# POLITECNICO DI MILANO

**Scuola di Ingegneria Industriale e dell'Informazione**

**Dipartimento di Elettronica, Informazione e Bioingegneria**

**MSc in Computer Science and Engineering**


# Document layout analysis: segmentation and classification with computer vision and deep learning techniques


**Supervisor: Prof. Matteo MATTEUCCI**

**Master Thesis by:**

**Simone BIFFI, ID 853393**



**Academic Year 2016-2017**

# Abstract

Page segmentation is the task of decomposing document scans into many different regions such as text, images, tables and graphs. It is the first step in document image recognition and it is still a challenging problem due to the variety of possible document layouts. In this thesis we study an effective method for accomplish the page segmentation phase and improve state of the art in this task. To accomplish this goal we explore two deep learning models with typical structures fine-tuned using the transfer learning technique: You Only Look Once (YOLO) and AlexNet. Our approach out-performs the state of the art in this task enriching the document parsing pipeline composed by recursive segmentation, homogeneity criterion, and the median filters, through the use of the convolutional neural networks. The proposed method has been tested with the dataset of "ICDAR2009 page segmentation" competition and compared with the state of the art in the text/non-text segmentation task. The results of these tests show that the proposed system overperform the previous one improving both text and non-text segmentation.

# Sommario

La segmentazione di una pagina consiste nello scomporre l'immagine di un documento nelle differenti regioni che lo compongono come: regioni di testo, immagini, tabelle e grafici. Questa è la prima fase nel contesto del riconoscimento dei documenti ed è ancora un problema non risolto a causa della varietà di stili differenti nei layout. In questa tesi abbiamo studiato un metodo efficace che migliorasse lo stato dell'arte per quanto riguarda la fase di segmentazione della pagina. Abbiamo studiato due diversi modelli di deep learning, AlexNet e You Only Look Once (YOLO), addestrati attraverso la tecnica transfer learning, per stabilire le loro performance sul dataset da noi creato. Abbiamo inoltre implementato l'algoritmo oggi considerato lo stato dell'arte nella segmentazione della pagina, modificandolo in parte, ovvero sfruttando la segmentazione ricorsiva, il criterio di omogeneità e i filtri mediani, e aggiungendo l'utilizzo della rete neurale convoluzionale che ha dato migliori risultati sul dataset da noi realizzato: AlexNet. Il nostro metodo è stato poi testato sul dataset della competizione "ICDAR2009 page segmentation" ed i risultati sono stati confrontati con lo stato dell'arte nella segmentazione della pagina considerando la separazione tra testo e non testo. I risultati mostrano che il sistema proposto migliora le prestazioni del precedente sia nella segmentazione delle regioni testuali sia in quella delle regioni non testuali.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the beginning of the 90's, methodologies, algorithms and systems were invented and developed for document image analysis in order to extract information from document images in a "human-like" way. Extracting information from a document refers to locating and extracting lines, photos or text region [1]. Such document processing can be useful in many application such: object-oriented rendering, extracting flowcharts and diagrams from scanned document for computer storage, document retrieval, query-images/texts or optical character recognition (OCR).

Document parsing is used as an initial step for document structure analysis prior to OCR and document retrieval [2] [3]. With the advent of modern publishing technologies, the layout of today's documents has never been more complex. Most of them contain not only text and background regions, but also graphics, tables and pictures. Such variety of document layouts makes document parsing still a challenging problem [4].

The purpose of the thesis is to improve the performance of a document layout analysis system combining deep learning algorithms with computer vision and statistical techniques, and using it to analyze real documents. More precisely, our work starts from a system that analyzes documents in order to find connected components and, through heuristic filters and recursive statistical segmentation, it classifies them in text and non-text components; successively the classification is refined by a deep learning classifier.

We firstly cope with the problem of finding connected components in a document and classifies them; structural features such as width, height, density of pixels, number of components in the same row and column, are evaluated to perform a first separation between text and non-text components. Then, via a recursive segmentation process, the text document is decomposed in homogeneous region. The components inside this region are successively grouped to obtain words, lines and paragraphs. In the last step we classify, through a convolutional neural network trained on our dataset with the transfer learning technique, each text region, and non-text region extracted from the non-text document into text, image, graph, table or noise.

We tested the enhanced document layout analysis on the PRImA Layout Analysis Dataset [5] and we evaluated it comparing with the state of the art [6] in terms of accuracy, recall, and f-measure on the text/non-text separation task. Results show that, in general, the added classifier is able to improve the performance of the analysis system. However, not all the documents benefit in the same measure from our work, due to their different physical features.

The thesis has the following structure:

- **Chapter 2** gives a detailed overview of the state of the art in terms of document analysis, along with a presentation of the transfer learning technique, AlexNet, i.e. the convolutional neural network used for classification, and YOLO.

- **Chapter 3** presents our solution to the problem of document parsing and segmentation, and the way in which the classifier has been included into a document parsing pipeline.

- **Chapter 4** shows the used tools, the dataset we created, and the way in which the classifier has been trained; reports the experimental evaluation of our approach, compares the results with the state of the art in the text, non-text segmentation task and shows the limits of our system as well as the limits of other tested methods.

- **Chapter 5** summarizes the purpose and the outcomes of this thesis; it also proposes some directions for future improvements.

# Chapter 2

# State of the art

In this chapter we present the state of the art in the document analysis strategies, focusing specifically on segmentation approaches. Then we show the structure of two popular neural network Alexnet (the convolutional neural network used in our system) and YOLO, and finally we give an introduction of the transfer learning technique.

Although there are many researches dealing with segmentation, few provide a pipeline that combines all the steps necessary for document parsing, and even fewer are those that use convolutional neural networks. The latter can be used as a classification step (Section 2.2) or as an end-to-end method (Section 2.3) for the detection and subsequent classification of the document regions.

The document parsing can be the preliminary step of different potential applications e.g. the automatic classification of all the documents related to a potential client [7] or a system which authenticates the identity card [8].

## 2.1 Document analysis

The process leading to digitalized document understanding is carried out in multiple steps, such as image preprocessing, physical structure recognition, and logical structure recognition; the last two may sometimes be mixed in a unique recognition process (Figure 2.1).

### 2.1.1 Image preprocessing

Starting from a physical printed document, the first step is the digitization of the hard copy in order to get an electronic version of the document. Such acquired document images contain noise and artifacts that are mainly introduced during the document digitization phase or during the scanning phase; they may be caused by dust, poor scanning conditions, or paper deterioration [9].

In order to make the analysis algorithms more robust to noise and deformations, an image preprocessing phase is applied on the digitized images, which tries to correct image or imperfections and prepares them for future high-level processing. The preprocessing phase may contain many steps such as noise reduction, skew detection and correction, image smoothing, and skeletonization.

A particular attention is usually given to binarization process which aims to convert an input gray level image $g(x, y)$ into a bi-level representation by quantization of the image into two values: 0 and 1. If the input document is colored, its RGB components are first combined to give a grayscale image [10] [6]. Binarization generally involves two steps including determination of a gray threshold according to some objective criteria and assigning each pixel

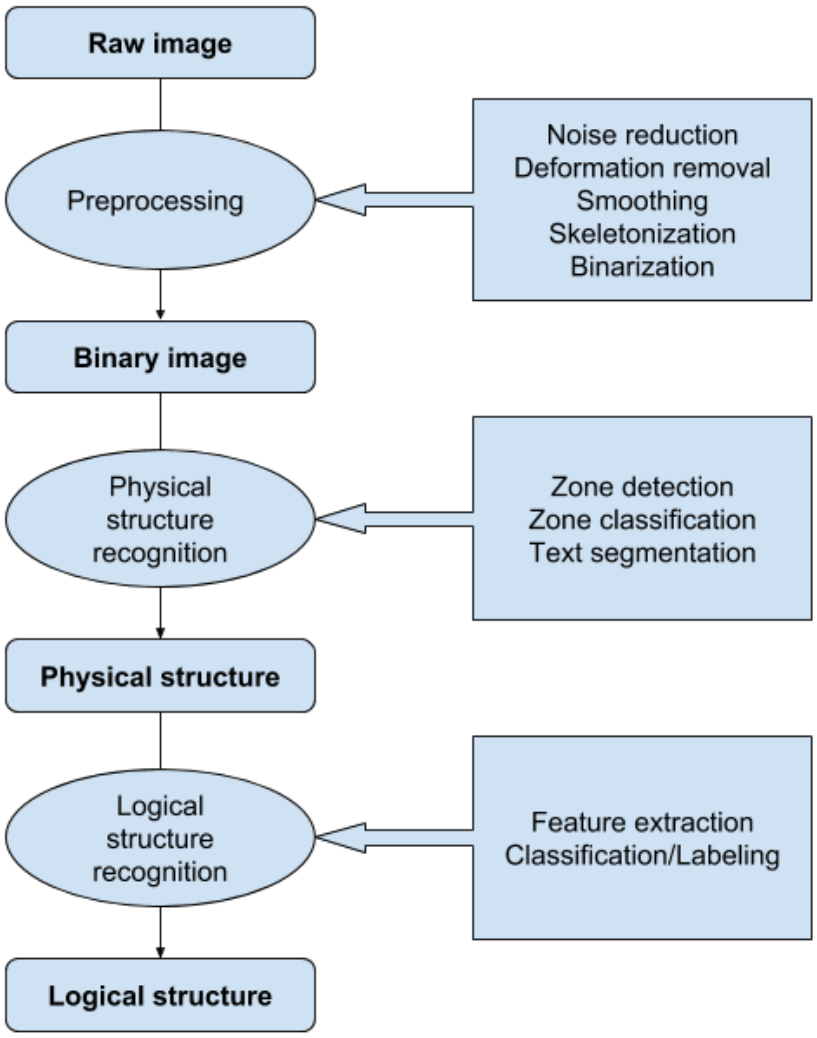


*Figure 2.1: The division of common document understanding process: image preprocessing, physical structure recognition, and logical structure recognition.*

to one class of background or foreground. If the pixel intensity is greater than the determined threshold then it belongs to foreground class, otherwise to the background [11].

**OTSU's Technique** The OTSU's technique is based on a very simple idea, which is to find the threshold that minimizes the weighted within-class variance [12]. This turns out to be the same as maximizing the between-class variance operating directly on the gray level histogram. There is no use of spatial coherence, or any other notion of object structure. Stationary statistics are assumed although it can be modified to be locally adaptive [13].

Otsu's thresholding method corresponds to the linear discriminant criteria that assumes that the image consists of only foreground and background. Otsu set the threshold so as to try to minimize the overlapping of the class distributions [13]. The Otsu's method segments the image into $T_0$ and $T_1$, where region $T_0$ is the set of intensity level from 0 to $t$ and region $T_1$ is the set of intensity from $t$ to $l$, where $t$ is the threshold value and $l$ is the image maximum gray level.

Otsu's method determines the threshold value based on the statistical information of the image where for a chosen threshold value $t$ the variance of clusters $T_0$ and $T_1$ can be computed. The optimal threshold value is calculated by minimizing the sum of the weighted group variances, where the weights are the probability of the respective groups [10].

Let $p(i)$ be the histogram probabilities of the observed gray value with $i = 1, ..., l$ :

$$P(i) = \frac{number\{(r,c)|image(r,c) = i\}}{(R,C)} \tag{2.1}$$

Where $r$, $c$ are the indexes for row and column of the image, respectively,

while $R$ and $C$ are the number of rows and columns of the image. Let $w_b(t)$, $\mu_b(t)$, $\sigma_b^2(t)$ and $w_f(t)$, $\mu_f(t)$, $\sigma_f^2(t)$ be respectively the weight, mean, and variance of the classes $T_0$ and $T_1$, the best threshold value, determined by the Otsu's method, $t^*$ is the value with the minimum within class variance defined as follow [10]:

$$\sigma_w^2 = w_b(t) * \sigma_b^2(t) + w_f(t) * \sigma_f^2(t) \tag{2.2}$$

### 2.1.2 Physical structure recognition

The physical structure recognition phase aims at decomposing a printed document into a sets of homogenous regions, such as images, graphics, and text blocks. Methods for the physical structure analysis fall roughly into three categories, depending on their approach: bottom-up, top-down, hybrid.

**Bottom-up** This family of algorithms proceed by merging low-level document primitives, i.e. pixels or connected components, in order to generate low-level document structures and form larger and homogeneous regions (Figure 2.2 ). The key advantage of bottom-up methods is that they can handle arbitrarily shaped regions. The drawback is that is tricky to take into account higher-level structures in the image, such as columns, and this often leads to over-fragmented regions (Figure 2.3).

