the typology of information extracted from images permits to differentiate among different approaches. Layout based features are often used achieving interesting results, but this kind of methods are limited to specif class of documents that presents a fixed visual structure. Adopting the same method to new types of document may involve a manual tuning of the chosen visual features. Moreover when documents do not present a constrained structure as in handwritten documents, fixed features may not be able to capture all the necessary characteristic of a class [8].

The success achieved by the Convolutional Neural Networks (CNN) in Computer Vision represents the turning point even in this context. Starting from the observation that document images usually present hierarchical patterns which are often repeated in different parts of the document, so that the layout can be learnt as a combination of small group of middle or lower level features, Convolutional Neural Network were applied to document image classification stating the state-of-the-art.

I have collected and discussed most of the interesting and relevant related works

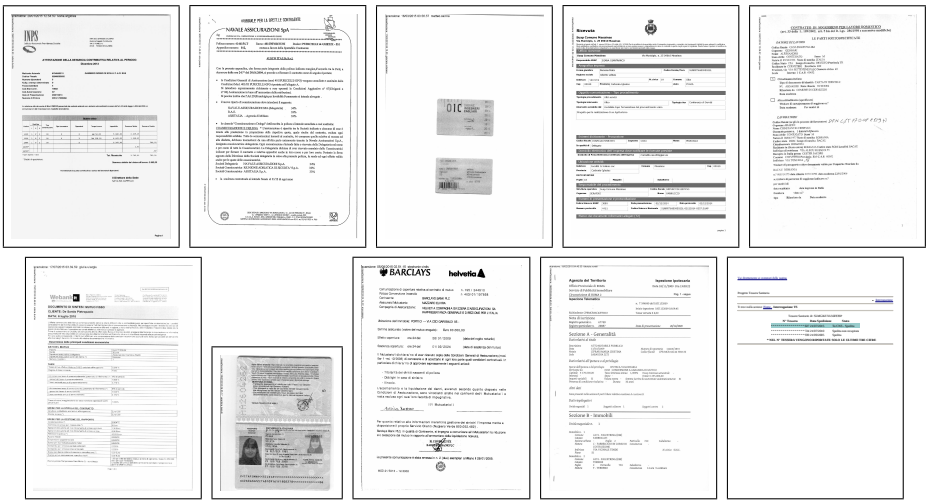


Figure 1.1: Examples of image documents collected from costumers and used in the core business of a company. Each document belongs to a specif class, and moreover images do not present same sizes and resolutions.

recently presented in literature in Chapter 3, starting from standard approaches and moving to very recent applications that exploit CNNs to perform the document classification.

Document image classification is not a trivial problem also because images often present problems due to image's quality: some documents may be cut, crumpled, torn, stained before the digitization step or images may be acquired using low resolutions for example. The document image quality is not the only issue for a classification purpose. Other difficulties come from the fact that documents belonging to the same class may present different layout and at the same time, documents extracted from more classes may appear similar, considering only the visual aspect.

Figure 1.1 presents several examples of document images that are collected from costumers and used in the core business of a company, and empathize the limitations of document images classification based on text or layout analysis. Each image belongs to a different class, and although for some of them classification may be easily performed relying for example only on visual aspect (e.g. passport), for others, the task is not trivial. Images presented different format and resolutions, some documents may be degraded and some may show a layout which is difficult to extract, and in these cases OCR results are hard to obtain.

In this thesis I present the final work on document image classification I carried out as a result of my three years of PhD study. Before proposing a work able to use techniques coming from two separated areas, it has been mandatory to explore state-of-the-art

methods for both image and textual classification. For this very reason, during my PhD career I carried out different activities dealing exclusively on text classification [9, 10, 11, 12]; then I moved to image classification [13, 14, 15, 16, 17, 18, 19]; and lastly I proposed a novel method that combine the information extracted from texts and images at the same time [20].

In Chapter 2 I collect a description of the most relevant preliminary cited works, focusing on the adopted methodologies and the achieved results, while in Chapter 4 I report all the details of the final work that exploits the knowledge acquired in the previous activities. The construction of the methods pipeline was done in collaboration with my colleague PhD Alessandro Zamberletti and my advisor Prof. Ignazio Gallo, therefore when presenting the method and discussing about the achieved results I will use the first-person plural to reflect the joint nature of this work.

The main addition of our work is the introduction of a preprocessing step embedding additional textual information into the processed document images. To do so, we combine Optical Character Recognition and Natural Language Processing algorithms to extract and manipulate relevant text concepts from document images. Such textual information is then visually embedded within each document image to improve the classification results of a Convolutional Neural Network. The experimental phase, reported in Chapter 5 proves that the overall document classification accuracy of a Convolutional Neural Network trained using these text-augmented document images is considerably higher than the one achieved by a similar model trained solely on classic document images, especially when different classes of documents share similar visual characteristics. Moreover we developed a comparison among our method and content-based methods, stating that our methodology outperforms the results achieved not only using solely visual features but also exploiting only textual features.

Although this thesis is about document image classification, the idea of using textual and visual features is not restricted to this context and comes from the observation that textual and visual information are complementary and synergetic in many aspects. News articles, blogs and social media are only few examples where images and text are used together to represent information, and from this comes the inspirations of the jointly exploitation of textual and visual features.

2
Preliminary Works

In this Chapter I reported the most salient preliminary works. Exploring both text and image analysis has been mandatory to achieve the final goal of proposing a method able to jointly use both the two techniques. In this Chapter I described explored methods dealing exclusively on text analysis. All these works were done in collaboration with my colleagues of ArteLab [1], the research laboratory where I worked during these three years. Therefore when I describe all these methods I will use the first-person plural to reflect this collaboration.

I choose reported two works. In particular, Section 2.1 contains an overview of our paper "Automatic Prediction of Future Business Conditions" [9] that I presented at the International Conference on Natural Language Processing(PolTAL 2014) and that was done in collaboration with the Digital Data Streams Laboratory (DDSLab) of the University of Pavia. Section 2.2 reports the works "Query and Product Suggestion for Price Comparison Search Engines Based on Query-product Click-through Bipartite Graphs" [12] that I presented at the International Conference on Web Information Systems and Technologies (WEBIST 2015).

2.1 Automatic Prediction of Future Business Conditions

We presented a novel method for automatically extracting forward-looking statement from a specific type of formal corporate documents called earning call transcripts. Our main objective is that of improving an analyst's ability to accurately forecast future

[1] http://artelab.dicom.uninsubria.it

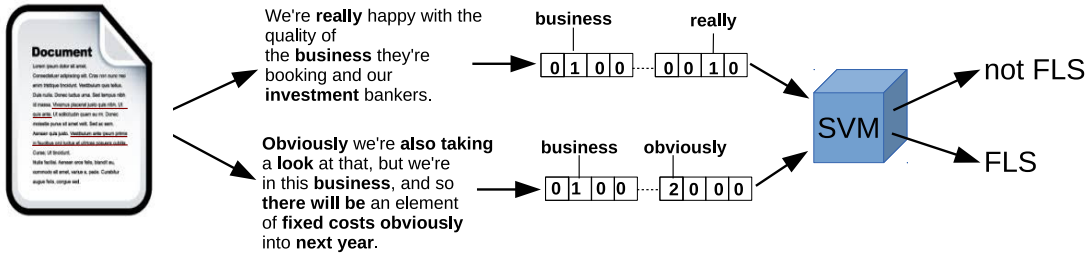


Figure 2.1: An example showing how the feature vectors provided as input to the supervised classifier are generated. From left to right, the sentences extracted from the starting earning call documents are given a vectorial representation using the final weighted dictionary $\mathcal{D}$; those feature vectors are finally provided as input to the SVM classifier that assigns them a label denoting whether they represent forward-looking statements or not.

events of economic relevance, over and above the predictive contribution of quantitative firm data that companies are required to produce. By exploiting both Natural Language Processing and Machine Learning techniques, our approach is stronger and more reliable than the ones commonly used in literature and it is able to accurately classify forward-looking statements without requiring any user interaction nor extensive tuning. Formally, a forward-looking statement is defined as a short sentence that contains information likely to have, or reft to, a direct effect in the foreseeable future (e.g., plans, predictions, forecasting, expectations and intentions). Recent research shows that, by analyzing the information included in these future-looking statements it is possible to improve the analysts' ability to accurately forecast future earnings, over and above the predictive contribution of quantitative firm data and consensus analysts forecasts.

In this work, the task of identifying and extracting forward-looking statements from earning call documents has been approached as a classical text classification problem, in which each sentence is assigned one or more labels from a finite set of predefined categories. More formally, given a finite non-empty set of documents $\mathcal{X} = \{x_0, \ldots, x_n\}$ and a set of categories $\Omega = \{\omega_1, ..., \omega_c\}$, the proposed task requires to assign to every pair $(x_i, y_j) \in \mathcal{X} \times \Omega$ a boolean label yes/no. Since it is difficult and ineffective for a supervised classifier to interpret natural text as is, each document $x_i \in \mathcal{X}$ is usually represented as a vector $\phi(x_i)$ in which each element measures the number of times that a given term in x_i, contained into a finite dictionary of terms $\mathcal{D}$, appears in x_i. More in detail, given a document $x_i \in \mathcal{X}$ and a dictionary of terms $\mathcal{D} = \{t_1, ..., t_d\}$, the document is represented as $\phi(x_i) = (x_i^1, \ldots, x_i^d)$, where each $x_i^j \in \phi(x_i)$ measures the frequency of occurrence of the term $t_j \in \mathcal{D}$ in the document $x_i \in \mathcal{X}$.

The main objective of this work is to extract forward-looking statements from earning call transcripts, for this reason, each earning call is split into a finite set of sentences

Table 2.1: Comparison between the results achieved by the proposed method and some other different approaches on the FLS dataset , in terms of Overall Accuracy (OA), precision (p), recall (r) and F-measure(f).

method	OA (%)	p (%)	r (%)	f (%)
Bozanic *et al.* [23]	35.56	20.05	78.21	31.91
BOW	84.93	69.37	48.94	57.39
PoS	81.03	69.10	48.07	56.69
Proposed	87.57	74.82	53.23	62.21

that were previously tagged by different experts as either forward-looking (FLS) or not (NFLS), for this reason we have that: $\Omega = \{\text{FLS}, \text{NFLS}\}$.

As summarized in Figure 4.2, once the starting set of earning call documents has been split into the set of sentences $\mathcal{X}$, we proceed to further split those sentences into single terms and use them to build a primitive dictionary $\mathcal{D}'$, as in classic Bag of Words (BOW) approaches. Instead of representing each sentence as a vector using the whole dictionary $\mathcal{D}'$, we prune it to obtain the final dictionary $\mathcal{D}$ by assigning a weight to each of its element $t_i \in \mathcal{D}'$, using the weighting formula proposed by Peñas *et al.* [21], with the main goal of detecting a small subset of terms that are characteristics of forward-looking sentences.

Once the relevance of every term $t_i \in \mathcal{D}'$ has been computed using Equation 4.1, it is possible to obtain the final smaller dictionary $\mathcal{D}$ simply by removing all those terms whose relevance value is lower than a threshold ψ, as follows:

$$\mathcal{D} = \{t_i \in \mathcal{D}' \mid \mathit{Relevance}(t_i, sc, gc) > \psi\} \tag{2.1}$$

As shown in Figure 2.1, given a sentence $x_i \in \mathcal{X}$ extracted from an earning call document, we compute its vectorial representation $\phi(x_i)$ as previously described, using the final dictionary $\mathcal{D}$. This vectorial representation is provided as input to a supervised machine learning algorithm with the goal of assigning one among the two possible categories in Ω to x_i. We decided to employ a Support Vector Machine (SVM) classifier [22]; this choice is motivated by the fact that the same model was used in most of the previous related works, as it can lead to optimal results with minimal tuning effort.

In summary, the proposed model is expected to take as input a sentence and to return a label that represents whether that sentence is a forward-looking statement or not.

We proved that the proposed future-looking statements identification method is able to reach and overcome other state-of-the-art approaches for the given classification task. Many experiments were conducted on a dataset we built and called "FLS dataset"; I will

not report the whole experimental phase in this thesis but all the details can be found in the original paper [9].

2.2 Query and Product Suggestion for Price Comparison Search Engines Based on Query-product Click-through Bipartite Graphs

Query suggestion is a technique for generating alternative queries to facilitate information seeking, and has become a needful feature that commercial search engines provide to web users. We focused on query suggestion for price comparison search engines. In this specific domain, suggestions provided to web users need to be properly generated taking into account whether both the searched and the suggested products are still available for sale. To this end, we propose a novel approach based on a slightly variant of classical query-URL graphs: the query-product click through bipartite graph. Such graph is built using information extracted both from search engine logs and specific domain features such as categories and products' popularities. Information collected from the query-product graph can be used to suggest not only related queries but also related products.

Query suggestion in price comparison websites is very similar to the query suggestion in e-commerce websites, because in both scenarios users search for products they want to buy and therefore it is reasonable to say that they issue to the two different search engine the same type of queries. Following the discussion of Hasan *et al.* [24] about e-commerce websites, the proposed method is a click-through-based query suggestion approach.

Our model exploits the two main entities of the website for which it has been designed for [25]: products and categories. These concepts are not restricted to our context but are used in many other major e-commerce and search engine websites.

More in details, the result of a generic query in our context is a list of commercial products sold by several retailers and sorted by descending price. Each item of the list is an offer. All the offers belong to a specific category, according to the type of the objects searched by the user. Offers may also be linked to products. Products represent aggregations of items/offers, and all the offers inherent to a specific item are collected under the related product.

In general, a click-through-based query suggestion algorithm is composed of two phases: (i) an offline module which manages the data and builds the model, (ii) and an online procedure which supplies the query suggestions. Our solution extracts information from web server logs. Starting from those logs, the offline phase of our method can be summarized in the following steps:

- Data Extraction: is the pre-processing phase, the web server log is parsed. Queries and related information are extracted, cleaned and normalized;
- Session creation: the sessions are created as associations between queries and clicked products;
- Building phase: creation of the click-through bipartite graph by exploiting the notion of product;

On the other hand, the subsequent online phase gathers query suggestions from the model and returns a fixed number of ranked related queries and products.