Wong [14] in his research presents a method called Run Length Smoothing Algorithm (RLSA). This method is used as the first of a two steps document analysis system and it aims at subdividing the area of a document into regions, each of which should contain only one type of data. The RLSA is applied to a binary sequence in which white pixels are represented by zeros

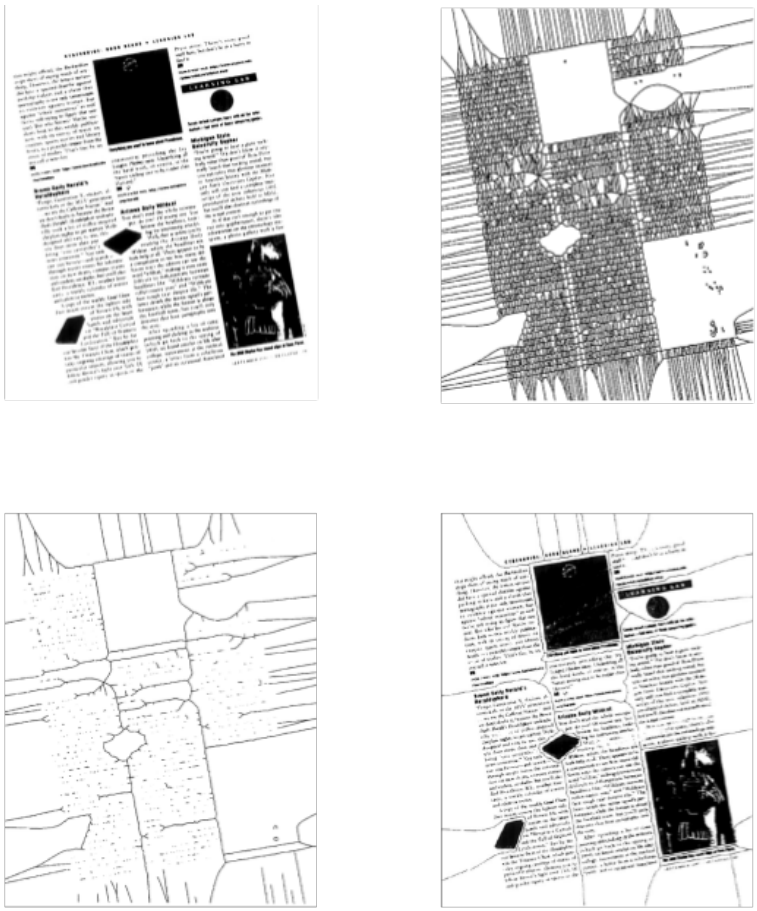

*Figure 2.2: Example of bottom-up algorithm execution.*

and black pixels by ones and it has the effect of linking together neighboring black areas that are separated by less than C pixels. The algorithm is applied row-by-row and column-by-column, yielding two distinct bit-maps, these are then combined in a logical AND operation (Figure 2.4).

There are certain limitations to the block segmentation described above, the main is the use of parameters an this on different documents can leads on text lines linked together to small line-to-line spacing compared to the used distance parameter [14].

Another existing possibility is to process the original image through some

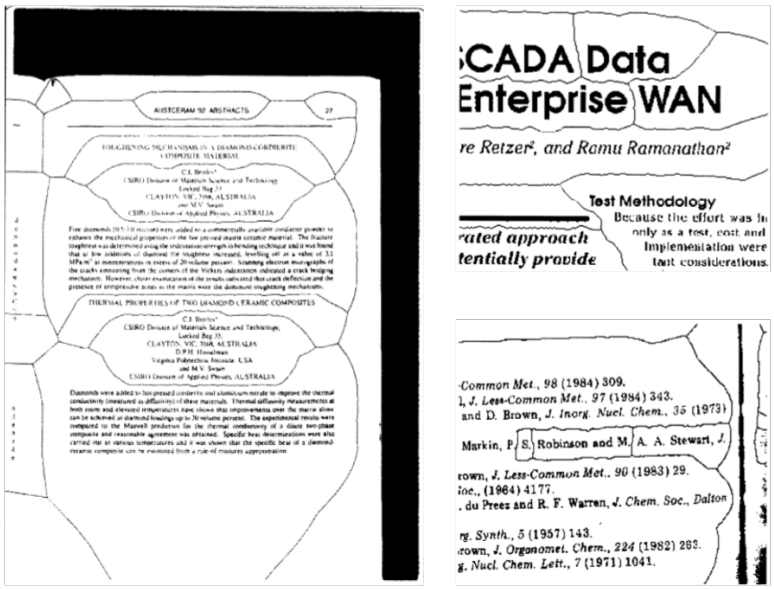


*Figure 2.3: Examples where bottom-up algorithms success, left, or fail, right. We can see at the top-right an over-fragmented title due to wide interword gaps, and at the bottom-right words fragmented from the body text region*

region filters. In [15], O'Gorman presents a system that does not use the connected components homogeneity as criterion any more, but it clusters connected components by analyzing their neighborhood relationships. To perform this he make use of Docstrum; it is a representation of the document page that describes global structural features of the page and which can be used for page analysis. It is based on k-nearest-neighbor clustering of connected components of the page. The docstrum spectrum is the plot of the tuple of each nearest-neighbor pairs in the document. It is a polar plot with origin at the center. Orientation and text line information can be determined directly from clusters on the docstrum plot (Figure 2.5).

In this paper [16] instead, Kise, proposed a method of page segmentation for document with non-Manhattan layout. For page with such type

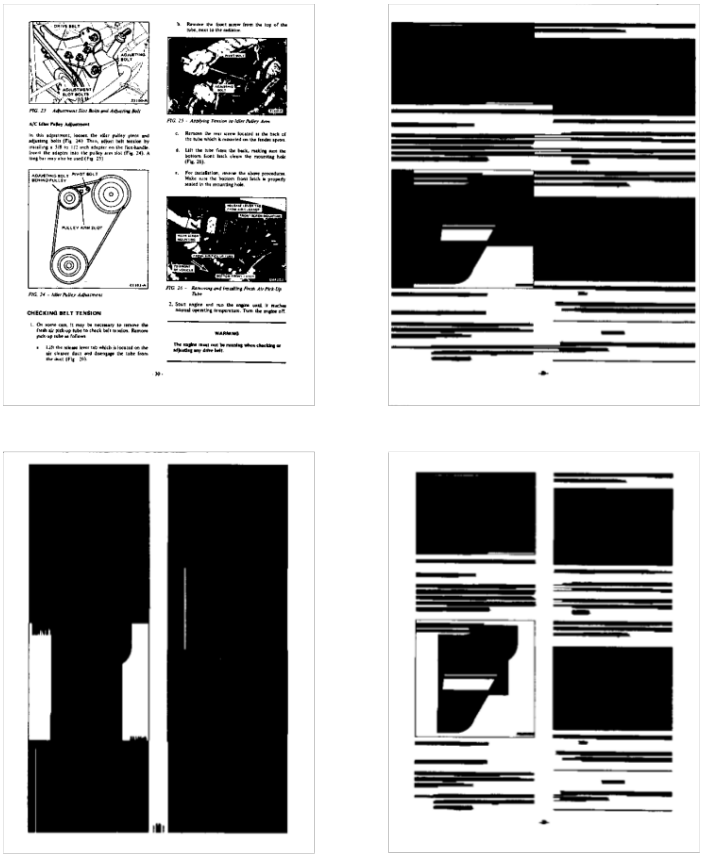

*Figure 2.4: Examples of application of the RLSA technique*

of layout, boundaries of document components cannot be represented by vertical and horizontal line segments; line segments of arbitrary orientation and length need to be utilized. In order to obtain candidates efficiently, a Voronoi diagram, which represents the neighborhood of connected components as polygons, was employed. Based on this representation, the process of page segmentation can be considered as the selection of the appropriate line segments as boundaries of document components. For this purpose, he utilizes two characteristic features: the Euclidean distance and the area ratio calculated from a pair of connected components that are neighbors in the Voronoi diagram. Intercharacter and interline gaps are then esti-

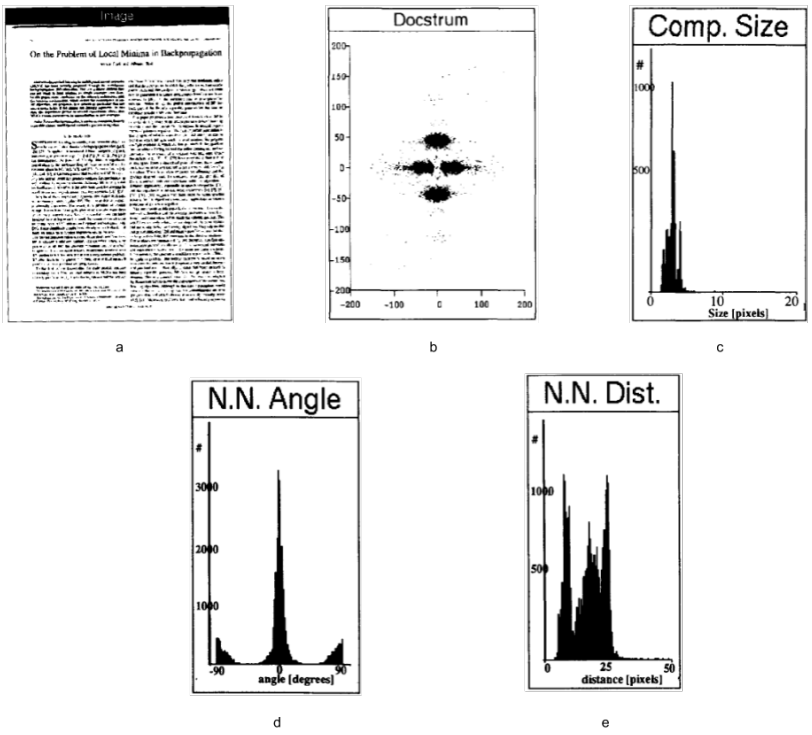


*Figure 2.5: (a) An image used for example, (b) the docstrum plot of the image, each point represents the distance and angle between a nearest-neighbor pair, (c) (d) (e) information on orientation and text lines extracted from the docstrum plot.*

mated from the Voronoi diagram to determine adaptively the thresholds of the charachters and lines distance.

**Top-down** This kind of algorithms start analyzing the complete document image and try to decompose it iteratively until homogeneous regions are detected (given a predefined homogeneity criterion). Top-down methods have the advantage that start by looking at the largest structures on the page, however, they are unable to handle the variety of formats that occur in documents having complex layouts, such as magazines or newspapers.

Antonacopoulos [17] in his research focus on page segmentation using image background. He segments the regions by identifying the background space and describing them by white tiles. These describe the background completely, ensuring that the shape of white spaces is preserved, then the contours of the printed regions can be constructed by selecting, for each region, the appropriate edges of the white tiles that border with that region. Indeed he exploited the streams of white space that surrounds the printed regions in a page. This is a typesetting convention [17] used in practically all documents. The whitespace so plays the dual role of both a delimiter of each printed region and separator between adjacent regions. Thus this delimiting space can be visualized as an irregular white net, with the printed regions thought of as the holes and the objective of this method is to reconstruct the irregular net of background space in order to identify the precise contours of the printed regions.

Jaekyu Ha, in his research [18], uses a modified version of the technique known as recursive X-Y cut to decompose a document image recursively into a set of rectangular blocks. This version of the method implements the recursive X-Y cut using bounding boxes of connected components of black pixels instead of using image pixels, bringing to an advantage in term of computational time. In fact, document page decomposition is accomplished by analyzing and handling the spatial configuration of some important feature of connected components called bounding boxes. To decompose the document, local peaks and valleys of the projection profiles are used to determine the location where the division has to take place. The projection profile of a set of bounding box is a histogram which shows frequencies of projected bounding boxes on a axis. At each step, the pixel-projection profiles are

calculated in both horizontal and vertical directions. Then a zone division is performed at the most prominent valley in either projection profile, and the process is repeated recursively until no sufficiently wide valleys are left in both profiles.

**Hybrid** Hybrid approaches try to combine the best of bottom-up and top-down methods.

Smith, in [19], proposes a hybrid page layout analysis algorithm, which uses bottom-up methods to form an hypothesis and locate the tab-stops that were used when the page was formatted. The detected tab-stops, are then used to deduce the column layout of the page. Indeed regions of a page are bounded by tab-stops and margins, column edges, indentation, and columns of a table are all placed at fixed x-positions at which edges or centers of text lines are aligned vertically. Tab-stops distinguish tables from body text, and they also bound rectangular non-column elements, such as images and quotes.