In Figure 2.2, both the offline and the online phases of the proposed approach are visually summarized. In the following sections all their major details are reported.

Data Extraction

The first step consists of the extraction of data from the web server log. Those logs contains rough data that has to be cleaned before being used for query suggestion.

Different cleaning stages are performed to gather only log lines containing the information we want to exploit. First we performed bot filtering to retrieve only log search lines that lead to a user click. Within a log line we extract only the fields of interest: the textual query, the id of the category for which the query has been performed, and the clicked product. Synonym mapping for queries like "mtb" and "mountain bike" and spelling correction for cases such as "dylandog" corrects into "dylan dog" are also performed. We also use stemming to detect equivalent queries, *e.g.* "woman bike" is equivalent to "women bikes", "samsung galaxy S6" represents the same query as "S6 samsung galaxy". Since the website is in italian, all the reported examples have been translated for better understanding.

Session creation

For building the model, we take into account only textual queries extracted from web server logs. A query is the list of terms that a user types to search for the item he needs, the same query may be issued to the search engine several times, each submission creates a query session.

We followed the approach that has been proposed by Wen *et al.* [26] and used by Baeza-Yates *et al.* [27] which considers a simple notion of query session that is composed of a query and the URLs clicked within its query results. In our method instead of URLs we collected the products returned by the query, as follows:

$$\text{QuerySession} = (\text{searchedText}, \text{clickedProducts})$$

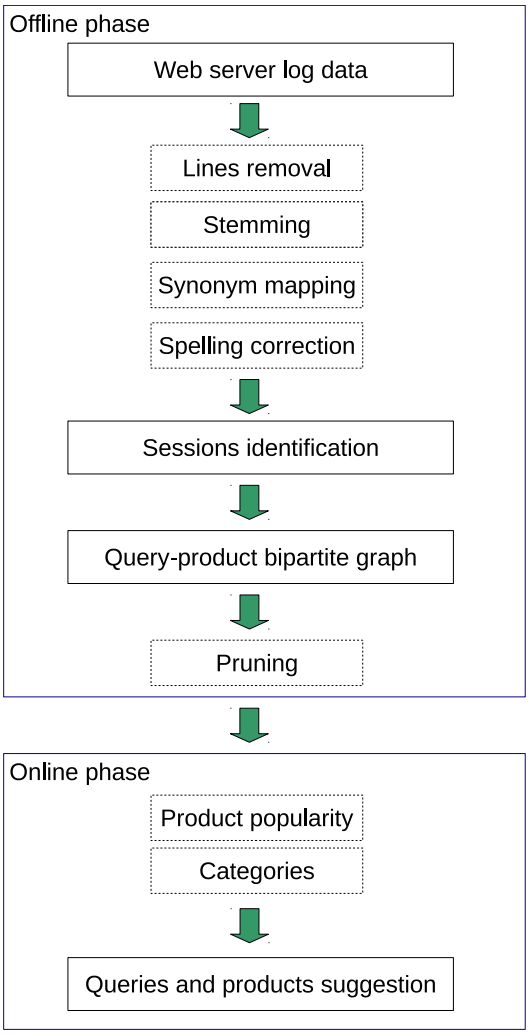


Figure 2.2: Visual summarization of the proposed pipeline. From top to bottom: starting from raw log data, all the information needed to determine query sessions are extracted; the query-product bipartite graph is built and suggestions are provided by exploiting both the product categories and the product popularities extra-fields.

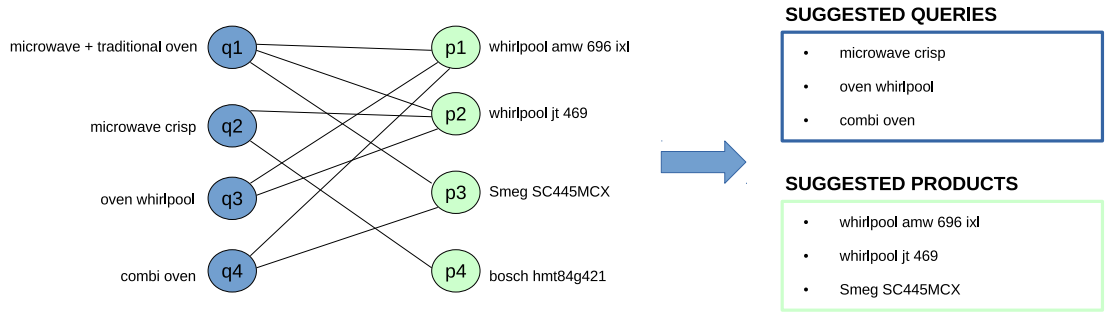


Figure 2.3: This visual example shows the usage of the query-product bipartite graph. Some details, such as the popularity score and the number of clicks, are omitted to make the image more readable. Assuming that the input query is equal to q_1 and that N (number of suggestions) is equal to 3, we determine p_1, p_2 and p_3 as the 3 most popular products related to the input query q_1, and we gather from them the three most clicked related queries q_2, q_3 and q_4.

The reason why we adopt a different notion of query session lies our application context. In price comparison engines, each time a query is submitted, a list of related offers is shown. However, each offer may be available for a limited amount of time, therefore it would be senseless to gather the direct offers URLs which may have been already destroyed. Creating query suggestions dealing to unavailable offers is improper and unattractive for users, and may cause a loss of clients for the website.

The taxonomy of our price comparison website allows us to exploit the notion of product. Not every offer is related to a product, but statistical studies made by the website administrator shows that the majority (97%) of the offers dealing to a user click are related to a product. Starting from this assertion we consider only offers related to products and sessions storing products related to the submitted queries. This not only allows us to provide a more reliable query suggestion system, but also permits to supply product suggestions to users.

Building phase

We build the proposed model using the notion of query session previously described. This way we create a different version of the well-known query-URL bipartite graph in which, instead of URLs, we use clicked products.

A query-product bipartite graph consists of two disjoint sets of nodes corresponding to queries and products respectively. Intuitively, in a click-through graph vertexes on one side correspond to queries, while vertexes on the other side represents products. An edge connecting a query q to a product p exists whenever p was clicked starting from q.

A visual example of such query-product bipartite graph is given in Figure 2.3. The

set of nodes in the left part are queries and the ones in the right part are products. An edge between a query q and an URL u indicates a user click of u when issuing q (for simplicity click numbers are omitted from the graph).

The query-product bipartite graph gathers a large amount of potential information that can be used for query suggestion, query clustering, query reformulation and so on.

In our method we consider two kind of graphs depending on whether a user uses a category when searching for a product or not. In our price comparison website, users are allowed to submit queries within a specified category, and they expect to receive product results from that same category. Showing query suggestions outside the selected category may be annoying for the users; as such so build two different kind of graphs. The first one does not exploit the concept of category and therefore it can be used to provide query suggestion results from all the website categories. On the other hand, query suggestion results from the second graph are restricted to a specific category selected by the user. By query suggesion results we intend both queries and products.

A final pruning phase is necessary to avoid the suggestion of queries that may have a negative impact on the user experience. We start removing products that are only related to one query, and then we remove queries that are only connected to one product.

Both the two versions of the query-product bipartite graph were tested and evaluated both in terms of time needed to complete their building phase, and in term of accuracy for the provided query suggestion results.

Online query suggestion

The online phase of our method exploits two basic concepts related to the taxonomy of the price comparison engine we used in this work: categories and popularity. As previously described, for query suggestion, we take into account only offers that belong to a product.

For each product the website administrator has provided us a numeric value which indicates the popularity of an offer. This value is computed considering both the whole server traffic and the available offers in a certain period of time, and it is periodically updated.

Given a user query q, if it belongs to the query set of the bipartite graph, we automatically obtain the list of the related queries and of the related products. The retrieved product list is then sorted by ascending popularity value, and the top k products are collected. Starting from this list of top k products, another list of related queries is produced and sorted by the number of clicks stored in the product-query bipartite graph edges. The top k retrieved queries are selected and provided as results.

On the other hand if a query q does not belong to the query-product bipartite graph we use the similarity measure described by Zanon *et al.* [28] to retrieve the more similar query q' among the bipartite graph. This measure exploits the concept of categories

and the Jaccard similarity coefficient proposed by Tan *et al.* [29]. Once q' is found the system acts as previously described.

Experiments and Results

We conducted several experiments to measure the effectiveness of our proposed query and product suggestion method. Both query and product suggestions were evaluated with and without using category information. All the datails of the experimenthal phase are reported in the original paper [12].

From our experimental evaluation results we can conclude that our model proposes reliable suggestions on the top-8 result not only using the "categorized" dataset (75% query, 71% product), but also using the "uncategorized" dataset (72% query, 68% product). When referring to top-5 results precisions are around 80% for both datasets. Among categories, according to the coverage measure, the higher results are achieved for the more populated class.

Due to the similarity of the context of our work and the one from Zanon *et al.* [28] and despite the fact that we used a different dataset, we propose a weak but but still significant comparison between our results and those achieved by their model. Unlike all the other works in literature, Zanon *et al.* [28] evaluate their query suggestion model on two price comparison engines, reaching a coverage rate equals to 37% on the top-8 suggestion and an overall quality equals to 70%.

Our model overcomes those results, achieving significantly higher coverage rate: 75% on the "uncategorized" dataset and 56% on the "categorized" dataset. The quality of our query/product suggestion method is also higher than theirs for top-8 results. This confirms that a click-through based model better fits query suggestions task in price comparison and more in general in e-commerce website, as asserted in [24].

3
Related Works and Background

In this Chapter I reviewed the existing methodologies related to this thesis and some fundamental concepts useful to understand our document image classification proposal presented in Chapter 4.

In particular in Section 3.1, I reported most of the previous approaches that deal with document classification starting from the methods that extract visual or textual features and ending with recent works that employed Convolutional Neural Networks (CNN) in their pipelines. Analyzing pros and cons of related works I'm giving an explanation of our choices, motivating the idea of embedding textual features in document images.

Since CNNs are exploited in our method, in Section 3.2 I will introduce the basic concepts of such networks, and in Section 3.3 I analyze their recent application on Natural Language Processing problems. The different datasets used during the evaluation phase are described in Section 3.4.

3.1 Related works

Existing approaches for document image classification and retrieval differ from each other based both on the type of extracted information (textual or visual) and/or the type of image analysis that is performed over the processed documents (global or local). Different supervised and unsupervised models have been proposed in literature throughout the last decade [3]: Random Forest based [30], Decision Tree and Self-organizing Map based [31], K-Nearest Neighbor based [32], Hidden Markov Model based [33], and Graph matching based [34] to name a few.

Features extracted from document images can either be visual, textual, or a combi-

nation of the two. The percentage of text and non-text elements in a content region of image, font sizes, column structures, document structure, bag-of-words, and statistics of features are only a few examples of extracted combined textual and visual characteristics adopted by some of the previously cited works for solving the task of document image classification [35, 30, 3].

In literature, visual-based local document image analysis was investigated and adopted for document images classification [36, 37]. Region-based algorithms reach interesting results when applied to structured model, such as letters or forms. Classifying a document based on its whole visual content is also possible [31]. However, all of these cited visual feature based approaches have limitations, such as the manual definition of document templates or specific geometric configurations of fixed features related to different document layouts.

Content-based document classification has also been extensively studied in literature. Content-based analysis of documents is typically performed relying on text extracted using OCR methods, although text allows retrieving information about document content, visual layout plays an equal important role and it's used to detect some image region where applies OCR in order to have a more accurate extraction of content elements [38]. Nonetheless, OCR is prone to errors and is not always applicable to all kind of documents *e.g.* handwriting text is still difficult to read and those document images must have high resolution.

The recent success of Convolutional Neural Networks (CNN) in Computer Vision research areas [39, 40, 41] inspired novel applications of those algorithms to other domains such as document image analysis, text categorization and text understanding.

A recent work by Kang *et al.* [8] shows that CNN are able to learn the entire supervised document image classification process, from feature extraction to final classification. Authors propose the use of a CNN for document image classification: a CNN is trained to recognize the class of given subsampled and pixel value normalized document images. Authors test their model on several challenging datasets, showing that such approach outperforms all previously explored methodologies.

Following the same intuition, Harley *et al.* [42] achieve outstanding results, setting new state-of-the-art results for image document image classification and retrieval. An extensive evaluation of their CNN model is reported, determining that features extracted from deep CNN exceed the performance of all alternative visual and textual features both on document image classification and retrieval by a large margin. They also investigate about transfer learning, asserting that features learnt using CNN trained on object recognition are also effective in describing documents. Moreover, authors present several experiments varying between a single holistic CNN and ensembles of region-based CNN, exploring different initialization strategies. The performances of the evaluated models are evaluated on 2 different subsets of the IIT CDIP Test Collection [43], the smallest

one [44] coincide with the one used by Kang *et al.* [8], while the largest one is composed by a significantly larger set of documents [42]. Evaluation against all the different CNN configuration and diverse bag-of-words approaches are reported.

CNN are also employed in text categorization and text understanding fields. The text obtained by applying OCR to a document image can be viewed as a document itself. In this manner, document image classification can be reconducted to a sentence-level classification task.

Several interesting works that use CNN for Natural language processing have been recently published. Kim [45] propose a simple CNN with one Convolutional Layer and prove that it performs well when applied to a wide range of difficult text classification tasks such as sentiment analysis and question classification. CNN are not the only Deep Model that proved to be effective for document classification tasks; Zhang *et al.* [46] use other Deep Learning algorithms for several text understanding tasks using Temporal Convolutional Networks [47] (ConvNets), working with both English and Chinese languages. Another recent work by Johnson *et al.* [48] studies CNN for text categorization exploiting word order to lead to more accurate predictions.

Our work exploits CNN in the same manner as in the approaches proposed by Harley *et al.* [42] and Kang *et al.* [8], our aim is to combine both content-based and image analysis, and to do so we embed content information extracted from text into subsampled document images and feed those visually enriched documents to a properly trained CNN model. A comparison between most of the previously cited approaches, provided in Chapter 5, demonstrate that our model can easily reach comparable results. We also compare our results applying CNN only to text extracted from document images following the same approach of Kim *et al.* [45]. Results show that even though content-based approach leads to accurate results, our proposal that uses both images and text information performs even better. This demonstrates that the combination of the two different genre of features (visual and content) allows Machine Learning models to reach higher accuracy values in difficult document image classification tasks especially in the case of high visual similarity between different classes of documents.

3.2 Convolutional Neural Network

In the last few years, deep neural networks have led to breakthrough results on a variety of pattern recognition problems, such as computer vision and voice recognition [49]. One of the essential components leading to these results has been a special kind of neural network called a Convolutional Neural Network.

CNNs take a biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. This idea was expanded upon by a fascinating experiment by Hubel and Wiesel [50] in 1962 where

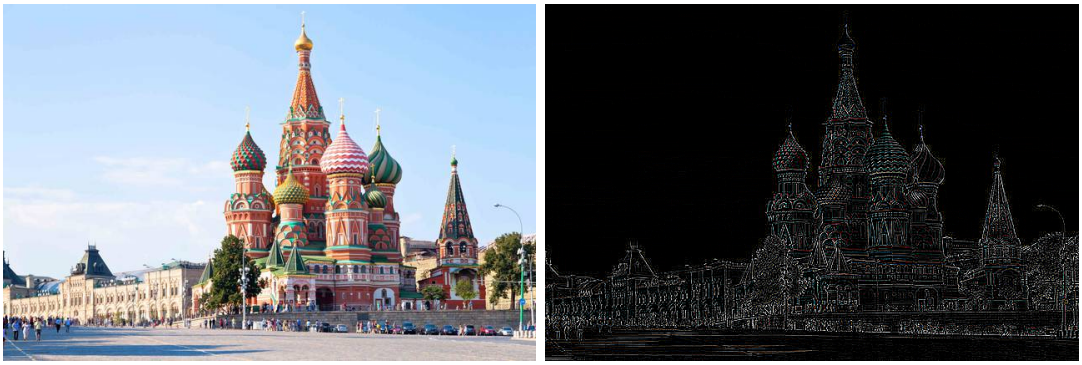


Figure 3.1: Convolution Filter: the image on the right side is obtained by convolving each channel of the image by the 3×3 kernel reported in equation 3.1, over the image on the left side. After this operation edges are detected.

they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation. For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges. Hubel and Wiesel [50] found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

In Machine Learning, these regions of cells are referred to as a receptive fields. Receptive Fields are implemented in the form of weighted matrices, referred to as kernels; which, similarly to their biologically inspired counterparts, are sensitive to similar local subregions of an image. The degree of similarity between a subregion of an image and a kernel may be computed simply convolving the subregion using the given kernel.

For example, a simple 3×3 kernel like:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.1}$$

may be convolved over one given image to detect and inform about the presence of edges as in Figure 3.2.

In a traditional feed-forward neural network each input neuron is connected to each output neuron in the next layer. That is also called a fully connected layer, or affine layer. In CNNs instead, convolutions over the input layer are used to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies to different filters, typically hundreds

or thousands like the ones showed above, and combines their results. Pooling layers, typically applied after the convolutional layers. Pooling layers subsample their input.

During the training phase, a CNN automatically learns the values of its filters based on the task you want to perform. For example, in Image Classification a CNN may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to determine higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features.

This architecture permits to implement two important aspects: location invariance and compositionality. Location invariance allows to classify whether or not there is a determined object in an image for example. Filters are slided over the whole image, and because of this, the position of the object to be detected is not important. Applying pooling also invariance to translation, rotation and scaling are performed. The second key aspect is (local) compositionality. Each filter composes a local patch of lower-level features into higher-level representation. From simple features like edges, gradients and blurs, kernels start to learn more complex features that are highly discriminative for the processed images. Figure 3.2, provided by [51], reports example images from all layers of a network similar to the famous AlexNet from Krizehvsky *et al.* [49] pointing out information learned by kernels.

Basically CNNs are composed of a series of several layers of convolutions with non-linear activation functions, pooling, and fully connected layers. As previously described CNNs adopt convolution operator. Given an image x with k channels, and a kernel w, the convolution operator generates another image y in the following way:

$$y_{i'j'k'} = \sum_{ijk} w_{ijkk'} x_{i+i',j+j',k} \tag{3.2}$$

where k' corresponds to the number of filters/kernels in the convolution.

A linear filter is followed by a non-linear activating function applied identically to each component of a feature map. The most used non-linear activating function function is the Rectified Linear Unit (ReLU), but other functions can be used. ReLu is defined as follow:

$$y_{ijk} = \max\{0, x_{ijk}\}. \tag{3.3}$$

An other key aspect of Convolutional Neural Networks are pooling layers. Pooling layers subsample their input through a pooling operators which operates on individual feature channels, merging nearby feature values into one. The most common way to do pooling it to apply a max operation (max-pooling) to the result of each filter. Max-pooling is defined as:

$$y_{ijk} = max\{y_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p\} \tag{3.4}$$

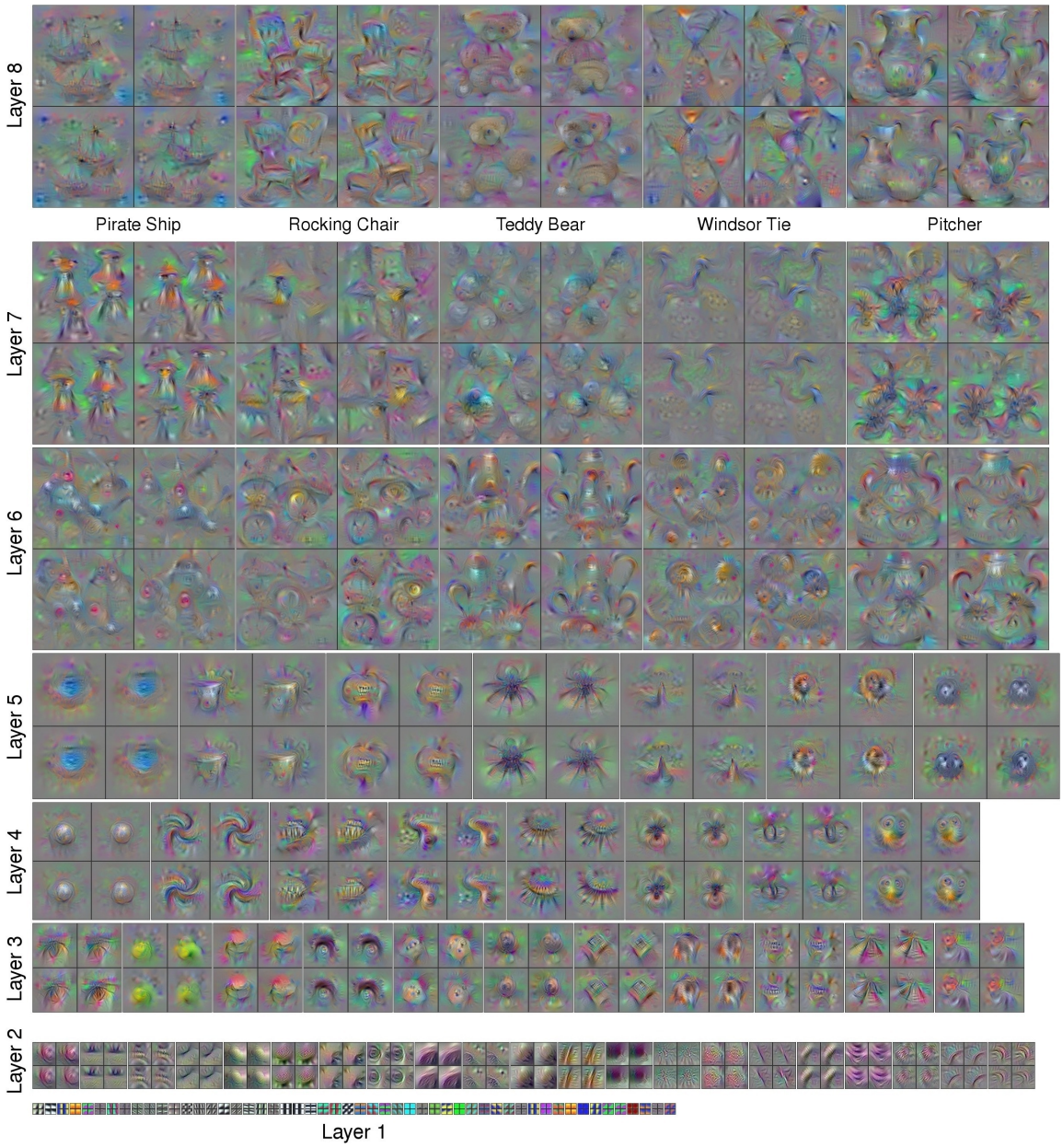


Figure 3.2: Visualization of example features of all eight layers of a CNN. The images reflect the true sizes of the features at different layers. For each feature in each layer, visualizations from 4 random gradient descent runs are shown. One can recognize important features at different scales, such as edges, corners, wheels, eyes, shoulders, faces, handles, bottles, etc. Complexity increases in higher-layer features as they combine simpler features from lower layers. The variation of patterns also increases in higher layers, revealing that increasingly invariant, abstract representations are learned.

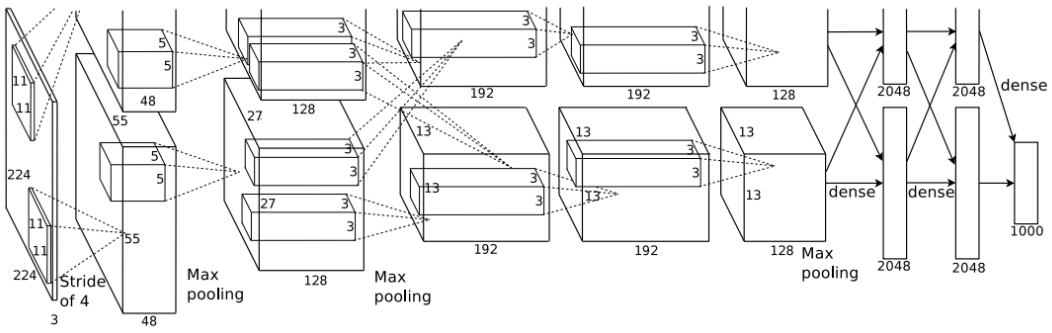


Figure 3.3: An illustration of the architecture of the CNN proposed by Alex Krizhevsky taken form the original paper. AlexNet consists of five convolution layers, followed by three dense layers.

Pooling reduces the output dimensionality but keeps the most salient information, and also it provides a fixed size output matrix, which typically is required for classification. In imagine recognition, pooling also provides basic invariance to translating (shifting) and rotation. When pooling is performed over a region, the output will stay approximately the same even if the image is shifted or rotated by a few pixels, because the max operations will pick out the same value regardless.

The success of Convolutional Neural Networks in Computer Vision started few years ago with the model proposed by Alex Krizhevsky et al. [49]: "AlexNet". This network was trained on the ILSVRC 2012 training data, which contained 1.2 million training images belonging to 1000 classes and brought down the error rate on that task by half, beating traditional hand-engineered approaches.

As summarized in Figure 3.3, the net is composed of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final softmax. Author employed dropout to reduce overfitting in the fully-connected layers.

Dropout is a powerful regularization method introduced by Hinton *et al.* [52], which has been shown to work well for large neural networks. The key idea of dropout is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much.

The performance of AlexNet motivated a number of CNN-based approaches, all aimed at a performance improvement over and above that of AlexNet's. Just as AlexNet was the winner for ILSVRC challenge in 2012, a CNN-based net Overfeat by Sermanet *et al.* [53] was the top-performer at ILSVRC-2013. Their key insight was that training a convolutional network to simultaneously classify, locate, and detect objects in images can boost the classification accuracy and the detection and localization accuracy of all tasks.

GoogleNet by Szegedy *et al.* [54], won the ILSVRC-2014, established that very deep networks can reach high accuracies in classification performance. Authors introduced a trivial 1×1 convolutional layer after a regular convolutional layer, this not only reduced the number of parameters but also results in CNNs with more expressive power.

Better details are reported in the work of Szegedy *et al.* [55] where the authors show that having one or more 1×1 convolutional layers is similar to have a multi-layer perceptron network processing the outputs of the convolutional layer that precedes it. Another trick that the authors utilize is to involve inner layers of the network in the computation of the objective function instead of the typical final softmax layer (as in AlexNet). The authors attribute scale invariance as the reason behind this design decision.

VGG-19 and its variants by Simonyan and Zisserman [56] is another example of a high-performing CNN. An interesting feature of VGG design is that it forgoes larger sized convolutional filters for stacks of smaller sized filters. These smaller sized filters tend to be chosen so that they contain approximately the same number of parameters as the larger filters they supposedly replace. This design decision provided efficiency and regularization-like effect on parameters due to the smaller size of the filters involved.

For the work presented in this thesis following the results achieved by Harley *et al.* [42] for image document classification, we adopted AlexNet.

3.3 Convolutional Neural Network for Natural Language Processing

Inspired by the great success in Computer Vision, Convolutional Neural Network were also applied to Natural Language Processing (NLP) problems reaching very interesting results. In NLP tasks the input of a CNN is composed of sentences or documents represented as a matrix. Each row of the matrix corresponds to a vector that represents one token, typically a word, but it could be a character. Usually, these vectors are word embeddings (low-dimensional representations) like word2vec [57] or GloVe [58], but they could also be one-hot vectors that index the word into a vocabulary.

In NLP filters slide over full rows of the matrix, instead of sliding over local patches of an image, as in computer vision. Thus the width of filters is usually the same as the width of the input matrix, and the height, or region size, may vary. Figure 3.4, summarizes the structure of a Convolutional Neural Network designed for NLP tasks.

CNNs are used in classifications tasks, such as Sentiment Analysis, Spam Detection or Topic Categorization.