Azokly in [20] presents a system combining both ascending and descending strategies in two levels: the first level takes care of microstructure recognition using rules developed to model the usual graphical aspect of microstructures and bases the recognition on the connected component classification. The analysis of textual blocks is realized by a hierarchically merging method while tables are analysed by a hierarchically splitting method. The second level manages the macrostructure recognition segmenting a document into regions and regions into blocks, the two segmentation phases are respectively guided by a description of the document class and by a hierarchically split-

ting method.

Tran in his research [6], proposes an hybrid method to separate the input document into two binary documents, text document and non-text document. In its system firstly, the colored input document is binarized by the use of the Sauvola technique [21]. After that, on the binary document, connected components and relatives properties are extracted and used to classify text and non-text elements with minimum homogeneity algorithm. This method is the combination of connected component analysis and multilevel homogeneity structure. Then, in the second stage, a new homogeneity structure is combined with an adaptive mathematical morphology and with the whitespace analysis in the text document to get a set of text regions. After reshaping all text elements by their coordinates, the non-text document can be obtained by the subtraction of the original binary image and the text document. It follows a process of refinement consisting in region and noise detection process in which all regions, both in the text document and non-text document, are refined to eliminate noise and get the final geometric layout.

### 2.1.3 Logical structure recognition

The last step of document parsing is logical structure recognition(see figure 2.1). Logical structure recognition is performed on the results of the physical structure recognition and it aims to assigning logical labels to the physical objects recovered during the previous phase. The labeling phase proceeds in two phases; i.e., features extraction and classification, and in literature they differ by the extracted features and the classification method.

**Feature extraction** Lots of features may be extracted from document images, these can be divided in two categories: those related to the intrinsic properties of a block, and they are called self-related attributes, and the features that use neighborhood relations or the position of a block in the page, and these are called cross-related attributes. Self-related attributes define only the intrinsic features of a block. These might consist in morphological features, such as the block dimension, elongation (width/height ratio), density (black/white pixels ratio), in structural features that describe textual blocks, such as number of lines, and typographical attributes such as justification (left, right, centered, justified), interline, font size, font name, font face (bold, italic, underlined) or in textual features consisting of a set of words or regular expressions in order to detect precise entities.

Wang and Srihari [22] based theirs classification approach on statistical textural features and feature space techniques. Morphological features are used to build two matrices, whose elements are the frequency counts of black-white pair run lengths transitions and black-white-black combination run lengths The features extracted from the matrices are used to determine a feature space in which block classification is accomplished using linear discriminant functions.

Rangoni [23], in order to deal with knowledge and learning, propose an artificial neural network (ANN) that combines morphological, structural, and textual features as input features and uses a Transparent Neural Network architecture (TNN). The TNN is a computational model that simultaneously contain two previously developed models of human reasoning: one for deductive reasoning about propositional logic, the other for inductive rea-

soning about number sequence problems. TNN is a restricted type of ANN, designed with the aim of facilitating human understanding and composed of a set of labeled nodes and a cycle-free relation, whose elements are called connections [24].
This method can improves the reasoning by introducing knowledge; indeed this system can be considered as a hybrid method between a model-driven (i.e. DTD integration) and data-driven approach (i.e. training phase). The recognition task is done by the forward step while backpropagation is used for an input correction process comparing it as the acts of human perception.

Cross-related attributes express features extracted from relations between blocks, and they are divided in:

- geometrical relations: often used as features where these can be qualified through their coordinates;
- textual relations: that specify the absolute or relative number of common words between blocks, by a predefined list of potential common words that need to be located in the related blocks;
- label relations: which enables a block to be qualified through other specific labels, this may be very useful, particularly in regard to document objects that are depending on each other but this relations imply an heuristic approach since the logical label is the researched information.

Azokly [20] exploiting the fact that in every culture there are conventions that steer the layout, e.g. in west countries writing goes from left to right and from top to bottom, presents a systematic study of the graphic appearance of the microstructures. This microstructures are recognized through a process guided by rules he has developed to model the microstructures usual

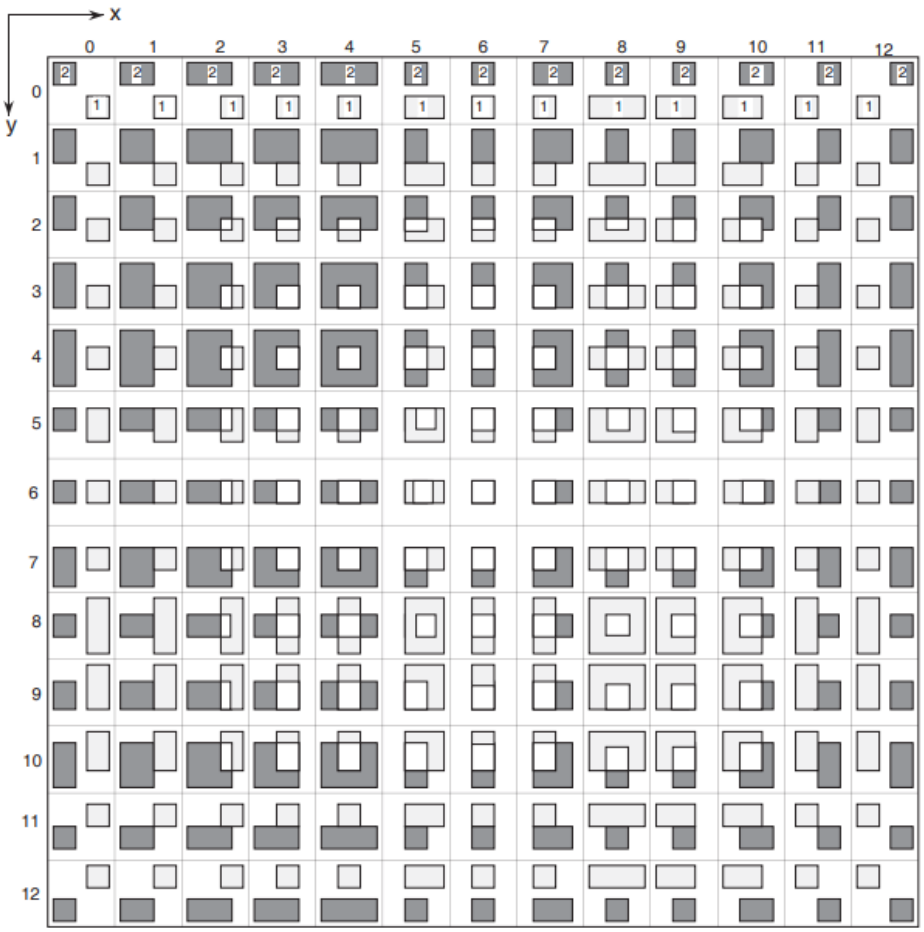


*Figure 2.6: The local topologies defined by Azokly, in his research 169 types of positions relatives between two blocks are presented*

graphical aspect. The recognition is based on a connected component classification, the analysis of textual blocks as well as mathematical expressions is realized by a hierarchically merging method while tables are analysed by a hierarchically splitting method. Through the study of the spatial relationships of two entities, in the XY plane, he identified a total of 169 possible topologies illustrated in Figure 2.6.

**Classification** The purpose of the classification phase is to assign labels to the previously segmented blocks. This task is performed by a classifier, i.e., a system that uses the features extracted from a block in order to classify it in a predefined class. A classifier needs a certain amount of knowledge about the analysed document in order to label each block; depending on the complexity of the documents, the extracted features, and the purpose of the recognition system, the classification methods may be different.

Cinque [23], after creating a set of four feature maps and a set of Gaussian pyramids (one for each feature map) satisfying the constraint that the segmentation process must be independent of a priori knowledge about the physical structure of the documents, uses a iterative process. For the classification he use a sets of rules such as "a heading has to be composed of at most two rows and must not exceed one quarter of the vertical dimension of the page" to label text blocks.

More elaborated classifiers can be trained in order to automatically infer their knowledge. Indeed, features extracted from a block are commonly represented as vectors in a multidimensional space. The classification then can consists in partitioning the feature space into classes corresponding to predefined block labels. As Heroux [25], that computes the mean vector of morphological features for each class and successively classifies matching each block vector with the class of the nearest vector.

### 2.1.4 International Conference on Document Analysis and Recognition (ICDAR)

There are not many opportunities for comparison between the various algorithms that deal with document analysis; ICDAR is the premier international

conference, ranked by CORE as a conference of category “A”, for researchers and practitioners in the document analysis community for exchanging ideas on the state-of-the-art technology in document analysis, understanding, retrieval, and performance evaluation. ICDAR is held every two years, since 1991, in a different city indeed the host country changes every time and the conference has taken place on four different continents so far. The term document in the context of ICDAR encompasses a broad range of documents from historical forms such as palm leaves and papyrus to traditional documents and modern multimedia documents. ICDAR deals with eight areas of interest: character and symbol recognition, printed/handwritten text recognition, graphics analysis and recognition, document analysis, document understanding, historical documents and digital libraries, document based forensics and camera and video based scene text analysis [26].

In the context of document analysis the competition ”ICDAR Competition on Recognition of Documents with Complex Layouts” presents challenges for page segmentation, region classification, and text recognition in an end-to-end scenario. The dataset to be used in this competition is a subset of the publicly available ”PRImA Layout Analysis Dataset” [5], a dataset created for the evaluation of layout analysis methods. It contains realistic documents with a wide variety of layouts, reflecting the various challenges in layout analysis and particular emphasis is placed on magazines and technical/scientific publications. Each image in the dataset has associated detailed ground truth enabling in-depth evaluation.

For that concern the evaluation the performance analysis method can be divided into three parts. Firstly, all regions are transformed into an interval representation, which allows efficient further processing. Secondly, correspondences between ground truth and segmentation result regions are

determined. Finally, errors are identified, quantified and qualified in the context of an application scenario. An evaluation profile is a set of weights and settings representing a specific evaluation scenario in the context of a document analysis application, several profile presets have been developed for frequently used evaluation scenarios, e.g. general recognition scenario, full text recognition scenario or images, graphics and charts scenario.

To evaluate the performance of our research we will compare with the winner of this competition.

## 2.2 AlexNet

State of the art convolutional neural network have achieved human-like performance in several recognition tasks such as: handwritten character recognition, face recognition, scene labelling, object detection and image classification among others.

Krizhevsky [27] in his research declares that to learn about thousands of objects from millions of images, with machine learning methods, a model with a large learning capacity and also lots of prior knowledge about the data are needes.

Convolutional neural networks (CNNs) constitute a class of models whose capacity can be controlled by varying their depth and breadth, and compared to standard feedforward neural networks with similar sized layers, CNNs have fewer connections and parameters and so they are easier to train.

When the paper [27] was written, this network was one of the largest convolutional neural networks to date on the subsets of ImageNet [28], which is a larger datasets consists of over 15 million labeled high-resolution images

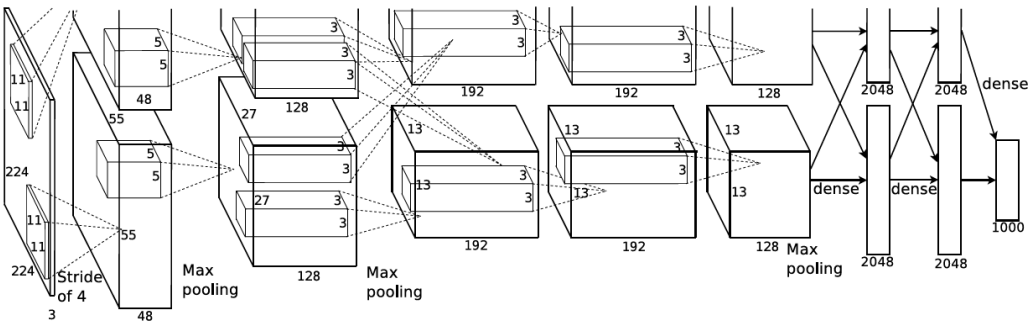


*Figure 2.7: Architecture of Alexnet*

in over 22,000 categories, and achieved the best results ever reported on these datasets.