Yoon Kim [45] evaluates a CNN architecture on various classification datasets, built for Sentiment Analysis and Topic Categorization tasks. The CNN architecture achieves very good performance across datasets, and new state-of-the-art on a few. The network

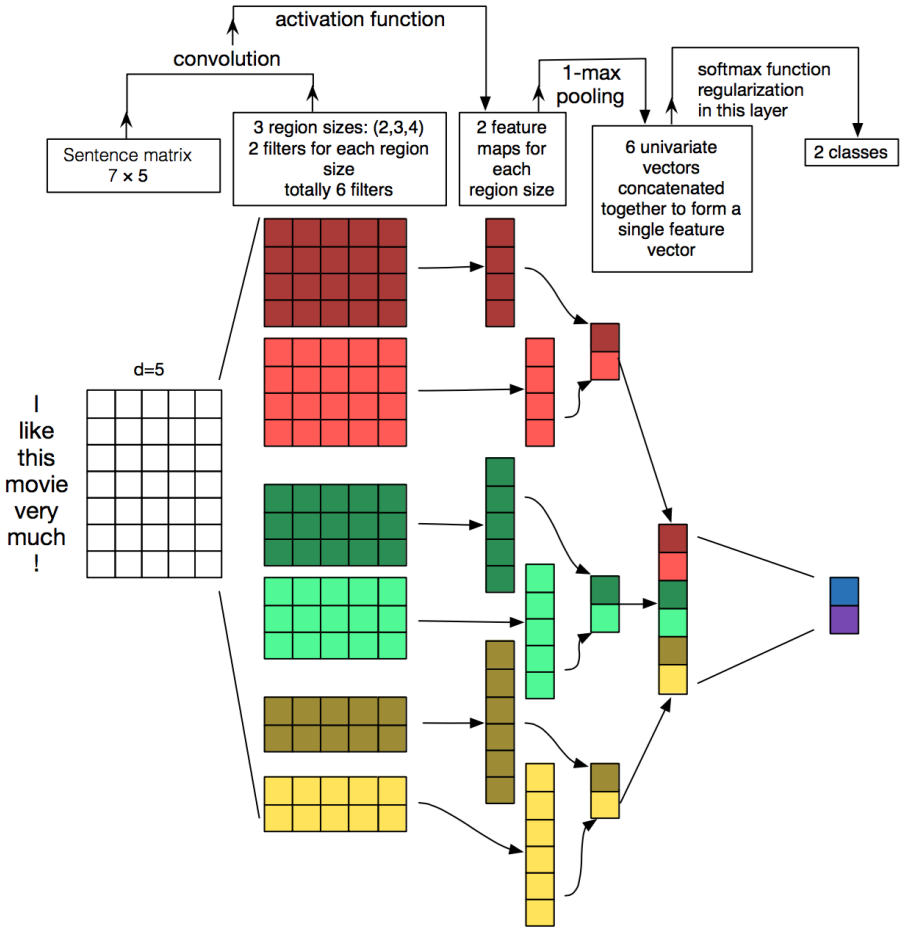


Figure 3.4: Illustration of a Convolutional Neural Network architecture for sentence classification. Here we depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states. Source: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

used in this paper is quite simple: the input layer is a sentence composed of concatenated word2vec word embeddings, followed by a convolutional layer with multiple filters, then a max-pooling layer, and finally a softmax classifier. The paper also experiments with two different channels in the form of static and dynamic word embeddings, where one channel is adjusted during training and the other is not. A similar, but somewhat more complex, architecture was previously proposed in [59]. Wang et.al[60]Adds an additional layer that performs "semantic clustering" to this network architecture.

Johnson and Zhang [61] train a CNN from scratch, without the need of pre-trained word vectors like word2vec or GloVe. It applies convolutions directly to one-hot vectors. The author also proposes a space-efficient bag-of-words-like representation for the input data, reducing the number of parameters the network needs to learn. In [62] the author extends the model with an additional unsupervised "region embedding" that is learned using a CNN predicting the context of text regions. The approach in these papers seems to work well for long-form texts (like movie reviews), but their performance on short texts (like tweets) is not clear. Intuitively, it makes sense that using pre-trained word embeddings for short texts would yield larger gains than using them for long texts.

Building a CNN architecture involves the choice of hyperparameters, such as: Input represenations (word2vec, GloVe, one-hot), number and sizes of convolution filters, pooling strategies (max, average), and activation functions (ReLU, tanh). The work proposed in [63] performs an empirical evaluation on the effect of varying hyperparameters in CNN architectures, investigating their impact on performance and variance over multiple runs. Results stand out that max-pooling always beat average pooling, that the ideal filter sizes are important but task-dependent, and that regularization does not seem to make a big different in the NLP tasks.

CNNs for Relation Extraction and Relation Classification tasks are explored in [64]. In addition to the word vectors, the authors use the relative positions of words to the entities of interest as an input to the convolutional layer. This model assumes that the positions of the entities are given, and that each example input contains one relation. Similar models have been explored by [65] and [66].

Another interesting use case of CNNs in NLP can be found in [67] and [68]. These papers describe how to learn semantically meaningful representations of sentences that can be used for Information Retrieval. The example given in the papers includes recommending potentially interesting documents to users based on what they are currently reading. The sentence representations are trained based on search engine log data.

Weston *et al.* [69] present a CNN architecture to predict hashtags for Facebook posts, while at the same time generating meaningful embeddings for words and sentences. These learned embeddings are then successfully applied to recommend potentially interesting documents to users, trained based on click-stream data.

All the cited models were based on words, but CNNs have been also applied directly

to characters. The model described in [70] learns character-level embeddings, joins them with pre-trained word embeddings, and uses a CNN for Part of Speech tagging. Zhang *et al.* [71] explore the use of CNNs to learn directly from characters, without the need for any pre-trained embeddings. Notably, the authors use a relatively deep network with a total of 9 layers, and apply it to Sentiment Analysis and Text Categorization tasks. Results show that learning directly from character-level input works very well on large datasets (millions of examples), but underperforms simpler models on smaller datasets (hundreds of thousands of examples). Application of character-level convolutions to Language Modeling is explored in [72], using the output of the character-level CNN as the input to an LSTM at each time step. The same model is applied to various languages.

Essentially, all of the reported papers were published in the past 1-2 years, underlining the interest of exploiting Convolutional Neural Network not only for Computer Vision tasks but also for other fields such as Natural Language Processing. Because of this, while exploring existing approaches for document image classification, it has been mandatory to evaluate results achieved applying CNNs to the content-based document classification task.

3.4 Datasets

We test our approach on two datasets, and to better emphasize the effectiveness of our proposal in solving intra-class similarity issue, we use some representative subsets of the same document image collections.

The first dataset, the so-called *Loan* dataset, has been provided by an Italian loan comparison website company. Through their platforms, this company provides a service which let customers to quickly compare the best rates and terms of available loans. To supply the best obtainable loan, website owners need to collect all the necessary documents in a digital format.

We collect a set of 14 different classes that present similar visual styles, this means that a document belonging to a class is often indistinguishable using just visual style features. Documents are used to provide loan offers to customers. A total of 16250 document images are collected and divided into training, evaluation and testing sets following similar proportions as in the ImageNet dataset: 80% training, 10% evaluation and 10% testing. Examples of image documents extracted from the different classes are reported in Figure 3.5.

Among all the 14 collected classes, we select 2 subsets made up of 3 classes each that have very similar visual layouts and due to this fact is hard to classify documents correctly. Visually similar subset are used to better underline the effectiveness of our proposal.

The first subset is composed by the following classes: “Family Status”, “Marriage

Certificate" and "Residence Certificate", previously introduced in Figure 4.1 and 4.3; we name it *Certificates subset*. The second subset is called *Contracts subset* and it contains the following document classes: "Preliminary Purchase", "Loan Contract" and "Preliminary Report".

In order to compare our approach to existing approaches, we test our method on the same dataset used by Harley *et al.* [42] and Kang *et al.* [8], we refer to it as *Tobacco* dataset. The *Tobacco* dataset is composed of 3482 images divided in 10 classes: Advertisement, Email, Form, Letter, Memo, News, Note, Report, Resume, Scientific. Figure 3.6 reports examples of documents for each of the 10 classes. The split of Tobacco dataset is the same used in related works [42, 8, 43]: 800 images are used for training, 200 for validations, and the remainder for testing.

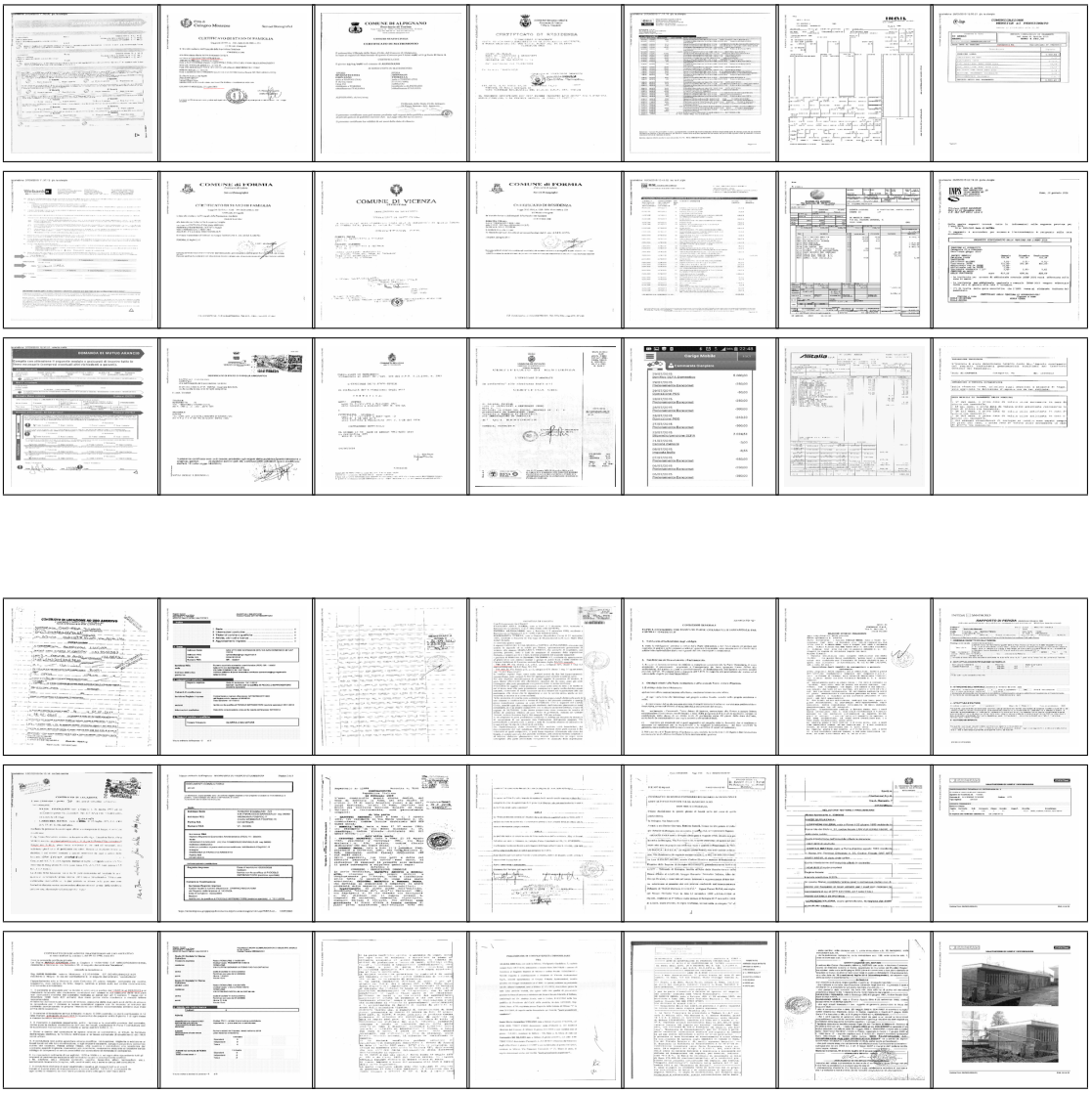


Figure 3.5: The Loan dataset: for every column, three representative documents of a specific class are displayed. From left to right classes are reported in the following order: “Loan Request”, “Family Status”, “Marriage Certificate”, “Residence Certificate”, “Account Balance”, “Payroll”, “Pension Payslip”, “Lease”, “Company Registration”, “Previous Contract”, “Preliminary Purchase”, “Loan Contract” ,“Preliminary Report” and “Expertise”. It can be noted that documents belonging to different classes present similar visual style.

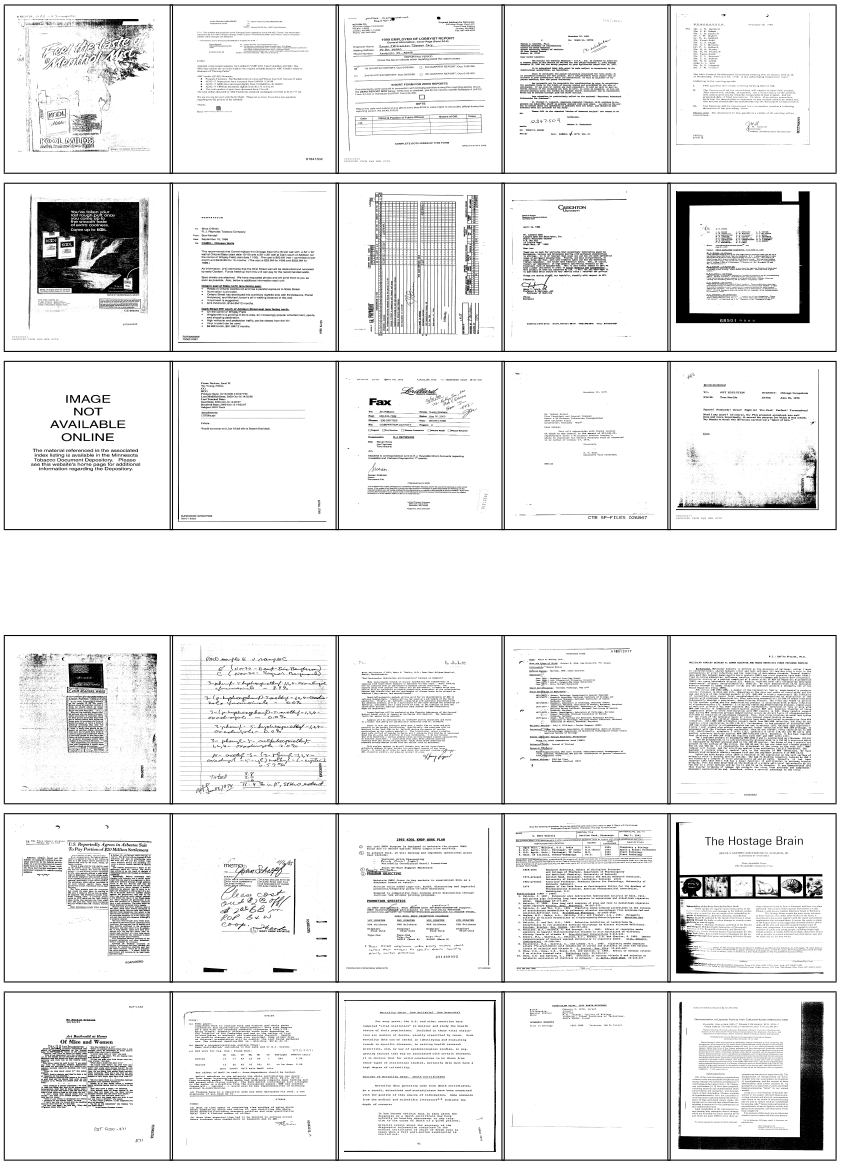


Figure 3.6: The Tobacco dataset: for every column, three representative documents of a specific class are displayed. From left to right classes are reported in the following order: “Advertisement”, “Email”, “Form’, “Letter”, “Memo”, “News”, “Note”, “Report”, “Resume”.

4

Image Document Classification

This chapter contains an overview of our article "Embedded Textual Content for Document Image Classification with Convolutional Neural Networks" [20] presented at the international ACM Symposium of document engineering (DocEng 2016).

4.1 Introduction

Document image classification and retrieval is an important task in document processing as it is a key element in a wide range of contexts, such as: automated archiving of documents, Digital Library constructions and other general purpose document image analysis applications [3].

Nowadays a large number of documents are produced, processed, transferred and stored as digital images everyday: forms, letters, printed articles and advertisement are only a few examples of them. While documents belonging to different macro-areas (*e.g.* financial, advertisement, *etc.*) typically show substantially different visual layouts from one to another and thus can be accurately classified just by comparing their visual characteristics; the same does not hold true for documents belonging to the same macro-area but different sub-areas (*e.g.* house ads, shop ads, *etc.*).

Given this situation, when the task is to perform a highly specific document classification among different document categories which are are both visually and semantically very similar, a combined content and visual analysis is mandatory to achieve satisfying fine-grained classification results. To this end, we propose a novel system to perform fine-grained document image classification exploiting both the content and the visual characteristics of the processed documents.

As introduced in Chapter 3, there are plenty of methods in literature that perform document classification relying exclusively on textual content extracted from the processed documents. While those algorithms can be effective for simple and well-made artificial documents, they have many limitations: (i) they ignore important visual document features (*e.g.* tables, images and diagrams) that may play an important role in the final document classification predictions, (ii) they are limited to printed documents due to Optical Character Recognition (OCR) limits, and (iii) they cannot be used to classify documents that do not contain any textual information or contain machine-unreadable text.

Accordingly, image analysis is complementary to content-based document classification for many classes of documents, and several techniques in literature successfully rely on structural and visual aspects to perform coarse-grained document classification [42].

For documents that present the same fixed structure, such as forms, template matching is adopted [37, 73]: for each class, a template is manually chosen and during the classification phase the input document is matched with one or more class-representative template. Various layout-based features are also adopted, and many works show their effectiveness for the document classification task [33, 74]. Following state-of-the-art results obtained in the Computer Vision research field, many recent works [42, 8] also perform document image classification using Deep Convolutional Neural Networks (CNN), obtaining outstanding results [42]. Starting from these state-of-the-art results and considering the previously described fine-grained document classification problem, we propose a novel method that combines textual and visual features using CNN.

More in details, our proposal consists of embedding content information extracted from text within document images, with the aim of adding elements which help the system in distinguish different classes that appear visually indistinguishable.

Our model was evaluated on two different dataset, and between different numbers and gender of document categories; the results from our experimental phase show that the proposed methodology achieves competitive results when compared to recent related works, and is able to effectively perform fine-grained document image classification.

4.2 Algorithm

The main idea behind this work comes out after considering that in document image classification, intra-class similarity represents an issue that can be solved adding textual information extracted from the processed document images.

Figure 4.1 shows some representative examples of intra-class similarity, if in our method, we had just focused on visual features, it would have been impossible to distinguish between the two classes of documents shown. However, by underlining significant textual information, fine-grained document classification becomes easier.

Figure 4.1: Documents belonging to three different representative classes are shown. In the first line examples of the class "Family Status" are shown, in the second line documents that belong to the class "Marriage Certificate" are reported, while in the third line documents were extracted from the "Residence Certificate" class. It's almost impossible to distinguish among the 3 classes only relying on the visual style of the documents.

With the currently available computational resources, the adoption of CNN only allows the use of small sized document images. Although the original document layouts are still distinguishable even in sub-sampled document images, text becomes unreadable. Our challenge consists of adding content information in a visual manner, to let the CNN model to exploit such information when performing the classification task to reach higher classification accuracies for documents having high intra-class similarity.

The pipeline of the building phase of proposed approach is shown in Figure 4.2, its three main phases can be summarized as follows:

- **Text extraction and analysis**: OCR is employed to extract textual information from original sized document images. Texts are analyzed and for each class a dictionary is built.
- **Embedding phase**: exploiting word position coordinates and dictionaries, relevant words are emphasized within each sub-sampled document image.
- **Training phase**: a CNN is trained using sub-sampled document images from the

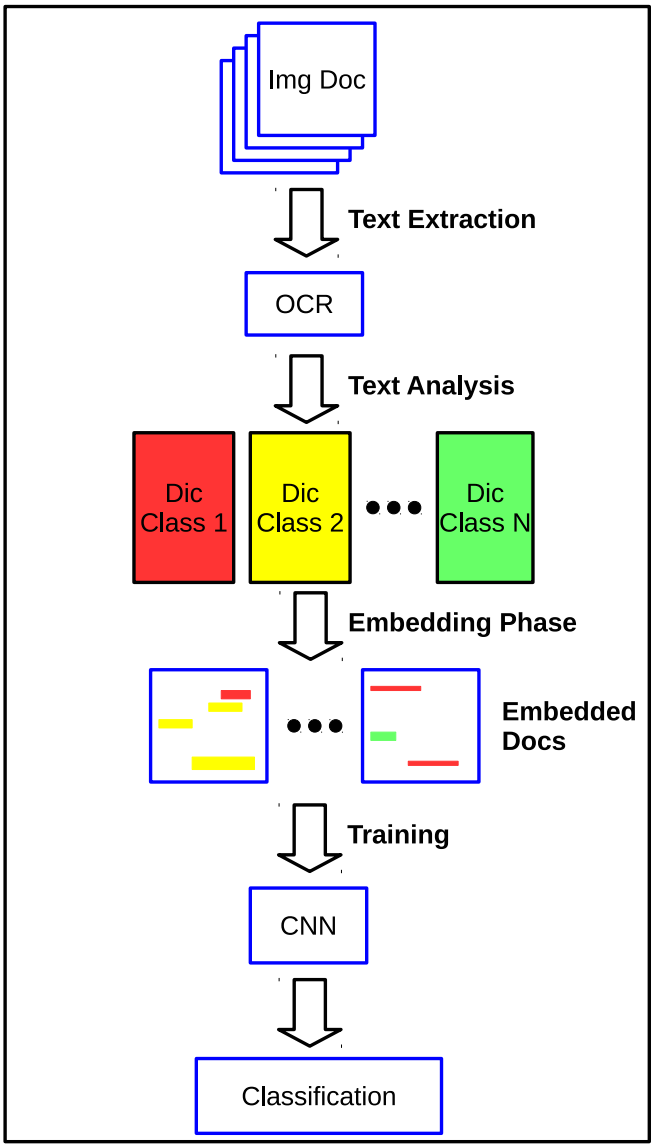


Figure 4.2: The building phase algorithm's pipeline.

previous phase.

4.2.1 Text extraction and analysis

Textual information is extracted from each document image through OCR, details about the adopted OCR system, and OCR engines evaluation are provided in Section 5.4.

Optical Character Recognition is a difficult task for noisy or low resolution documents [75], and thus, to discard reading errors, we preprocessed all the automatically

extracted text using Natural Language dictionaries and stop-word lists.

To emphasize class relevant textual content within each document image, for each class, a dictionary containing representative words is generated. This is done by collecting all the words extracted by the OCR engine, for all the images belonging to a specific class. To build the final dictionary, we adopt the weighting formula of Peñas *et al.* [21].

The relevance formula, associates a weight to each word comparing it to other classes' words, the higher the value is, the higher the relevance of the word for the specific class becomes. In detail, the relevance of a term t_i is computed as follows:

$$Relevance(t_i, sc, gc) = 1 - \frac{1}{\log_2\left(2 + \frac{F_{t_i,sc} \cdot D_{t_i,sc}}{F_{t_i,gc}}\right)} \tag{4.1}$$

where: (i) *sc* is the *specific corpus*, it corresponds to the subset of words, extracted from the starting set of words which are extracted from images of the specified class; (ii) *gc* is the *generic corpus*, it is composed by the whole set of words, extracted from all the classes ; (iii) $F_{t_i,sc}$ is the relative frequency of the term t_i in the specific corpus *sc*; (iv) $F_{t,gc}$ is the relative frequency of the same term t_i in the generic corpus *gc* and (v) $D_{t,sc}$ is the relative number of documents of *sc* in which the term t_i appears.

Once we have the relevance value associated to each word of each dictionary, we prune the set of word at the fixed threshold $r = 0.8$ where r has been empirically determined.

Following all the previously described steps, we obtain a set of dictionaries that represent classes' key-words that are used to underline text within images in the next step.

4.2.2 Embedding phase

Starting from the dictionaries of relevant words per classes, the aim of this phase is to embed textual information obtained from OCR within document images. We perform this to let relevant key-words information become recognizable even at low resolutions where the text is unreadable.

We create a specific visual color feature for each class key-word contained in the processed image and in, at least, one of the dictionaries built in Sec. 4.2.1. The added visual feature consists of a rectangle of the class color drawn across each class key-word found in the document image.

More in details, given an image, OCR is performed. The OCR engine output is composed both of the sequence of recognized words and their positions within the image. Once the words are extracted, for each word the system checks whether it belongs to one or more of class key-word dictionaries.

If the word belongs only to one dictionary, a rectangle of the associated class color is drawn across it using the obtained position coordinates, otherwise if it belongs to more

Figure 4.3: Key-words of three classes are underlined within images. It is easier to distinguish among the three classes “Family Status”, “Marriage Certificate” and “Residence Certificate”. The colors red, green and blue emphasize the content information in images, making it available for the training and classification phases.

than one dictionary, the rectangle is divided by the number of corresponding dictionaries and each part is colored using the associated class colors.

In Figure 4.3 the same documents shown in Figure 4.1 are shown after the embedding phase. Rectangles of respective classes’ colors are drawn; it can be easily noted that, for documents that belong to the same class, the associated class color is the mainly used: in the first line the red color is the most used, then, there is green, while in the third, the majority of the key-words’ rectangles are blue. Experiments reported in Chapter 5 show the effectiveness of the embedding phase for these specific three classes of documents.

During CNN training, document images are sub-sampled to fixed dimension therefore text becomes unreadable; however, the marked key-word rectangles remain visible and allow the model to infer textual content. Not only classes information are added but key-words’ positions are underlined, giving the model extra characteristics that are exploited during the classification phase.

4.2.3 Training phase

A deep Convolutional Neural Network is employed as classification model. Our proposal consists of using the images from the previous steps, where textual content information

are transformed into visual features and stored in document images, to train the network.

A common practice with CNN, is to exploit transfer learning [76]. This technique consists of pre-train a network on a large dataset, and then exploit it either as a fixed feature extractor or as a fine-tuning for the adopted CNN. In the first scenario, given a CNN a training phase on a different dataset is performed, after that the last fully-connected layer is removed and the remaining Convolutional Network is treated as a fixed feature extractor for the new dataset. On the other hand, the second strategy consists of fine-tune the weights of the pre-trained network by continuing the back-propagation.

A popular pre-training dataset is ImageNet. ImageNet dataset [77] is composed of over 15 million labeled high-resolution images in over 22000 categories, a subset of 1.2 million of images divided in 1000 categories is used in ILSVRC ImageNet challenges as training set, networks trained with such a training set are often used to transfer learning methodology.

Supported by the state-of-the-art results obtained by Harley *et al.* [42], we also implement transfer learning using ImageNet dataset and the CNN model of Krizhevsky *et al.* [49]. Implementation details are given is Section 5.1. Multiple experiments demonstrating the effectiveness of our method are reported in the next Section of this manuscript.

4.3 OCR Engines evaluation

We explored the performance of two Optical Character Recognition engines by executing several experiments. We evaluated two OCR engines: Tesseract OCR [78] and Leadtools OCR[1].

Tesseract is a widely used open-source OCR engine[2], it was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994. In 2005 Tesseract was open sourced by Hewlett-Packard and since 2006 it has been developed by Google.