The architecture of this network (Figure 2.7) contains eight learned layers, five convolutional and three fully-connected. The input image is resized to 244 x 244 x 3 and is filtered with 96 kernels of size 11 x 11 x 3 with a stride of 4 pixels. A second convolutional layer takes as input the output of the first convolutional layer and filters it with 256 kernels of size 5 x 5 x 48. The third, fourth, and fifth convolutional layers are connected to each other directly and respectively they have 384 kernels of size 3 x 3 x 256, 384 kernels of size 3 x 3 x 192 and 256 kernels of size 3 x 3 x 192. The fully-connected layers instead have 4096 neurons each.

Alexnet is one of the first to model a neuron's output as a non-saturating non-linear function, specifically they use Rectified Linear Units (ReLUs) [29]. Another important feature of this architecture is the parallelization, in fact half of the kernels are putted on each GPU, and the GPUs communicate only in certain layers, this means that, for example, the kernels of layer 3 take input from all kernel maps in layer 2. However, kernels in layer 4 take input only from those kernel maps in layer 3 which reside on the same GPU [27]. The latest meaningful technique is the dropout, it acts setting to zero the

output of each hidden neuron with probability 0.5, so that some neurons do not contribute to the forward pass and do not participate in backpropagation and every time an input is presented, the neural network samples a different architecture. This last technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons [30].
This network was tested on ILSVRC-2010 and achieves top-1 and top-5 test set error rates of 37.5 % and 17.0 % and on the Fall 2009 version of ImageNet with 10,184 categories and 8.9 million images and their top-1 and top-5 error rates on this dataset are 67.4% and 40.9%.

We explore this architecture considering that unlike the other convolutional networks available, it presents a simpler and less deep architecture. These features allowed us to use this network with not high computing power and presented good performances on the dataset we created, despite being small. In this work AlexNet is used as classification step of the different document regions.

## 2.3 You Only Look Once (YOLO)

Unlike the convolutional network described above, neural networks that perform detection are not only classifying but also precisely localizing objects of various classes. As part of our research, this has been studied because it would allow us to create an end-to-end system avoiding having to use heuristic parameters for the segmentation of documents [31].

Before YOLO [32], the state-of-the-art methods in the detection task

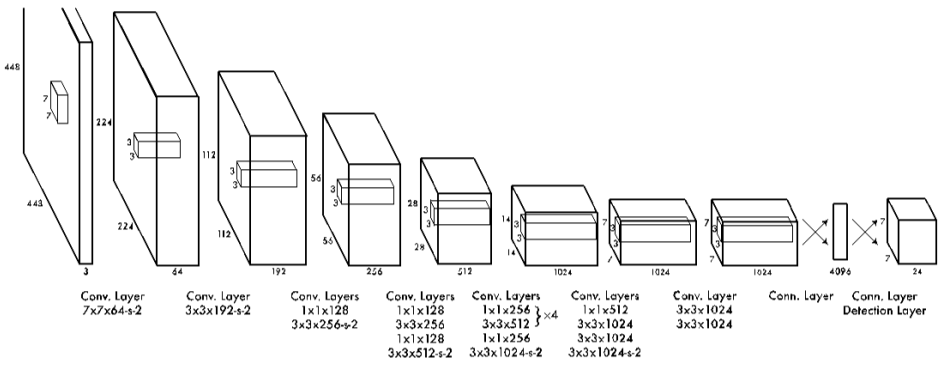


*Figure 2.8: Architecture of YOLO*

were mainly developed based on classifier based systems that divided detection problems into a pipeline of region proposal extraction, bounding box regression and classification. However, the multi-stage pipeline of detection makes for classifier-based methods hard to learn the required information with an end-to-end training and the detection speed hard to be improved. YOLO network was inspired by the fact that, in real world, human eyes are able to detect objects at one glance, so that it is more natural to treat the problem as a pure regression and directly predict bounding boxes as well as the corresponding classes and confidence levels with a single network.

YOLO network is inspired by the GoogLeNet model and, as shown in Figure 2.8, the structure consists of 24 convolutional layers and 2 fully-connected layers. YOLO divides resized images (of fixed size 448 x 448) into S x S grid cells. Each grid cell is responsible for the bounding box whose center is at the location of the grid cell, and predicts B bounding boxes, the confidence level and class probability. For a dataset with C class labels, the output tensor is S x S x (C + B x 5) and in the original paper, S, C and B are respectively 7, 20 and 2, so that the output tensor is 7 x 7 x 30. As a result, the network can produce at most S x S x B predictions, which are filtered by a threshold on confidence levels and a non-maximum suppression [32].

The loss function is also divided into 3 parts: the penalization for bounding box location accuracy, the penalization for the confidence level of an object's center located in a grid cell and the penalization for classification error. In this way, the model is regarded as a full pipeline and it can be trained end-to-end, which largely improves the speed both in training and testing.

This network was tested on VOC 2012 test set and scored 57.9% mAP, that is lower than state of the art on the paper publications at the time, for authors this was due to the presence of small objects. YOLO was also tested on PASCAL and result on the fastest object detection method with 52.7% mAP, more than twice as accurate as prior work on real-time detection [32].

## 2.4 Transfer learning

Transfer learning is a machine learning method which refers to the situation where what has been learned in one setting is exploited to improve generalization in another setting; this can therefore be considered an optimization that allows rapid progress and improve performance when modeling the second setting [33] [34].

Transfer learning is related to problems such as multi-task learning and the concept is not exclusively an area of study for deep learning. Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large datasets on which deep learning models are trained, e.g. if already exists a network that is pre-trained on a similar task and trained on massive amounts of data.

The usual transfer learning approach is to train a base network and then

copy its first n layers to the first n layers of a target network. The remaining layers of the target network are then randomly initialized and trained toward the target task.

There are two possible choices for what concern the backpropagation: the first is that the errors are backpropagated from the new task into the base features to fine-tune them to the new task, or the transferred feature layers can be left frozen, so that they do not change during training on the new task.

The choice depends on the size of the target dataset and the number of parameters in the first n layers: if the target dataset is small and the number of parameters is large, fine-tuning may result in overfitting, so the features are often left frozen, if the target dataset is large or the number of parameters is small, the overfitting is not a problem, the base features can be fine-tuned to the new task to improve performance, finally if the target dataset is very large, there would be little need to transfer because the lower level filters could just be learned from scratch on the target dataset [35].
There are three situations that suggest to use or not transfer learning (Figure 2.9):

- the initial performance achievable in the target task using only the transferred knowledge, is better compared to the initial performance of an ignorant agent;
- the amount of time it takes to learn the target task given the transferred knowledge is less than the amount of time to learn it from scratch;
- the final performance achievable is better compared to the final level without transfer.

However the choice of source data and the source model is still an open

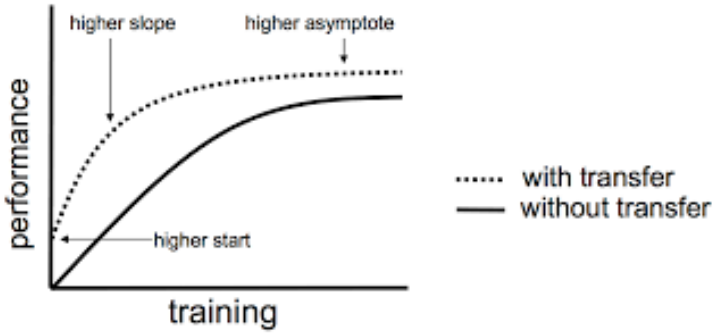


*Figure 2.9: The three situations in which transfer learning is useful.*

problem and may require domain expertise and intuition developed via experience.

# Chapter 3

# Document parsing

In this chapter we describe the document parsing system we designed as part of this thesis work. First of all we aimed at achieving the results obtained by the state of the art [6] by following their path, and then improve on them by adding a deep learning classifier for the classification of the extracted region so as to eliminate heuristics used by the authors in recognition.

Initially our idea was to exploit the convolutional neural network called *YOLO* [32] both for the segmentation and for the classification of the different parts of the document, i.e. text, images, tables and graphs. This approach did not give good results (Section 4.5) in fact the neural network did not reach the desired performances. We then focus on improve the document segmentation pipeline in [6] making use of the convolutional neural network called *AlexNet* [27], trained to classify text, images, tables and graphs through the dataset we created (Section 4.2), for the classification phase instead of using the heuristics proposed by the authors.

We describe an overview of the overall system in the Section 3.1 and, in the next sections, the detailed explanation of each phase of the system.

## 3.1 System overview

The input of the proposed system is a color, or a binary, image obtained via a scanning process or directly a digital document. The output of the proposed system is a set of separated regions with their labels and coordinates. Let's $f$ the input image, the analysis of the document layout proceeds as it follows (Figure 3.1):

- **Step 1**: binarization of the document image by Otsu algorithm (Section 3.2).
- **Step 2** : connected component analysis (Section 3.3).
- **Step 3** : heuristic filter (Section 3.4).
- **Step 4** : recursive segmentation (Section 3.5).
- **Step 5** : white space analysis and median filter (Section 3.6).
- **Step 6** : post processing and noise removal (Section 3.7).

After this first set of steps we obtain the document connected components separated in two groups: the first group includes the text components while the second non-text components. From the set of text components we obtain the new document $f_T$ and on that we execute the next steps:

- **Step 7**: text region segmentation (Section 3.8).

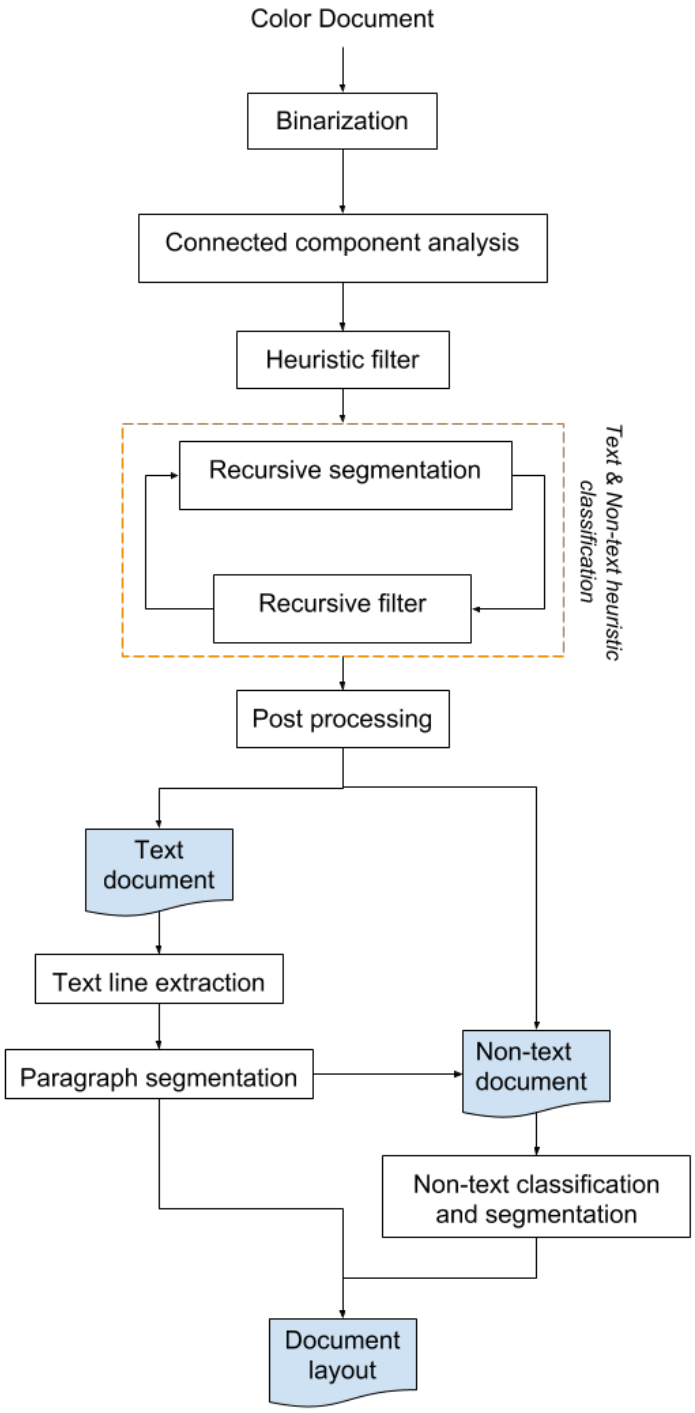


*Figure 3.1: The flowchart of our method*

The output of this phase is a set of segmented text zones according to heuristic rules; on these zones we apply the convolutional network, to confirm or correct their class. After that we subtract the classified text zone from the input document image and we obtain the non-text document $f_{NT}$ and we compute the last steps:

- **Step 8**: morphological transformation (Section 3.9).
- **Step 9**: recursive segmentation (Section 3.5).
- **Step 10**: zone classification (Section 3.10).

## 3.2 Preprocessing

Preprocessing is the first step after reading a document image into memory. The aim is to prepare the document for text and graphics separation. Our method, in its first part, exploits connected components. Each component can be either a character, punctuation, noise, part of a handwritten word, rule line or graphical element.

To extract these components we first apply binarization, and to do so we use a global thresholding and the Otsu binarization. We assume that in the images of documents the background is clearly separated from the foreground so we can infer the input images as bi-modal.

A bi-modal image is an image whose histogram of the pixel intensity values, i.e., the graph showing the number of pixels in an image at each different intensity value found in that image, can be clustered around two well-separated values.

To exploit this peculiarity and standardize the algorithm, the first operation performed on the input image is the color-space conversion from the


*Figure 3.2: On the left the original document, on the right the binarized document through the Otsu method*

BGR space to the gray space. After that we use the Otsu's method to automatically reduce the gray level image to a binary image, the Otsu's method calculates the optimum threshold separating the two classes so that their combined intra-class variance, defined as the weighted sum of the variances of the two classes is minimal. Weights, computed from the bins of the histogram, are the probabilities that the classes are separated by a threshold. The output of this phase is then the binarized document $f$ (Figure 3.2).

## 3.3 Connected component analysis

In this second step we aim to extract and analyze the connected components present in the binary document resulting from the first phase. Connected components labelling is the process that extracts all the pixels that are connected and have the same value; it assigns these to a separate components.

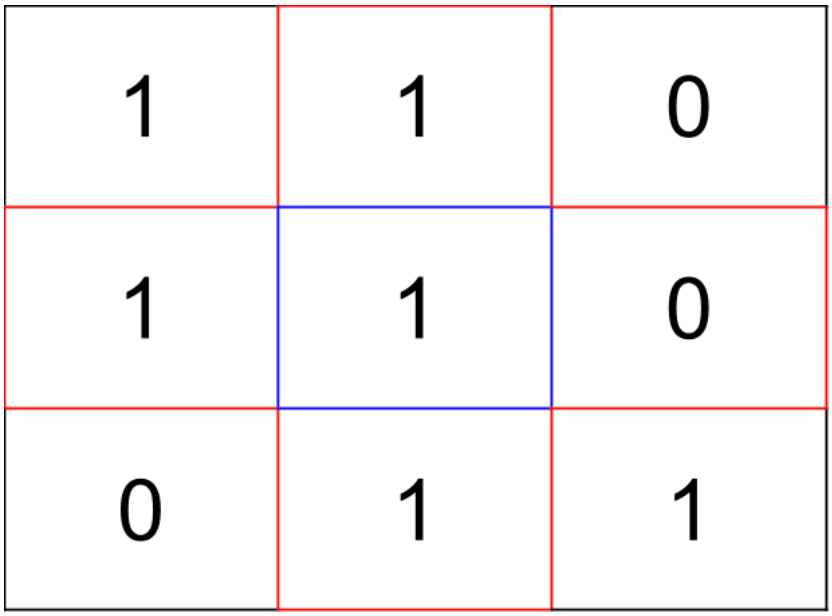


*Figure 3.3: An example of the border following algorithm; the examined pixel, with blue borders, is a border pixel because his value is 1 and among its 4-connected pixels, with the red borders, there is a 0.*

The technique we used is called border following, it extracts a sequence of coordinates from the border between a connected component of 1s and a connected component of 0s (background), where the pixels with values 0 and 1 are called the 0-pixel and the 1-pixel, respectively.

It works considering that in the 4-connected case, also known as a direct-neighbor case, if a pixel with value 1 has a 0-pixel in its 4 neighborhood, i.e., when the 1-pixel shares an edge with a 0-pixel, is called a border point and this is used for find the components [36] (Figure 3.3).

Once we have extracted the connected components we start the analysis to obtain, for each component, the self related attributes. Let $L_{cc}$ be the list of all the connected the components, and be $CC_i$ is the $i_{th}$ connected component of $f$. For every $CC_i$ we compute and associate the following set

of features:

- B(CCi) : is the bounding box, i.e. the smallest rectangle which circumscribes the connected component, of $CC_i$ stored as $X_{min}$, $Y_{min}$, $X_{max}$, $Y_{max}$, respectively the leftmost, the topmost, the rightmost and the lowermost coordinate.

- $CC_{area}(CC_i)$: is the area of the $CC_i$, that is the number of non-zero pixels.

- $B_{size}(CC_i)$: is the area of $B(CC_i)$, computed as:

$$B_{size}(CC_i) = (X_{max} - X_{min}) * (Y_{max} - Y_{min}) \tag{3.1}$$

- $CC_{dens}(CC_i)$: is the ratio between the $CC_{area}(CC_i)$ and the $B_{size}(CC_i)$, computed as:

$$CC_{dens}(CC_i) = \frac{CC_{area}(CC_i)}{B_{size}(CC_i)},\ CC_{dens} \in (0, 1] \tag{3.2}$$

- $CC_{ratio}(CC_i)$ : is the aspect ratio, that in a rectangle is the ratio of its longer side to its shorter side, of the bounding box of the $CC_i$ , computed as:

$$CC_{ratio}(CC_i) = \frac{\min(Wi,\ Hi)}{\max(Wi,\ Hi)},\ CC_{ratio} \in (0, 1] \tag{3.3}$$

  where:

$$W_i = X_{max} - X_{min},\ and\ H_i = Y_{max} - Y_{min} \tag{3.4}$$

After calculating and storing this features for each $CC_i$ we calculate also the cross-related attributes, so for every components we calculate also:

- $CC_{same\_column}(CC_i)$: are the list of components that stay on the same column of $CC_i$ computed as:

$$CC_{same\_column}(CC_i) = \{CC_j \in L_{cc} \mid \max(X_{min_i}, X_{min_j}) - \min(X_{max_i}, X_{max_j}) < 0\} \quad (3.5)$$

- $CC_{same\_row}(CC_i)$: are the list of components that stay on the same row of $CC_i$ computed as:

$$CC_{same\_row}(CC_i) = \{CC_j \in L_{cc} \mid \max(Y_{min_i}, Y_{min_j}) - \min(Y_{max_i}, Y_{max_j}) < 0\} \quad (3.6)$$

- $CC_{ins}(CC_i)$: is the list of $B(CC_j)$, $i \neq j$ located inside the $B(CC_i)$. These are computed as:

$$CC_{ins}(CC_i) = \{CC_j \in L_{cc} \mid X_{min_i} < X_{min_j} \wedge X_{max_j} < X_{max_i} \wedge Y_{min_i} < Y_{min_j} \wedge Y_{max_j} < Y_{max_i}\} \quad (3.7)$$

For a later analysis we extract in this step also (Figure 3.4):

- $CC_{RN}(CC_i)$: is the list of the right neighbors of $CC_i$. To compute this we first order the $CC_{same\_row}(CC_i)$ with respect to the $X_{min}$ and after that the right neighbors are these that satisfied:

$$CC_{RN}(CC_i) = \{CC_j \in CC_{same\_row}(CC_i) \mid (X_{min_j} - X_{max_i}) > 0\} \quad (3.8)$$

- $CC_{LN}(CC_i)$: is the list of the left neighbors of $CC_i$. We first order the $CC_{same\_row}(CC_i)$ with respect to the $X_{max}$ and after that the right

neighbors are those satisfying:

$$CC_{RN}(CC_i) = \{CC_j \in CC_{same\_row}(CC_i) \mid (X_{min_i} - X_{max_j}) > 0\} \tag{3.9}$$

- $CC_{RNN}(CC_i)$: is the right nearest neighbor of CCi . Starting from the $CC_{RN}(CC_i)$, we compute this as:

$$CC_{RNN}(CC_i) = \begin{cases} |\min(X_{min_j})| & if\ CC_{RN}(CC_i) \neq 0 \\ -1 & elsewhere \end{cases}$$
$$with\ CC_j \in CC_{RN}(CC_i) \tag{3.10}$$

  If $CC_{RNN}(CC_i)$ is equal to -1 $CC_i$ is the rightmost component in its row.

- $CC_{LNN}(CC_i)$: is the right nearest neighbor of CCi . Starting from the $CC_{LN}(CC_i)$, we compute this as:

$$CC_{LNN}(CC_i) = \begin{cases} |\max(X_{max_j})| & if\ CC_{LN}(CC_i) \neq 0 \\ -1 & elsewhere \end{cases}$$
$$with\ CC_j \in CC_{LN}(CC_i) \tag{3.11}$$

  If $CC_{LNN}(CC_i)$ is equal to -1 $CC_i$ is the leftmost component in its row.

## 3.4 Heuristic filter

The use of heuristic conditions or threshold values is common in image processing. These are used to speed up processing and prune some individual cases. Heuristics conditions must be precise and strict because they have a strong influence on the precision of the successive steps of the algorithm.

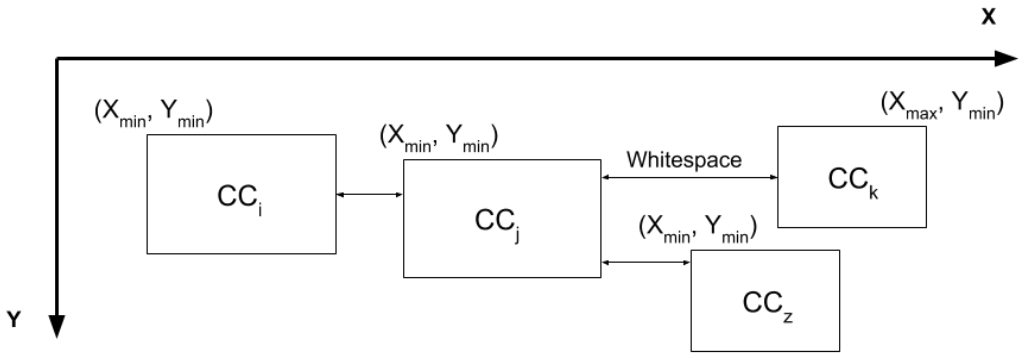


*Figure 3.4: Example of connected component analysis.* $CC_{RN_j} = \{CC_k, CC_z\}$ *while* $CC_{RNN_j} = CC_z$. $CC_{RN_k} = \emptyset$, *so* $CC_{RNN_k} = -1$. *In the same way* $CC_{LN_i} = \emptyset$, *so* $CC_{LNN_i} = -1$.

Through the heuristic filter we look for the components, extracted in Section 3.3, that cannot be text and we begin to separate components into text and non-text.

This filter not only reduces the computational time of the whole process, and at the same time increases the accuracy of the proposed system.

Let $L_{CC}$ be the set of all the connected components extracted in the previous step and let $CC_i$ the $i_{th}$ connected component, if it satisfied one of the following conditions we insert it in the list of non-text component, $L_{NT}$, and eliminate it from $L_{CC}$. A component is considered a non-text component if:

- $CC_{area}(CC_i) < T_{area}$ where $T_{area} = 6$; we remove the components that have tiny area, they are probably noise and are very difficult to identify even for human eyes.

- $CC_{dens}(CC_i) < T_{dens}$ where $T_{dens} = 0,15$; when the density of $CC_i$ is too low, it can be a diagonal component, noise element, separators, or rectangular shape. In fact the normal density of text element according

to [6] is greater than $0,2$.

- $CC_{ratio}(CC_i) < T_{ratio}$ where $T_{ratio} = 0,06$; the ratio between the height and the width is not too low, in fact is not possible that a dimension of a text component is $16,66$ times larger than the other.

- $CC_{ins}(CC_i) < T_{ins}$ where $T_{ins} = 4$; if the bounding box of $CC_i$ contains more than four other components, we regarded it as a non- text element. This value is optimized by an empirical analysis on letters from different languages [6].

At the end of this process we assumed all the component in $L_{CC}$ are text element so we call them $L_T$, however, it may still contain some non-text element to be identified in the next steps.

From the binary document $f$ we obtain a new document $\hat{f}$ (Figure 3.5), as:

$$\hat{f} = \begin{cases} f & if\ f(x,\ y) \in L_T \\ 0 & if\ f(x,\ y) \in L_{NT} \end{cases} \tag{3.12}$$

We set this as the root of our regions tree and proceed to the next step.