On the other hand Leadtools is a commercial framework that provides with a family of toolkits designed to integrate raster, document, medical, multimedia and vector imaging into various type of applications. In particular the OCR module adds methods and classes for incorporating optical character recognition (OCR), magnetic ink character recognition (MICR), and optical mark recognition (OMR) technology into applications. The OCR engine is developed in-house by LEAD Technologies. Many companies that must deal with a large amount of image documents adopt Leadtools to elaborate images to solve various kind of tasks.

In developing OCR engines evaluation we have taken into account several aspects

[1] https://www.leadtools.com
[2] https://github.com/tesseract-ocr

such as image elaborating time, word positions within the document and reading correctness. A wide description of the evaluation metrics adopted, the dataset and image preprocessing, are shown in Section 5.4. A comparison of the results obtained during the proposed method evaluation phase varying the two OCR engines is also provided.

5
Experimental Phase

This chapter contains all the experiments conducted to evaluate our proposal. As discussed in the previous Chapters, we aim to add textual content information to document images, in order to help the CNN model to better distinguish among classes that have high intra-class visual similarity.

We compare our results against state-of-the-art outcomes in the document image classification field, and provide a comparison between a CNN that performs classification using the proposed textual content embedded document images and a CNN that uses exclusively textual information, showing that our methodology is significantly more effective.

5.1 Implementation details

The CNN is implemented using Caffe[79], and is based on the work of Krizhevsky *et al.* [49]. As described in Section 3.2, the network is composed of 5 Convolutional Layers, some of which are followed by Max-pooling Layers, three Fully-connected Layers with a final Softmax.

The full architecture can be written as 227×227 – $11 \times 11 \times 96$ – $5 \times 5 \times 256$ – $3 \times 3 \times 384$ – $3 \times 3 \times 384$ – $3 \times 3 \times 256$ – 4096 – 4096 – N. Where N is the number of categories and varying in respect with the utilized dataset. The input images size is 227×227, we down-sampled all the images to a fixed resolution of 256×256, in order to have a constant dimensional input. The Caffe implementation of the adopted CNN, randomly crops images from 256×256 to 227×227, this technique is usually employed

Table 5.1: Results achieved by the proposed method are reported, the CNN was trained using original images and with elaborated images. More accurate results are obtained training the network with images in where textual information are embedded.

Dataset	**Overall Accuracy**
Tobacco V	74.2%
Tobacco V & T	79.8%
Loan V	74.73%
Loan V & T	87.85%
Loan Certificates subset V	60%
Loan Certificates subset V & T	90%
Loan Contracts subset V	62.31%
Loan Contracts subset V & T	88.33%

because it helps data augmentation and reduces overfitting.

5.2 Results

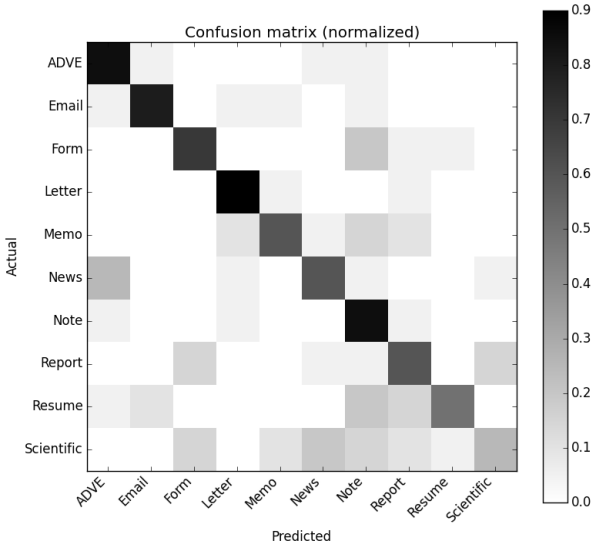
Experiments are performed to compare different results achieved through training the CNN model with original images and images containing textual information created through the embedding phase (Section 4.2.2).

For this set of experiments, the adopted OCR engine is Tesseract, a comparison between the two OCR engines is reported in Section 5.4.

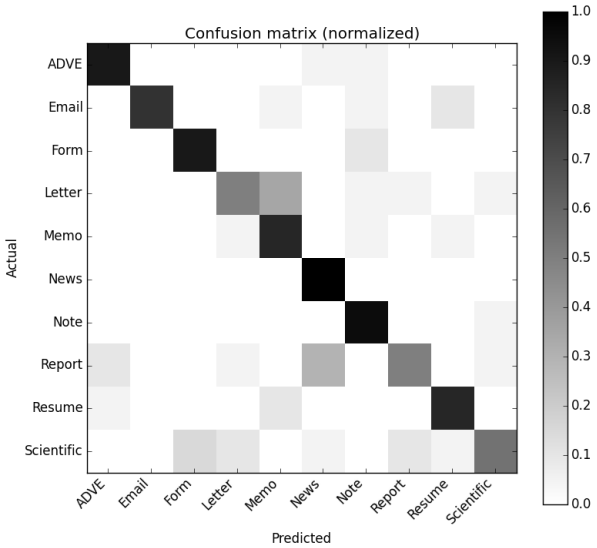
Overall accuracies achieved by the proposed model are reported in Table 5.1. The adopted CNN was trained by using the two different types of images. Results demonstrate that when the CNN is trained on the embedded images it reaches higher accuracy, passing from 74.2% to 79.8% on the Tobacco dataset and reaching 87.85% from 74.73% on the Loan dataset. Figure 5.1 shows the two confusion matrices computed for the Tobacco dataset respectively before and after the embedding phase; matrices underline that information deriving by the text is helpful for the classification task and show the achieved improvement.

Experiments conducted on the two subsets of the Loan dataset show a high groth in terms of accuracy, the values increase by 30% passing from original images to textual embedded document images for the Certificates subset, and by 26% for the Contracts subsets.

Confusion matrices of the testing phase for both the Loan and the two subsets are provided. Figure 5.2 displays the confusion matrices related to the Loan dataset and its two subsets, all the three matrices reflect the overall accuracy values obtained, demon-

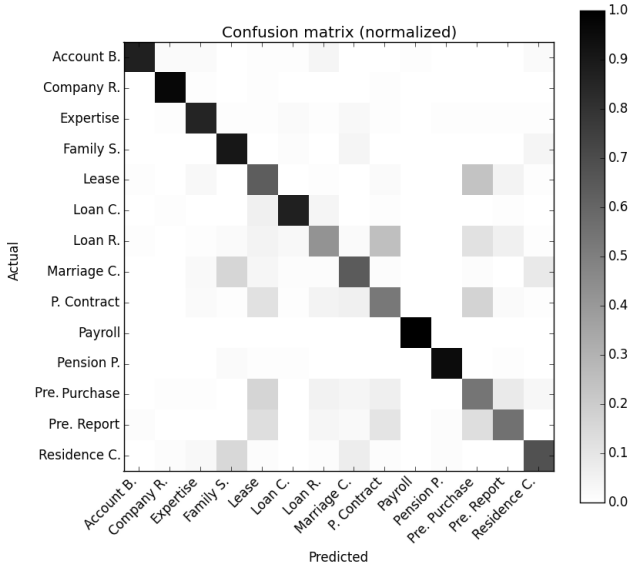


(a) Visual Features

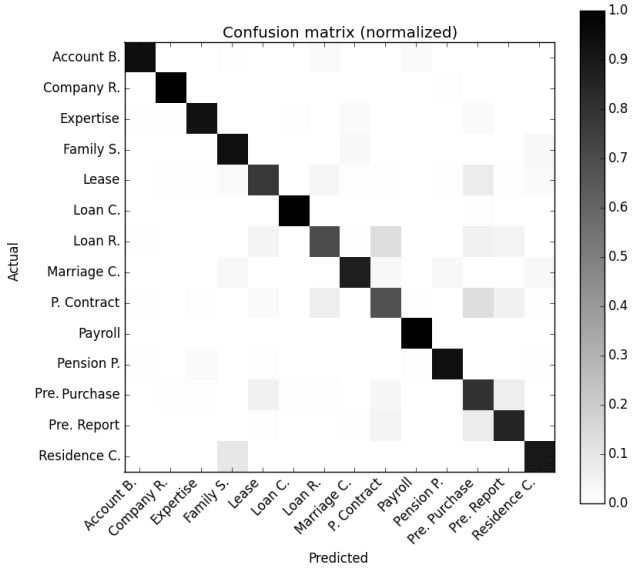


(b) Textual and Visual Features

Figure 5.1: Confusion matrices reporting the results achieved on the Tobacco dataset. Results have been calculated testing the model trained using original images (a) and using images that contained textual information (b).

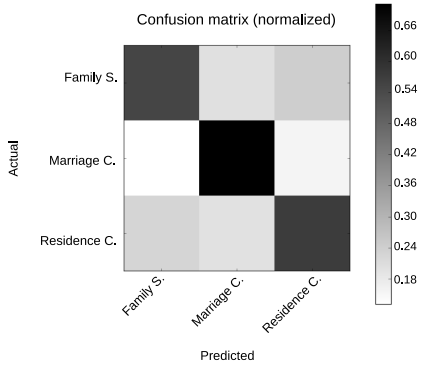


(a) LOAN - Visual Features

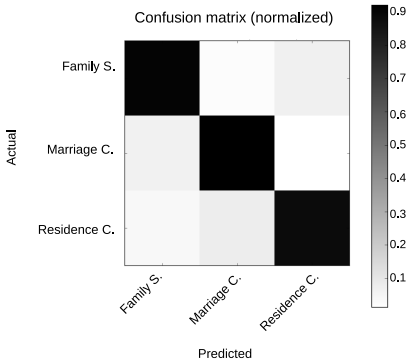


(b) LOAN - Textual & Visual Features

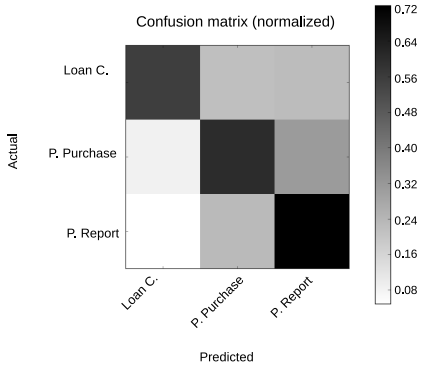
Figure 5.2: Confusion matrices reporting the results achieved on the Loan dataset. Outcomes have been calculated testing the model trained using original images (a) and using images that contained textual information (b).



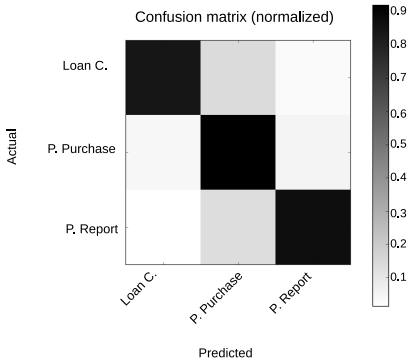
(a) CERTIFICATES - Visual Features



(b) CERTIFICATES - Textual & Visual Features



(c) CONTRACTS - Visual Features



(d) CONTRACTS - Textual & Visual Features

Figure 5.3: Confusion matrices reporting results achieved on the two subsets Certificates (a), (b) and Contracts (c), (d) are displayed. Matrices on the left side report the outcomes of the test phase, training the model using original images, while on the right side, the model was trained using images that contain underlined textual information. Matrices summarize the improvement achieved by using the elaborated images in the training phase.

Table 5.2: Comparison against models that use only visual features or only textual features are reported, our proposal overcomes or reaches almost the same results achieved by other methods on the Tobacco dataset.

Model	**Overall Accuracy**
Proposed V & T	79.8%
text-CNN Kim [45]	68.92 %
CNN Harley *et al.* [42]	79.9 %
CNN Kang *et al.* [8]	65.35%
Holistic BoW [42]	64.5%
H0V3 BoW [42]	67.9%
H2V0 BoW [42]	65.2%
H2V3 BoW [42]	68.1%
Pyramid BoW [42]	68.7%

strating the effectiveness of the propose methodology.

Moreover Table 5.2, reports a comparison carried on the Tobacco dataset among our proposal and related works that deals with only visual features[42, 8] or only textual features [45], which will be better analyzed in Section 5.3. To be thorough, results obtained by Harley *et al.* [42] using different state-of-the-art bag of words (BoW) approaches on the Tobacco dataset are also reported as Holistic, H0V3, H2V0, H2V3 and Pyramid BoW. The experiments were conducted by implementing different approaches to document representation [80, 81] that involve k-means clustered SURF [82] features, spatial pyramid and various combinations of horizontal and vertical partitions. For classification of the BoW features authors used a random forest with 500 trees.

Although the proposed method does not overcome state-of-the-art results, it reaches comparable outcomes and demonstrates that combining visual and textual feature is an interesting and effective approach to follow.

5.3 CNN applied to text

In this work, we aim to classify images adding textual information to them. Our goal is to demonstrate that a combination of textual and visual features is necessary for better understanding the document image content. To this end, we evaluate the classification of images using just the extracted text.

CNN proved effective in different Natural Language Processing tasks [59, 83]. A recent work proposed by Yoon Kim [45] implements a simple CNN, with one Convolutional Layer and achieves good classification performance across a range of text classification

Table 5.3: The same experiments performed exploiting visual features are performed using only textual features. The results show that the text is a relevant feature for the selected classes and can not be ignored to better perform the classification task, but at the same time a combinations of textual and visual feature is more effective.

Dataset	Text CNN	Proposed
Loan	69.89%	87.85%
Loan Certificates subset	87.15%	90%
Loan Contracts subset	86.31%	88.33%

tasks.

We use the model proposed by Kim *et al.* [45] to classify texts extracted from document images. For each document, text is divided into sentences. Each of these represents a document that has to be classified into one of the available classes. The results are shown in Table 5.3 and demonstrate that although the text-CNN reaches interesting results achieving 70% of accuracy on the whole Loan dataset, the exploitation of both visual and textual feature is more effective.

5.4 OCR evaluation

Several experiments were conducted in order to analyze the performances of Leadtools and Tesseract for our purpose.