## 3.5 Recursive segmentation

A document is usually divided into various regions, including text regions (paragraph), image regions, tables, and graphs. The aims of this step is to separate recursively a region R in a set of smaller homogeneous sub-regions $R_i$ such that:

$$R = R_1 \cup R_2 \cup ... \cup R_N \tag{3.13}$$

FIRST

future. According to Dealogic, 87 SPACs have begun trading in the U.S. since the end of 2003, buying some well-known companies like smoothie purveyor Jamba Juice. Last year alone, 40 SPACs worth $3.4 billion were announced, up from $484 million two years earlier.

SPACs are essentially shell companies. They go public with little more to show investors than a management team and an agreement that the money raised will be used to fund an acquisition in a particular sector, such as retail, or in an emerging market like China. Ergo, a SPAC is a roll of the dice. "Imagine paying $50 to go to a Broadway show, and you have no idea what's behind the curtain," says lawyer Mitchell Littman of Littman Krooks LLP, who works with a lot of SPACs. "You are relying on the fortune and integrity of the management team." Once a deal is completed, the SPAC's managers (who typically receive 20% of the public shares as compensation) are free to sell their holdings, usually after a lockup period.

Today's SPAC boom harkens back to the flurry of "blank check" companies that sprang up in the 1980s. Those were widely discredited by a wave of scams in which fraudsters would take a shell company public, announce a merger, pump the stock, and then dump it before everyone realized that the hot target company was anything but. Investors have a lot more protection with SPACs, which must hold almost all the money in escrow until a deal is done.

From Charney's perspective, there is no downside to the arrangement. It lets him raise cash relatively quickly, remain the largest shareholder, and avoid having to sell other investors on his vision until after his company is already publicly traded. "I'm in a hurry to get the money," he says, citing his plan to expand beyond his current 145 stores. "What's great about the SPAC is, it's kind of in reverse. First you get publicly traded, then you do the compliance stuff. And if you brought in private equity, they might want to run the company in a more [cautious] manner," he says, referring to American Apparel's risqué image.

Keeping control is particularly important for a guy who is the creative force behind the brand, as well as one whose flamboyant management style—there is one sexual-harassment suit outstanding against him by a former employee; two others have been dismissed—might scare off many investors. "He is very talented," says former J.C. Penney CEO Allen Questrom, a partner in private-equity firm Lee Equity Partners, "but he is a *meshugana* [read: nutty] type guy. It's very difficult to see him in a public sphere." (Charney, for his part, dismisses the talk about his reputation as "all tabloid.")

**FIRST YOU GET PUBLICLY TRADED, THEN YOU DO COMPLIANCE.**

Charney's new bedfellows don't appear concerned, even if they're more likely to wear dark suits than American Apparel's colorful leggings. Endeavor's president, Jonathan Ledecky, is familiar with unusual financing vehicles; he founded U.S. Office Products, the once-hot roll-up that went bankrupt in 2001 (long after Ledecky cashed out). Boldfaced names on the board include Kerry Kennedy, Bobby's daughter, and Edward Mathias, a managing director at the Carlyle Group. On Jan. 31 the same group formed a new $250 million SPAC, Victory Acquisition, underwritten by Citi. The filing notes that it could compete directly with Endeavor. And as with Endeavor, once Victory goes public it will pay a Ledecky "affiliate" $7,500 monthly to run its Manhattan office.

**SPAC Attack!**

**The volume of special purpose acquisition companies going public in the U.S. is rising sharply.**




Hedge funds are drawn to these investments. Since SPACs typically must announce a deal within 18 months of their IPOs or return the money raised, with interest, hedgies see them as safe places to park cash while potentially profiting from the price fluctuations between a deal's announcement and its close—a sort of arbitrage play. That may be why Steven Cohen's $12 billion hedge fund SAC Capital snapped up 8.7% of Endeavor's stock when the deal was announced.

Even if investors are SPAC-happy, not all regulators are: So far only the American Stock Exchange lists SPACs, and many bankers and lawyers say the SEC is deliberately taking its time vetting deals, hoping to slow down the pace of offerings. SPACs are probably best left to those, like Charney, who like to live life on the edge. F

**UNHEALTHY RETURN** Since smoothie chain Jamba Juice went public in November via a SPAC, its shares have fallen 19%.

FORTUNE CHART SOURCE: DEALOGIC PHOTO: DAVID MCNEW—GETTY



*Figure 3.5: The $\hat{f}$ document obtained applying the heuristic filter. The filter removes very small or very large components according to the heuristics. All the component that are not filtered, show on this image, are considered text components and further filtered in the next steps.*

until all the sub-regions satisfy the homogeneity criterion, i.e., until the variance in $R_i$ is under the threshold value.

For each region, supposed as rectangulare and with $a$ $x$ $b$ as size, we collect the following features:

- $B(R)$: is the bounding box of the region, namely the coordinates of the four vertex of the rectangle that contains the region, stored as $X_{min}$, $Y_{min}$, $X_{max}$, $Y_{max}$.
- $BW\_IMG$: is the binary portion of $\hat{f}$ surrounded by $B(R)$.
- $C_{list}$: list of all the components of $\hat{f}$ contained in the region.
- $H_{vertical}(R)$:is the vertical histogram of the pixels projection on the x axis computed as (Figure 3.6):

$$H_{vertical}(R) = \{h_x \mid h_x = \sum_{y=1}^{b} \hat{f}(x,\ y), 1 \leq x \leq a\} \tag{3.14}$$

  After that, the value of the histogram is converted to a bi-level value as (Figure 3.6):

$$H_{vertical}(R) = \begin{cases} 1 & if\ h_x > 0 \\ 0 & elsewhere \end{cases}, with\ h_x \in H_{vertical}(R) \tag{3.15}$$

- $H_{horizontal}(R)$:is the horizontal histogram of the pixels projection on the y axis computed as (Figure 3.6):

$$H_{horizontal}(R) = \{h_y \mid h_y = \sum_{x=1}^{a} \hat{f}(x,\ y), 1 \leq y \leq b\} \tag{3.16}$$

  After that, the value of the histogram is converted to a bi-level value as (Figure 3.6):

$$H_{horizontal}(R) = \begin{cases} 1 & if\ h_y > 0 \\ 0 & elsewhere \end{cases}, with\ h_y \in H_{horizontal}(R) \tag{3.17}$$

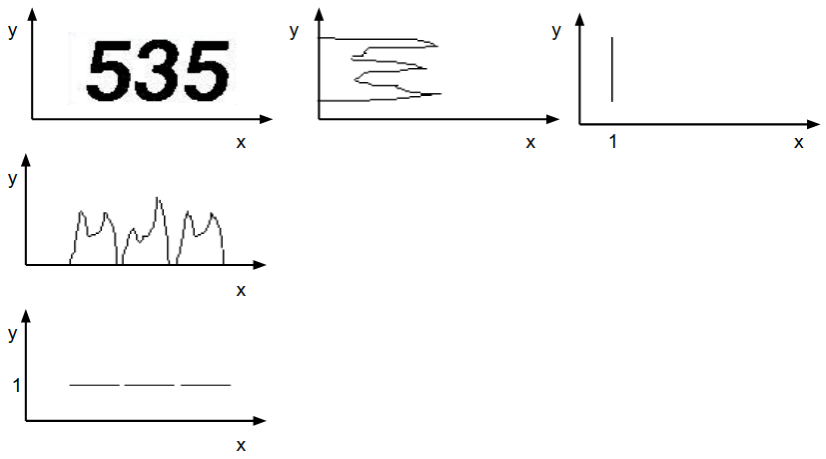


*Figure 3.6: Example of projection on the axis and trasformation on bi-level histrogram.*

After that we use the Run Length Encoding to calculate the run length of all elements in our histogram, either vertical or horizontal.

RLE is a data compressing method in which a sequence of data can be presented by his value and a counter; e.g., the result of RLE of sequence $\{1,1,1,0,0,1,1,0,0,0,1,1,1,1\}$ is $RLE = \{1,3,0,2,1,2,0,3,1,4\}$.

We calculated the RLE for the histogram than we separate the sequence of black and white lines.

- $L_{vertical\_black}$:

$$L_{vertical\_black} = \{b_i \in RLE_{vertical} \mid b_i > 0 \wedge b_{i-1} < 0\} \tag{3.18}$$

- $L_{vertical\_white}$:

$$L_{vertical\_white} = \{w_i \in RLE_{vertical} \mid w_i > 0 \wedge w_{i-1} = 0\} \tag{3.19}$$

- $L_{horizontal\_black}$:

$$L_{horizontal\_black} = \{b_i \in RLE_{horizontal} \mid b_i > 0 \wedge b_{i-1} < 0\} \tag{3.20}$$

- $L_{horizontal\_white}$:

$$L_{horizontal\_white} = \{w_i \in RLE_{horizontal} \mid w_i > 0 \wedge w_{i-1} = 0\} \quad (3.21)$$

So continuing with the previous example we obtain $L_{black} = 3, 2, 4$ and $L_{white} = 2, 3$

Finally for each sequences we calculate:

- *Median*: is the value separating the higher half of the data from the lower half. The advantage of the median in describing data compared to the mean is that it is not affected so much by extreme values.

  To find the median of a data set of n numbers we first sort these in ascending order than the index of the number representing the median is given by:

$$i = \frac{(n+1)}{2} \quad (3.22)$$

  This gives the middle number if we have an odd number of values, or the halfway point between the two middle values in the other case. So median can calculated by:

$$median(L) = \frac{L\lfloor i \rfloor + L\lfloor i + 0.5 \rfloor}{2} \quad (3.23)$$

  So for every sequence we have the corresponding median: $M_{LVB}$, $M_{LVW}$, $M_{LHB}$, $M_{LHW}$.

- *Variance* measures how far a set of numbers are spread out from their average value. Calling $\hat{\mathrm{I}}_4^1$ and n respectively the mean and the number of a sequence of black or white line, we can calculate the variance as:

$$Variance(L) = \frac{\sum(L_i - \mu)^2}{n} \quad (3.24)$$

  We computed then $V_{LVB}$, $V_{LVW}$, $V_{LHB}$, $V_{LHW}$.

In this algorithm we start analyzing the vertical homogeneity of the root of our regions tree, e.g. the whole page or a region to be further segmented after the recursive filter (Section 3.6), $R$, that is also the only leaf we have at this point.

**Step 1** **:** For the input region $R$, we evaluate the vertical variance of the consecutive line of black and white pixel and we can consider it not homogeneous if:

$$V_{LVB} = T_{variance} \; or \; V_{LVW} > T_{variance} \; where \; T_{variance} = 1,3 \tag{3.25}$$

according to [6] (Figure 3.7). If the region is homogenous we proceed analyzing the next unanalyzed leaf, setting $R = R_{next}$, conceding that there is at least one in the tree, otherwise we advance to the Step 4. Differently we proceed with the next step.

**Step 2** **:** If $V_{LVB} > V_{LVW}$ we split the document searching the maximum sequence of black vertical line else we search the maximum sequence of white vertical line 3.7):

$$Max_L = \begin{cases} \max(L_{vertical\_black}) & if \; V_{LVB} > V_{LVW} \\ \max(L_{vertical\_white}) & else \end{cases} \tag{3.26}$$

If $Max_L$ is greater, respectively, than $M_{LVB}$ or $M_{LVW}$ we split the region as follow in Step 3, differently, we return to the Step 1.

**Step 3** **:** In this step we split the region relying on the $Max_L$: we split the region in two or three sub-region according to the position of $Max_L$, in fact if is a line residing at the beginning or end of the dimension of the region, we have only two sub-region, otherwise if it is a line on the middle of the dimension of the region we split it in three different sub-regions 3.7).

We add this new regions, if $C_{list}$ has at least one component, to the regions tree as child of $R$, so we return to the Step 1.

At this point we have a tree which leaves are regions that satisfy the vertical homogeneity criterion. So we proceed, in the same way, analyzing the resulting regions with respect to the horizontal homogeneity criterion.

**Step 4 :** We take as input region, one at a time, all the leaves of our region tree, until there is at least one region not yet analyzed.

For each region, $R$, we evaluated if is homogenous or not based on:

$$V_{LHB} = T_{variance} \; or \; V_{LHW} > T_{variance} \; where \; T_{variance} = 1,3 \tag{3.27}$$

If the region is not homogenous we proceed to the next step otherwise we analyze the next region.

**Step 5 :** We find the maximum sequence of white or black lines with:

$$Max_L = \begin{cases} \max(L_{horizontal\_black}) & if \; V_{LHB} > V_{LHW} \\ \max(L_{horizontal\_white}) & else \end{cases} \tag{3.28}$$

and if $Max_L$ is greater, respectively, than $M_{LVB}$ or $M_{LVW}$ we proceed to split it as in Step 6 , otherwise we return to Step 4.

**Step 6 :** We split the region according to $Max_L$, and we obtain the horizontal homogeneous region as in Step 3. We add the region to the region tree only if $C_{list}$ contains at least one component. We then return to the Step 4.

When the process is finished we have a region tree that has as leaves the regions respecting both vertical and horizontal homogeneity criteria (Figure 3.8).

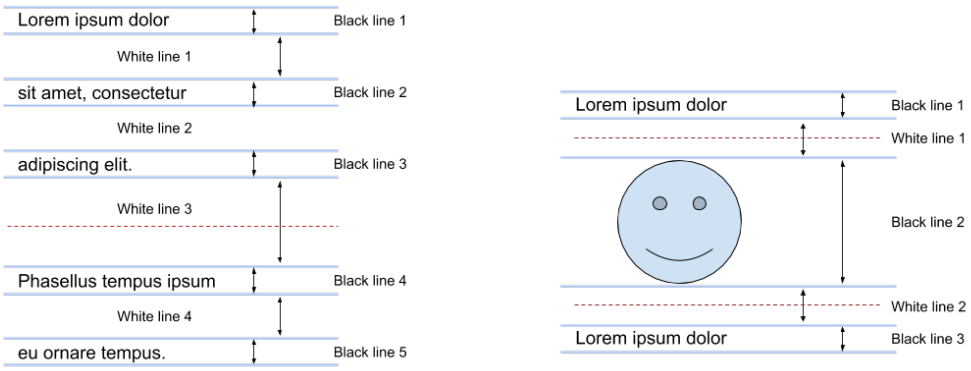


*Figure 3.7: Example of the application of the recursive segmentation step. On the left a region that has the $V_{LW}$ greater than $V_{L_B}$ and $T_{variance}$ is splitted at the middle of $Max_L$ (White line 3). On the right a region that has the $V_{LB}$ greater than the $V_{L_W}$ and $T_{variance}$ is splitted externally to $Max_L$ (Black line 2).*

## 3.6 Recursive filter

Different from the heuristic filter, with this filter we take care of the analysis on neighboring components in each homogeneous region to identify non-text elements.

Considering that the structure of the text is usually in rows and columns, we can use a statistical approach, examining the relationship between the connected components, using the median to figure out the non-text candidates.

This recursive filter is the combination of maximum median and minimum median filter based on the features of the components of each homogeneous region.

As input for this filter we have all the leaves of our region tree: let $R_i$ be the analyzed region and let $L_T$ be the list of all the components inside $R_i$,


*Figure 3.8: The $\hat{f}$ Regions extracted from the document at the end of the recursive segmentation phase.*

for every region we extract:

$$\bullet\ \Omega_1 = \{CC_{area}(CC_i)\ |CC_i \in L_T\} \tag{3.29}$$

$$\bullet\ \Omega_2 = \{H_i = Y_{max_i} - Y_{min_i}\ |CC_i \in L_T\} \tag{3.30}$$

$$\bullet\ \Omega_3 = \{W_i = X_{max_i} - X_{min_i}\ |CC_i \in L_T\} \tag{3.31}$$

And for each components $CC_i$ in $L_cc$ we compute also:

- $WS_{left}$: is the whitespace between $CC_i$ and $CC_{LNN}(CC_i)$ computed as:

$$WS_{left} = X_{min_i} - X_{max_j},\ where\ CC_j = CC_{LNN}(CC_i) \tag{3.32}$$

- $WS_{right}$: is the whitespace between $CC_i$ and $CC_{RNN}(CC_i)$ computed as:

$$WS_{right} = X_{min_j} - X_{max_i},\ where\ CC_j = CC_{RNN}(CC_i) \tag{3.33}$$

- $WS$: is the set which contains all the whitespaces, computed as:

$$WS = \{WS_{right} > 0 \mid CC_i \in L_T\} \tag{3.34}$$

Based on these, and on the features extracted in Section 3.3, we use an efficient method which uses the combination of whitespace analysis and a statistical approach to classify the text and non-text elements in each region, for this purpose we execute two filters simultaneously: maximum median filter and minimum median filter.

**Maximum median filter :** The $i_{th}$ connected component is suspected to be a non-text element if it satisfies the following condition:

$$CC_{area}(CC_i) = \max(\Omega_1) \wedge CC_{area}(CC_i) > k_1 * median(\Omega_1) \tag{3.35}$$

and one of the two following condition:

$$\bullet\ H_i = \max(\Omega_2) \wedge H_i > k_2 * median(\Omega_2) \tag{3.36}$$

$$\bullet\ W_i = \max(\Omega_3) \wedge W_i > k_3 * median(\Omega_3) \tag{3.37}$$

$K_1$, $K_2$, $K_3$ are the thresholds for the difference between the area, height, width of $CC_i$, and the median of them and they depend on the dispersion of the region. These are calculated as:

$$K_j = \max(\frac{mean(\Omega_j)}{median(\Omega_j)},\ \frac{median(\Omega_j)}{mean(\Omega_j)}) \tag{3.38}$$

If the $i_th$ component is suspected to be a non-text component, we will classify it definitively if it satisfied both the following two conditions:

$$\bullet \min(WS_{left_i},\ WS_{right_i}) > \max(median(WS), mean(WS)) \tag{3.39}$$

$$\bullet (max(WS_{left_i},\ WS_{right_i}) = max(WS)) \vee \min(WS_{left_i},\ WS_{right_i}) > 2 * mean(WS)) \tag{3.40}$$

or this condition:

$$(|CC_{LN}(CC_i)| = |WS_{left}| \wedge |CC_{LN}(CC_i)| > 2) \vee (|CC_{RN}(CC_i)| = |WS_{right}| \wedge |CC_{RN}(CC_i)| > 2) \tag{3.41}$$

The first expression provides a method for identifying non-text elements when the document has a complex layout. The second expression instead gives an efficient method to identify non-text component; at least theoretically, in each homogeneous region, the number of left neighbors and right neighbors of every text connected component is one, however, due to the noise and the skew of document, we can fix the maximum number of the neighbors to two [6].

**Minimum median filter :** Unlike maximum median filter, this filter is primarily responsible for finding the small lines or noise elements.

The $CC_i$ is suspected to be non-text element if it satisfies one of following two conditions:

$$\bullet H_i = \min(\Omega_2) \wedge H_i > \frac{median(\Omega_2)}{k_2} \tag{3.42}$$

$$\bullet\, W_i = \min(\Omega_3) \wedge W_i > \frac{median(\Omega_3)}{k_3)} \tag{3.43}$$

And is definitively classified if it satisfies the same condition of the maximum median filter.

If a component is classified as non-text component, we remove it from $L_T$ and insert in $L_NT$, so we insert the region $R_i$, in which resides the component, in a list of modified regions $MR$. When we have analyzed all the input regions, we send the $MR$ as input of the Section 3.5 and we repeat this two steps until the $MR$ list is empty.

## 3.7 Post processing

Our algorithm provides an effective method to classify the text and non-text components, however the overall process depends on the quality of the binary images. In fact binary images contain some noise, this leads often to have many missing or unexpected components, especially for the non-text elements which usually are not uniformly binarized due to their range of colors.

This components are often not small enough to be removed or classified as non-text components by the heuristic or the recursive filter.

So, in this step, we develop a method to eliminate the noise and correct the classification for misclassified components. First, we use the bounding box of all the text connected components and apply them as mask to the input binary document $f$. Let $CC_i \in L_T$ be the set of all the text connected components extracted and classified from the previous phases, the

new document $f_{NT}$ is obtained from:

$$fNT(x,y) == \begin{cases} 0 & if\ f(x,y) \in B(CC_i) \\ f(x,y) & elsewhere \end{cases},\ \forall CC_i \in L_T \qquad (3.44)$$

Second, we apply a morphological transformation, that is a simple operation based on the image shape. This operation takes as input our binary image, $f_{NT}$, and a small rectangular kernel, calculated as the 0,5% of each dimensions of the document. The morphological operation that we perform is called dilation and in this operation the kernel slides through the image, as in 2D convolution, and a pixel has value 1 if at least one pixel under the kernel is 1. So it increases the size of foreground objects in the image.

Finally we compare the text component $CC_i \in L_T$ with the non-text document $f_{NT}$, and if the pixels intersection $P_{intersection}$, calculates as:

$$P_{intersection} = f_{NT} \cap B(CC_i) \in L_T \qquad (3.45)$$

is greater than zero we delete it from the text component list $L_T$ and add it to the non-text component list $L_{NT}$.

## 3.8 Text region segmentation

In this section, we present a method for grouping the text components into a set of text regions. First of all we group the text components in $L_T$ row by row to extract the text lines relying on the white-space analysis [4]. Then, we use this lines to perform the paragraph segmentation, i.e., to separate the text document into individual paragraphs.

**Text line extraction** In this step, we group together the text elements in $L_T$ to extract the list of text lines ($T_L$). To figure out which components belong to the same line we use only heuristic and features already extracted in Section 3.3.

First of all we exploit the $CC_{same_row}(CC_i)$, associated to each component $CC_i \in L_T$, to build the lines, paying attention to the fact that each component belongs to one and only one line. We then sort the component, in each line, in ascending order of $X_{min_i}$.

After that we separate the lines taking into account the distances between the component $CC_i$ and the component $CC_{i+1}$ and their height, as long as there are components in the line.

Let $CC_i$ be the first component of the line, if the following expression is verified we include in the line also the component $CC_{i+1}$, contrarily we set the component $CC_{i+1}$ as the initial component of a new line:

$$|CC_{i+1} - CC_i| \leq 1,2 * \max(H_i, H_{i+1}) \tag{3.46}$$

The output of this step is a list of line, $T_L$, each identified by the coordinates of their bounding box (Figure 3.9).

**Paragraph detection** We then group the lines in $T_L$, exploiting their alignment that usually is vertical, to create the list of paragraphs.

First of all we sort all the lines by their $Y_{min}$ coordinate, after that, for each line, we search for the lines that intersect horizontally $T_{L_i}$ by:

$$\begin{aligned} &Intersect(T_{L_i}) = \\ &\{T_{L_j} \in T_L \mid X_{min_i} \leq X_{min_j} \leq X_{max_i} \vee \ X_{min_i} \leq X_{max_j} \leq X_{max_i}\} \end{aligned} \tag{3.47}$$

Then we filter all the lines that are too far from the considered one and

*Figure 3.9: On the left the lines extracted from the text line extraction step, on the right the paragraphs detection step.*

we obtain:

$$Comparable(T_{L_i}) = \{T_{L_j} \in Intersect(T_{L_i}) \mid Y_{min_j} - Y_{max_i} < \frac{\min(H_i,\ H_j)}{1,5} X_{max_i}\} \quad (3.48)$$

Finally we add all the lines considered comparable to the same paragraph, which have as contours, the extreme edges of the text lines that compose it. We continue until every text line has been assigned to a paragraph, $T_P$, therefore the output of this step is a list of paragraphs, containing all the text lines (Figure 3.9).

## 3.9 Non-text region classification

The last steps, after having check the paragraphs to ensure they really are text, aims to extract and classify the non-text regions. First of all we take all the output paragraphs from the Section 3.8 and we classify the region of pixels of the original colored image limited by the bounding box of the paragraph, by the use of the convolutional neural network called *AlexNet*; indeed the heuristics presented in [6] can lead to misclassified regions impacting negatively on the entire system. For this reason we, unlike the authors, make use of a neural network trained to classify text, images, tables and graphs and that does not need heuristic parameters.

The paragraphs can be classified as text, and then we leave them in the list $T_P$, images, tables, graphs or noise, and in these cases we remove they from the list of paragraphs. After that we create the non-text document built through the use of a mask composed of all paragraphs in $T_P$ as it follows:

$$fNT(x,y) == \begin{cases} 0 & if\ f(x,y) \in T_{P_i} \\ f_T(x,y) & elsewhere \end{cases} \tag{3.49}$$

Subsequently we classify each region belonging to $RGB_{NT}$ with the use of the same convolutional neural network, *AlexNet*, previously used for the paragraph correction; also in this step, unlike us, [6] make use of heuristics for the classification of the regions.