In the experimental phase we adopt hOCR, a format for representing OCR output that includes layout information, character confidences, bounding boxes, and style information[1]. We built the so called dataset *OCR dataset* using 10 images from each of the 14 classes of the *Loan* dataset, totally 140 images were gathered and exploiting hOCR format we manually construct the OCR ground truth.

Three measures have been used during the evaluation phase:

- **Time**: measures how many seconds the OCR system needs to process an image.
- **Geometry**: evaluates if the detected position of the read words is correct. We employed DetEval [84], an evaluation tool used for object detection. DetEval takes as input two different bounding boxes of object locations. One corresponds to ground truth rectangle, the other one to detected rectangle. The bounding boxes are matched, and performance measures are calculated.
- **Edit Distance**: estimates whether the characters of the read word are correct. We used hOCR OCR-eval, it evaluates the actual OCR with respect to the ground

[1] https://github.com/tmbdev/hocr-tools

Table 5.4: Comparison between the evaluated OCR engines employed in the embedding phase. Experiment results show that the employ of Tesseract leads to better results in the classification task, but at the same time Leadtool achieved competitive performance and may be preferred to Tesseract in case of time consuming restriction.

Dataset	**Leadtools**	**Tesseract**
Loan	85.21%	87.85%
Loan Certificates subset	88%	90%
Loan Contracts subset	91.31%	88.33%

truth. This outputs the number of OCR errors due to incorrect segmentation and the number of OCR errors due to character recognition errors. It works by aligning segmentation components geometrically, and for each segmentation component that can be aligned, computing the string edit distance of the text the segmentation component contains.

For both the Geometry measure and the Edit Distance we calculated Precision Recall and F-measure, considering for the Geometry measure a detected bounding box as correct whether the IOU (intersection over union) of the detected rectangle and the ground truth is greater than 0.5, while in the case of the Edit Distance, a word is considered correct when the edit distance between the read word and the ground truth is equal to zero.

Different experiments were performed to understand which OCR engine better suite our purpose, we varied images DPI (dots per inch) and preprocessed documents using TEXTCLEANER [2], a tool designed to process scanned documents of text to clean the text background.

Results in term of Time and Precision (P) Recall (R) F-measure(F1) for Geometry and Edit Distance are reported in Tables 5.5, 5.6, 5.7 and 5.8 showing that although Leadtools is faster, Tesseract generally better perform on the OCR dataset.

Table 5.5 and Table 5.6 respectively report results achieved by Tesseract and Leadtools while varying images DPI, using original image documents (no preprocessing operations were applied), the better results are achieved by fixing DPI around 200 for both OCR engines. Figure 5.4 shows the time employed by Tesseract and Leadtools in performing OCR, underlining that both Leadtools and Tesseract elaborating times are directly proportional to the image DPI, but Tesseract is significantly slower than Leadtools, specially when DPI are grater than 200. Although Leadtools is faster in elaborating images, Figures 5.5 and 5.6 show that Tesseract reached more accurate results

[2] http://www.fmwconcepts.com/imagemagick/textcleaner/

Table 5.5: Results in terms of Time, Precision(P) Recall (R) and F-measure (F1) of Geometry (Geo.) and Edit Distance (Edit D.) measures, achieved by Tesseract on the *OCR dataset*, while varying images DPI. Values underline that Tesseract achieved the highest performance working on images of 200 DPI.

DPI	Geo. R	Geo. P	Geo. F1	Edit D. P	Edit D. R	Edit D. F1	Time ms.
150	0.62	0.49	0.53	0.20	0.24	0.22	27878.87
200	0.67	**0.50**	**0.56**	**0.30**	**0.36**	**0.32**	31219.53
250	0.67	0.46	0.53	0.25	0.31	0.28	40063.10
300	**0.68**	0.49	0.55	0.27	0.33	0.30	48369.29
350	0.66	0.48	0.54	0.27	0.33	0.30	50446.76
400	0.66	0.49	0.55	0.25	0.30	0.27	54290.16
450	0.64	0.47	0.53	0.25	0.31	0.28	52191.81
500	0.65	0.46	0.52	0.22	0.29	0.25	58686.72
550	0.64	0.44	0.51	0.21	0.27	0.23	64118.89
600	0.63	0.46	0.51	0.20	0.27	0.23	64012.64
650	0.62	0.46	0.51	0.19	0.24	0.22	63678.80

in terms of Geometry and Edit Distance.

Preprocessing was done varying 5 parameters provided by TEXTCLEANER: (1) enhance "-e": enhance image brightness before cleaning; (2) filtersize "-f": size of filter used to clean background; (3) offset "-o": offset of filter in percentage used to reduce noise, default=5; (4) threshold "-t": text smoothing threshold; (5) sharpamt "-s": sharpening amount in pixels; default=0. Tables 5.7 and 5.8 shows results obtained fixing DPI at 200 and varying preprocessing parameters. From experiments, preprocessing comes out to be useless for this type of images, the highest accuracy values are reached without any preprocessing step for both Leadtools and Tesseract.

Table 5.4 reports results achieved by the proposed method employing Tesseract and Leadtool during the Embedding Phase. Tesseract and Leadtools achieved competitive results in terms of accuracy; Leadtools is significantly faster than Tesseract, but Tesseract generally achieved higher accuracies. In conclusion, both the OCR engines may be employed in our proposed method; the choice of the OCR method to adopt could be based on considering which aspect is more important for the application, time or classification accuracy.

Table 5.6: Results in terms of Time, Precision(P) Recall (R) and F-measure (F1) of Geometry (Geo.) and Edit Distance (Edit D.) measures, achieved by Leadtools on the *OCR dataset*, while varying images DPI. Results achieved by Leadtools show that OCR is better performed on images with 200 DPI.

DPI	Geo. R	Geo. P	Geo. F1	Edit D. P	Edit D. R	Edit D. F1	Time ms.
150	0.58	**0.58**	0.56	0.23	0.24	0.23	9961.51
200	**0.62**	0.55	**0.58**	**0.26**	**0.30**	**0.31**	10772.38
250	0.57	0.48	0.50	0.20	0.25	0.26	13370.58
300	0.57	0.47	0.49	0.20	0.26	0.28	15509.90
350	0.61	0.53	0.54	0.20	0.24	0.27	12478.28
400	0.59	0.51	0.53	0.18	0.22	0.25	13007.70
450	0.57	0.48	0.50	0.16	0.21	0.24	13943.92
500	0.60	0.52	0.53	0.17	0.21	0.23	14584.58
550	0.59	0.50	0.52	0.16	0.20	0.22	14737.63
600	0.57	0.48	0.50	0.14	0.18	0.20	15047.45
650	0.57	0.45	0.49	0.13	0.17	0.19	16356.45

Table 5.7: Results in terms of Time, Precision(P) Recall (R) and F-measure (F1) of Geometry (G) and Edit Distance (E) measures, achieved by Tesseract on the *OCR dataset*, using preprocessed images. Results show that the preprocessing operations performed on images do not help the OCR engine.

Preprocessing	G R	G P	G F1	E P	E R	E F1	Time ms.
NO PREPROC	**0.67**	**0.50**	**0.56**	**0.29**	**0.35**	**0.32**	29366.45
-e none -f10 -o5 -s1	0.61	0.45	0.50	0.24	0.31	0.27	27402.53
-e none -f10 -o5 -t25 -s1	0.62	0.46	0.52	0.24	0.31	0.27	24573.97
-e none -f10 -o5 -t50 -s1	0.62	0.46	0.51	0.22	0.29	0.25	25405.88
-e none -f10 -o5 -t75 -s1	0.48	0.34	0.39	0.11	0.15	0.13	22725.61
-e normalize -f25 -o20	0.57	0.42	0.48	0.19	0.25	0.21	21286.05
-e normalize -f15 -o5	0.60	0.46	0.51	0.22	0.28	0.25	24175.92
-e stretch -f15 -o5 -t15 -s1	0.62	0.47	0.52	0.24	0.30	0.26	24701.14
-e stretch -f25 -o20 -s1	0.58	0.42	0.48	0.19	0.25	0.22	21952.77

Table 5.8: Results in terms of Time, Precision(P) Recall (R) and F-measure (F1) of Geometry (G) and Edit Distance (E) measures, achieved by Leadtools on the *OCR dataset*, using preprocessed images. The highest values are reached without preprocessing.

Preprocessing	G R	G P	G F1	E P	E R	E F1	Time ms.
NO PREPROC	**0.63**	**0.60**	**0.60**	**0.28**	**0.31**	**0.30**	9880.23
-e none -f10 -o5 -s1	0.62	0.48	0.52	0.22	0.29	0.25	8451.30
-e none -f10 -o5 -t25 -s1	0.63	0.50	0.54	0.24	0.30	0.26	8739.89
-e none -f10 -o5 -t50 -s1	0.60	0.47	0.51	0.19	0.25	0.21	7742.48
-e none -f10 -o5 -t75 -s1	0.41	0.29	0.33	0.07	0.12	0.09	6844.71
-e normalize -f25 -o20	0.50	0.37	0.42	0.14	0.19	0.16	6674.77
-e normalize -f15 -o5	0.63	0.50	0.54	0.22	0.28	0.24	7267.79
-e stretch -f15 -o5 -t15 -s1	0.61	0.56	0.57	0.27	0.30	0.29	8898.50
-e stretch -f25 -o20 -s1	0.51	0.38	0.42	0.14	0.19	0.16	6869.16

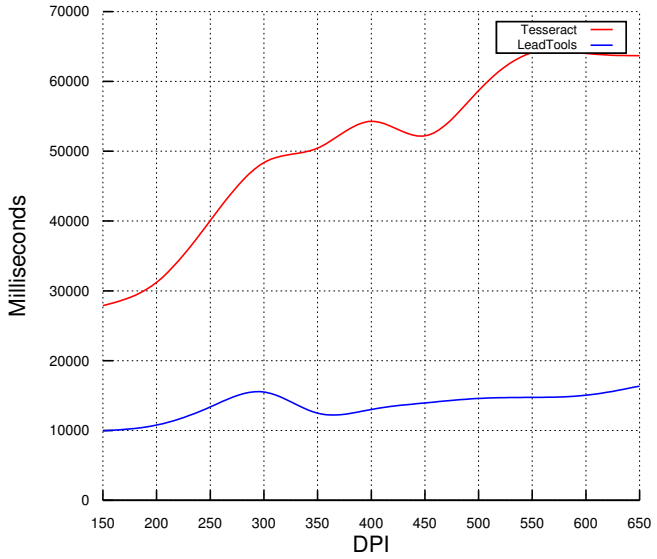


Figure 5.4: OCR elaborating Time: the graph shows the time that Leadtools and Tesseract need, in average, to process an image of the OCR dataset. Values are calculated varying images DPI from 150 to 650. Unsurprisingly, the time is directly proportional to the number of DPI. Nevertheless Leadtools performs significantly better in terms of time even with an high number of DPI.

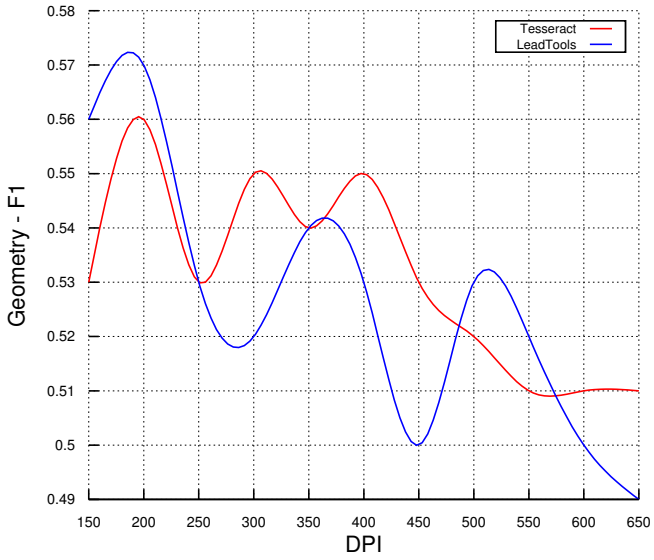


Figure 5.5: OCR Geometry: the graph reports values achieved by the two OCR engines (Leadtools and Tesseract) considering the F-measure of the Geometry on the OCR dataset. On average Tesseract performs better than Leadtools.

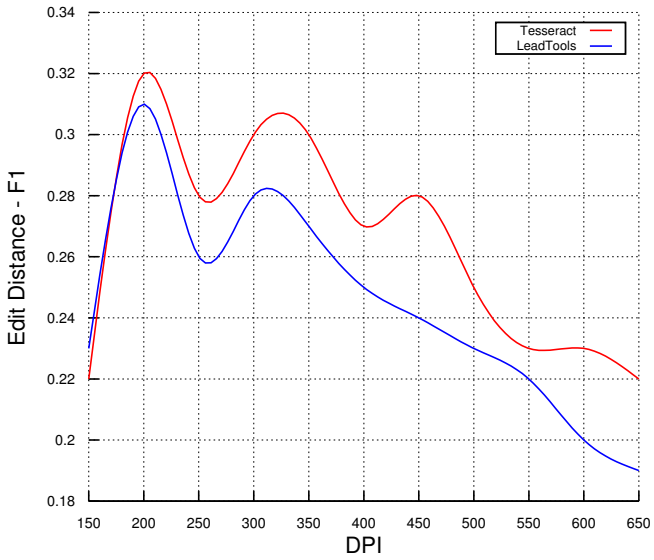


Figure 5.6: OCR Edit Distance: results obtained by the two OCR engines (Leadtools and Tesseract) considering the F-measure of the Edit Distance metric on the OCR dataset. Starting from approximately 190 DPI Tesseract overcomes Leadtools performance.

6

Conclusions and Future Directions

6.1 Conclusion

In this thesis a new methodology that exploits both textual and visual features for document image classification has been proposed.

Document image classification is not a trivial task and presents lots of obstacles (as introduced in Chapter 1) due to the large number of different documents that may present several different issues (such as same visual style among different classes, damages, content impossible to retrieve etc) and therefore are not easy to categorize.

In literature this problem is often faced by performing document classification relying exclusively on textual content extracted from the processed documents (content-based) or relying on structural and visual aspects to perform the classification (image analysis). As introduced in Chapter 3, both approaches present limitations. Content-based approaches can be effective for simple and well-made artificial documents, but they ignore important visual document features that may play an important role in the final document classification predictions and they are limited to well printed documents which contain textual information. On the other hand, techniques that rely on visual aspects usually involve manually chosen template or ad hoc layout-based features.

Following the outstanding results obtained with Deep Convolutional Neural Networks in the Computer Vision research field, such models were also adopted to perform document image classification stating the state-of-the-art.

The main contribution of this thesis is the exploitation of textual and visual features through an approach that uses Convolutional Neural Networks; the approach is widely

described in Chapter 4. Summarizing, our proposal consists of embedding content information extracted from text within document images, with the aim of adding elements which help the system to distinguish different classes that appear visually indistinguishable. Our method is able to take advantage of the extra textual key-word information provided by colored rectangles to reach more satisfying document image classification accuracy, especially for document classes having similar visual styles.

Experiments shown in Chapter 5 prove that the overall document classification accuracy of a Convolutional Neural Network trained using these text-augmented document images, is considerably higher than the one achieved by a similar model trained solely on classic document images, especially when different classes of documents share similar visual characteristics. The comparison between our method and state-of-the-art approaches demonstrates the effectiveness of combining visual and textual features.

6.2 Future Directions

Future extensions of our work may rely on testing different embedding methods that permit to apply this approach on classification tasks that involve a consistent number of classes. The approach we followed was born after considering that in most cases solely visual style or content were not enough to distinguish among very similar but still different classes. These cases were reproduced through the two subsets of the Loan dataset, all documents belonging to the two subsets are extracted from the same subareas (Contracts and Certificates) and therefore present very similar layouts. On the other hand all the documents were produced by costumers and the images quality was not always as good as an OCR engine require to obtain an acceptable result. We demonstrate that our proposal achieves excellent results in this particular cases, but what if we think to extend it to a consistent number of classes? During the embedding phase a color is associated to each dictionary class, but this may not work well considering hundreds of classes. The first future extension may consist of working on a consistent number of classes, evaluating different embedding methods.

All the works I carried out during my three years Ph.D were based on extract information considering visual and textual features. The final work "Embedded Textual Content for Document Image Classification with Convolutional Neural Networks" [20], can be viewed as an example of a real application scenario, but this approach is not limited to image document classification. Another direct future direction may consist of exploiting the extraction of visual and textual features in other research fields. Text and images are associated in plenty of contexts, for example in most of the social networks and market places, such as Facebook, Amazon, TripAdvisor to mention the most well known, users are allowed to insert comments or reviews that may contain text and images at the same time. News, blogs and articles also represent information using both text

and images. Taking all this into consideration, extract information from both images and text may represent an effective approach in several research areas such as Sentiment Analysis, Image and Document Retrieval, Image ranking, Information Extraction and Natural Language Processing.

Colophon

– This thesis was written using LaTeX.

– The LaTeX template for this thesis was made by Carullo Moreno.

– The algorithm described in Chapter 4 was developed using MATLAB® 2014b and Java for what concerns the text extraction, the embedding phase was developed using C++.

– The Apache OpenNLP library[1] was used for text analysis; the OpenCV (Open Source Computer Vision) library[2] was used to implement the embedding phase; NVIDIA digits[3] and Caffe[4] were used during the CNN's creation, training and testing phases.

– Two NVIDIA GTX 980 were used: one donated by 7pixel and the other given to Artelab[5] by NVIDIA Corporation for research purposes.

– All of the experiments of this thesis were executed on an Intel® Xeon® CPU E5-1620 v3 @ 3.50GHz with 64Gb of RAM and Linux Mint 17.1 Rebecca OS.

[1] http://opennlp.apache.org/
[2] http://opencv.org/
[3] http://developer.nvidia.com/digits
[4] http://caffe.berkeleyvision.org/
[5] http://artelab.dicom.uninsubria.it/

Bibliography

[1] L. O'Gorman, *Document Image Analysis: An Executive Briefing*, L. O'Gorman and R. Kasturi, Eds., 1997.

[2] R. Kasturi, L. O'Gorman, and V. Govindaraju, "Document image analysis: A primer," *Sadhana*, 2002.

[3] N. Chen and D. Blostein, "A survey of document image classification: Problem statement, classifier architecture and performance evaluation," *International Journal on Document Analysis and Recognition (IJDAR)*, 2007.

[4] T. Brükner, P. Suda, H. Block, and G. Maderlechner, "In-house mail distribution by automatic address and content interpretation," in *Symposium on Document Analysis and Information Retrieval (SDAIR)*, 1996.

[5] F. Cesarini, M. Lastri, S. Marinai, and G. Soda, "Encoding of modified x-y trees for document classification," in *International Conference Document Analysis and Recognition (ICDAR)*, 2001.

[6] C. Shin, D. Doermann, and A. Rosenfeld, "Classification of document pages using structure-based features," *International Journal on Document Analysis and Recognition*, 2001.

[7] H. Ogata, S. Watanabe, A. Imaizumi, T. Yasue, N. Furukawa, H. Sako, and H. Fujisawa, "Form-type identification for banking applications and its implementation issues," in *DRR*, 2003.

[8] Le Kang, Jayant Kumar, Peng Ye, Yi Li, and David Doermann, "Convolutional Neural Networks for Document Image Classification," in *International Conference on Pattern Recognition (ICPR)*, 2014.

[9] L. Noce, A. Zamberletti, I. Gallo, G. Piccoli, and J. A. Rodriguez, "Automatic prediction of future business conditions," in *International Conference on Natural Language Processing (PolTAL)*, 2014.

[10] I. Gallo, A. Zamberletti, and L. Noce, "Content extraction from marketing flyers," in *Computer Analysis of Images and Patterns - 16th International Conference, CAIP 2015*, 2015.

[11] L. Noce, I. Gallo, and A. Zamberletti, "Combining textual and visual features to identify anomalous user-generated content," *International Journal of Computational Linguistics and Applications (IJCLA)*, 2015.

[12] ——, "Query and product suggestion for price comparison search engines based on query-product click-through bipartite graphs," in *International Conference on Web Information Systems and Technologie (WEBIST)*, 2016.

[13] A. Zamberletti, I. Gallo, and L. Noce, "Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions," in *International Workshop on Robust Reading, (IWRR in conjunction with ACCV)*, 2014.

[14] I. Gallo, A. Zamberletti, S. Albertini, and L. Noce, "High entropy ensembles for holistic figure-ground segmentation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[15] I. Gallo, A. Zamberletti, and L. Noce, "Interactive object class segmentation for mobile devices," in *Conference on Graphics, Patterns and Images, (SIBGRAPI)*, 2014.

[16] A. Zamberletti, I. Gallo, S. Albertini, and L. Noce, "Neural 1d barcode detection using the hough transform," *IPSJ Transactions on Computer Vision and Applications*, 2015.

[17] A. Zamberletti, I. Gallo, and L. Noce, "Augmented text character proposals and convolutional neural networks for text spotting from scene images," in *Asian Conference on Pattern Recognition (ACPR)*, 2015.

[18] I. Gallo, A. Zamberletti, and L. Noce, "Robust angle invariant GAS meter reading," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2015.

[19] A. Calefati, I. Gallo, A. Zamberletti, and L. Noce, "Using convolutional neural networks for content extraction from online flyers," in *Proceedings of the 16th ACM Symposium on Document Engineering (DocEng 2016)*, 2016.

[20] L. Noce, I. Gallo, A. Zamberletti, and A. Calefati, "Embedded textual content for document image classification with convolutional neural networks"," in *ACM Symposium on Document Engineering (DocEng 2016)*, 2016.

[21] A. Penas, F. Verdejo, and J. Gonzalo, "Corpus-based terminology extraction applied to information access," in *Corpus Linguistics*, 2001.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] Z. Bozanic, D. T. Roulstone, and A. Van Buskirk, "Management earnings forecasts and forward-looking statements," 2013.

[24] M. A. Hasan, N. Parikh, G. Singh, and N. Sundaresan, "Query suggestion for e-commerce sites," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, 2011.

[25] ShoppyDoo. (2015) http://www.shoppydoo.it/.

[26] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine," in *Proceedings of the 10th International Conference on World Wide Web*, 2001.

[27] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *Proceedings of the 2004 International Conference on Current Trends in Database Technology*, 2004.

[28] R. Zanon, S. Albertini, M. Carullo, and I. Gallo, "A new query suggestion algorithm for taxonomy-based search engines," in *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, 2012.

[29] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[30] Jayant Kumar and David Doermann, "Unsupervised Classification of Structurally Similar Document Images," in *Intl. Conf. on Document Analysis and Recognition (ICDAR 13)*, 2013.

[31] C. Shin, D. S. Doermann, and A. Rosenfeld, "Classification of document pages using structure-based features." *International Journal on Document Analysis and Recognition (IJDAR)*, 2001.

[32] S. Baldi, S. Marinai, and G. Soda, "Using tree-grammars for training set expansion in page classification," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2003.

[33] M. Diligenti, P. Frasconi, and M. Gori, "Hidden tree markov models for document image classification." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2003.

[34] J. Liang, D. S. Doermann, M. Y. Ma, and J. K. Guo, "Page classification through logical labelling." in *International Conference on Pattern Recognition (ICPR)*, 2002.

[35] C. Shin and D. S. Doermann, "Document image retrieval based on layout structural similarity." in *International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2006.

[36] J. Hu, R. Kashi, and G. Wilfong, "Comparison and Classification of Documents Based on Layout Similarity," *Information Retrieval*, 2000.

[37] P. Sarkar, "Learning image anchor templates for document classification and data extraction," in *International Conference on Pattern Recognition (ICPR)*, 2010.

[38] S. Argamon, O. Frieder, D. A. Grossman, and D. D. Lewis, "Content-based document image retrieval in complex document collections," in *Document Recognition and Retrieval (DRR)*, 2007.

[39] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *The IEEE Conferenbasedce on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[40] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee, "Improving object detection with deep convolutional networks via bayesian optimization and structured prediction," *Computing Research Repository (CoRR)*, 2015.

[41] R. Wu, B. Wang, W. Wang, and Y. Yu, "Harvesting discriminative meta objects with deep CNN features for scene classification," *Computing Research Repository (CoRR)*, 2015.

[42] A. W. Harley, A. Ufkes, and K. G. Derpanis, "Evaluation of deep convolutional nets for document image classification and retrieval," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[43] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J. Heard, "Building a test collection for complex document information processing," in *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[44] Jayant Kumar, Peng Ye, and David Doermann, "Structural Similarity for Document Image Classification and Retrieval," *Pattern Recognition Letters*, 2013.

[45] Y. Kim, "Convolutional neural networks for sentence classification," *Computing Research Repository (CoRR)*, 2014.

[46] X. Zhang and Y. LeCun, "Text understanding from scratch," *Computing Research Repository (CoRR)*, 2015.

[47] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *IEEE*, 1998.

[48] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *Computing Research Repository (CoRR)*, 2014.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[50] H. D. H. and W. T. N., "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, 1962.

[51] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[52] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, 2012.

[53] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[54] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, 2013.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.

[57] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[58] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[59] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *Computing Research Repository (CoRR)*, 2014.

[60] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic clustering and convolutional neural network for short text categorization," in *Association for Computational Linguistics (ACL)*, 2015.

[61] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," in *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015.

[62] ——, "Semi-supervised convolutional neural networks for text categorization via region embedding," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[63] Y. Zhang and B. C. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification." *CoRR*, 2015.

[64] T. Nguyen and R. Grishman, "Relation extraction: Perspective from convolutional neural networks," in *Workshop on Vector Space Modeling for NLP at NAACL 2015*, 2015.

[65] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang, "Modeling mention, context and entity with neural networks for entity disambiguation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[66] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *International Conference on Computational Linguistics (COLING)*, 2014.

[67] M. G. X. H. L. D. Jianfeng Gao, Patrick Pantel, "Modeling interestingness with deep neural networks," Tech. Rep., 2014.

[68] J. G. L. D. G. M. Yelong Shen, Xiaodong He, "A latent semantic model with convolutional-pooling structure for information retrieval," in *International Conference on Information and Knowledge Management (CIKM)*, 2014.

[69] J. Weston, S. Chopra, and K. Adams, "#tagspace: Semantic embeddings from hashtags," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[70] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *International Conference on Machine Learning (ICML-14)*, 2014.

[71] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *International Conference on Neural Information Processing Systems (NIPS)*, 2015.

[72] D. S. A. M. R. Yoon Kim, Yacine Jernite, "Character-aware neural language models," in *Advances in Neural Information Processing Systems (AAAI)*, 2016.

[73] H. Peng, F. Long, Z. Chi, and W.-C. Siu, "Document image template matching based on component block list." *Pattern Recognition Letters*, 2001.

[74] E. Appiani, F. Cesarini, A. M. Colla, M. Diligenti, M. Gori, S. Marinai, and G. Soda, "Automatic document classification and indexing in high-volume applications," *International Journal on Document Analysis and Recognition (IJDAR)*, 2001.

[75] A. Kae, G. B. Huang, C. Doersch, and E. G. Learned-Miller, "Improving state-of-the-art OCR through high-precision document-specific modeling." in *The IEEE Conferenbasedce on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[76] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Computing Research Repository (CoRR)*, 2014.

[77] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *The IEEE Conferenbasedce on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[78] R. Smith, "An overview of the tesseract OCR engine," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[79] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia (ACMMM)*, 2014.

[80] J. Kumar, P. Ye, and D. Doermann, "Structural similarity for document image classification and retrieval," *Pattern Recognition Letters (PRL)*, 2014.

[81] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[82] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision (ECCV)*, 2006.

[83] C. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *International Conference on Computational Linguistics (COLING)*, 2014.

[84] C. Wolf and J.-M. Jolion, "Object count/area graphs for the evaluation of object detection and segmentation algorithms," *International Journal on Document Analysis and Recognition (IJDAR)*, 2006.