We apply to $f_{NT}$ the recursive segmentation explained in Section 3.5, obtaining:

$$R_{NT_1} \cup R_{NT_2} \cup ... \cup R_{NT_i} = f_{NT} \tag{3.50}$$

We then collect, from the original rgb document, the regions corresponding to each region extracted from our algorithm, $RGB_{NT}$. After that we identify and draw, with a different color, the contours of the bounding box


*Figure 3.10: The resulting document after the whole process, we can see the different contour for each region: orange for the text, green for the image and blue for the graph.*

of each region according to the class which it belongs.

We then return the document with drawn the contours of each identified region divided in text, image, table or graph regions (Figure 3.10).

# Chapter 4

# Experimental results

In this chapter we present the experiments performed in order to evaluate the performance of our approach. Considering that we aim at outperforming the state of the art, we compare our performance with those of [6].

We start by presenting the used tools, we then illustrate the dataset and the settings used to train the convolutional neural network exploit for the classification of the regions. Next, we explain our evaluation procedure and the achieved results obtained by differents criteria.

Finally we explore the limits and the failed experiments from the qualitative point of view.

## 4.1 Tools and libraries

Both computer vision and deep learning are becoming fields broadly diffused; a lot of tools are being developed from private companies or from the community and released as open-source. To speed up the implementation of our research, and also for the optimization performed in such tools, we used the

following tools and libraries:

**Numpy** NumPy [37] is the fundamental package for scientific computing in Python. It is a library that provides a multidimensional array object, various derived objects, and a collection of routines to operate on arrays, including mathematical, logical, shape manipulation, sorting, selecting, basic linear algebra, and basic statistical operations.

Since the calculations on the pixels are computationally heavy intense we have used NumPy to model pixels in a multidimensional array object.

This object, ndarray, is the core of the NumPy packages, it encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.

**OpenCv** OpenCV (Open Source Computer Vision Library) [38] is an open source computer vision and machine learning library. Its purpose is to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

The library has more than 2500 optimized algorithms which include both classic and state-of-the-art computer vision and machine learning algorithms, as well as I/O operations on the images. We used these algorithms to fit an input image in a numpy array, transform the color space of the image from the bgr to the bw space, binarize the documents, find the connected component, and apply morphological transformation to the images.

**LabelImg** The training datasets are the main asset for the deep learning algorithms. Building a dataset of annotated images with many objects is a costly and lengthy work.

In object supervised detection and recognition the algorithms need large

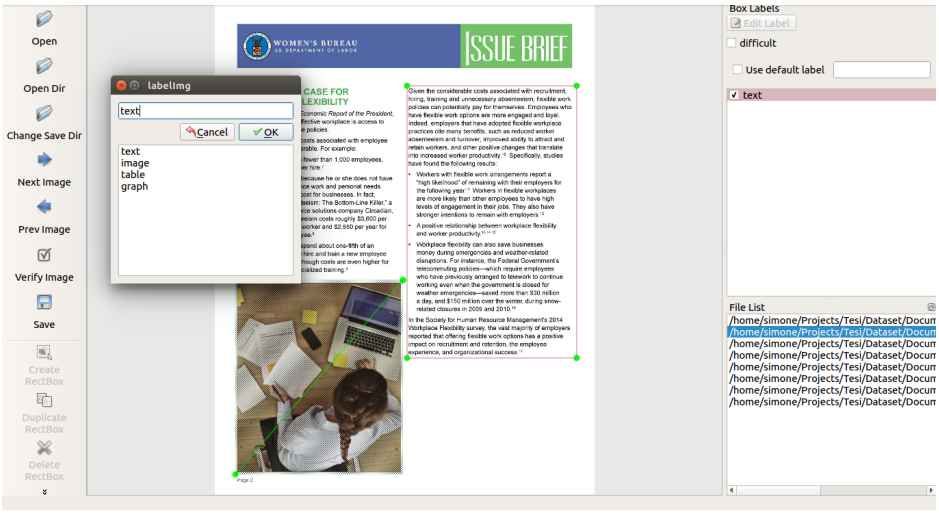


*Figure 4.1: Screenshot of the labelImage tool.*

image or video collections with ground truth labels spanning many different object categories. For each object present in an image, the labels should provide information about the object's identity, shape, location, and possibly other attributes such as pose.

LabelImg [39] is a graphical image annotation tool, that provides functionalities such as drawing bounding box and extract the corresponding annotation as XML files in PASCAL VOC format, the format used by ImageNet (Figure 4.1).

We used this tool to label our dataset, to proceed faster without having to create a new one.

**Tensorflow** TensorFlow [40] is an open source software library for numerical computation using data-flow graphs. It was originally developed by the Google Brain Team within Google's Machine Intelligence research organization for machine learning and deep neural networks research.

At a high level, TensorFlow is a Python library that allows users to express

arbitrary computation as a graph of data flows. Nodes in this graph represent mathematical operations, whereas edges represent data that is communicated from one node to another. Data in TensorFlow are represented as tensors, which are multidimensional arrays.

The implementation of *AlexNet* (Section 4.2) that we used for the classification is build on top of tensorFlow. Also the implementation of *YOLO* (Section 4.5) that we tried without giving good results is build in TensorFlow.

## 4.2 Dataset and AlexNet

Convolutional neural networks, as all supervised algorithms in machine learning, need data to be trained to recognize unseen examples.

Considering that the data are the most important and costly assets for a deep learning algorithm, we, comprehensively, do not have found anyone disposed to offer their own labeled data for the task under analysis. So we had to create our dataset manually, starting with images of publicly available documents.

It was our intention to train and test both AlexNet and YOLO, but having their input labels different formats, we exploited the fact that from the labeled data and the labels in the YOLO format, with a Python script, we can obtain the labeled data and the labels according to the AlexNet format. To augment the training set of AlexNet we also use some regions extracted directly from the recursive segmentation phase of our algorithm (Section 3.5).

As for the dataset used to train AlexNet, this was composed from publicy available images, taken from [41], we had manually labeled, as follows:

- 3770 text regions

- 381 image regions
- 334 graph regions
- 221 table regions
- 376 noise regions.

In this section we explore only the settings for what concern the training phase of AlexNet, while we explore the parameter used for the training of YOLO in the Section 4.5.

We fine-tune the neural network with our data, so to do this we had to first download the pre-trained weights [42] from which to start.

Given the size of our dataset, we arbitrarily chose to finetune only the last three layers.

We used the same parameter settings of the original paper, i.e., the batch size, the dropout probability, while we had change the learning rate and set it to a value of 0,001 instead of the original value of 0,01.

We split the dataset in training and validation set, using a stratified random sampling [43] in which we used the 70% of the data of each class for the training set and the 30% for the validation set.

Note that we do not have further splitted the dataset in the test set, this is because our test set is a completely different dataset [5] which allowed us to compare our work with [6].

## 4.3 Evaluation procedure

The performance of our method is evaluated based on the same measures of [6], we then use: the recall that is the number of pixels within ground truth regions that are also within a region in the segmentation result, divided by the overall number of pixels in ground truth areas, and the precision that is the number of pixels within ground truth regions that are also within a region in the segmentation result, divided by the overall number of pixels in segmentation result regions we also used the F-measure that shows the harmonic mean of the two previous measures.

$$Recall = \frac{Output \cap GroundTruth}{GroundTruth} \tag{4.1}$$

$$Precision = \frac{Output \cap GroundTruth}{Output} \tag{4.2}$$

$$F - measure = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{4.3}$$

We calculate these measure, for each document in the dataset [5], separately for the text and non-text regions; since there are not overlapping regions this is not a lack and does not affect the results.

## 4.4 Results

In this section we present the obtained results both for that concern the convolutional neural network, AlexNet, that for our algorithm comparing it with the result of [6].

Our system was implemented in Python with the help of the library presented in the section 4.1, our workspace is configured with a system Intel Core i7-4510U, 8G RAM, Ubuntu 16.04 - 64 bits. Our test set for what concern the evaluation of our method is composed of 55 English documents of ICDAR2009 page segmentation competition dataset [5], that are document images with various layouts.

We trained AlexNet network entirely on CPU; since we do not have an early stopping mechanism we trained the network for ten epochs and manually, for each epoch, we compared the values related to the accuracy of the validation set for checking overfitting problems. In Table 4.1 we compare the accuracy values obtained re-training the whole network, called experiment 1, and only the last three layers, experiments 2, on the same dataset.

| Epoch | Accuracy experiment 1 (%) | Accuracy experiment 2 (%) |
|---|---|---|
| *1* | *76,78* | *86,72* |
| *2* | *76,63* | *89,26* |
| *3* | *75,64* | *90,82* |
| *4* | *74,64* | *90,23* |
| *5* | *75,07* | *92,37* |
| *6* | *76,99* | *93,08* |
| *7* | *75,92* | *93,84* |
| *8* | *80,68* | *94,66* |
| *9* | *77,70* | *96,22* |
| *10* | *81,04* | *95,57* |

*Table 4.1: Accuracy on the validation set during the training of AlexNet, finetuning the whole network (left) and only the last three layers (right).*

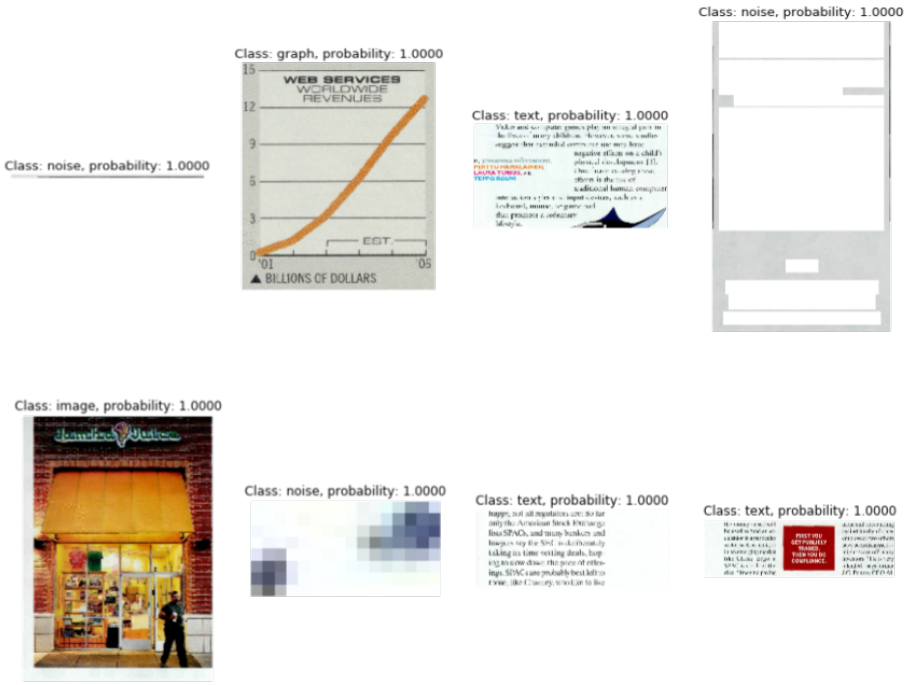


*Figure 4.2: Some images extracted from our algorithm and classified after training AlexNet.*

As we expected, given the dimension of our dataset, the model trained by fine-tuning only the last three layers fits better the data and reaches a higher accuracy. We also test qualitatively the classifier on some regions extracted by our algorithm as we can see in Figure 4.2.

Regarding our overall algorithm, we compared it with the results presented in [6], during the evaluation we did three different tests: the first one in which we try to segment every zone that we extract (Table 4.2), a second experiment in which we use for the evaluation the bounding box of each zone (Table 4.3) and finally we mixed the two using the segmentation criteria for the text region and the bounding box criteria for the non text region (Table 4.4).

| | Our Method | Tran et al [6] |
|---|---|---|
| *Text Recall (%)* | *97,35* | *94,46* |
| *Text Precision (%)* | *91,27* | *93,18* |
| *Text F-Measure (%)* | *94,21* | *93,8* |
| *Non-Text Recall (%)* | *74,26* | *85,28* |
| *Non-Text Precision (%)* | *95,4* | *83,22* |
| *Non-Text F-Measure (%)* | *83,5* | *84,19* |

*Table 4.2: Comparison of our method with [6] segmenting all the extracted regions.*

| | Our Method | Tran et al [6] |
|---|---|---|
| *Text Recall (%)* | *95,37* | *94,46* |
| *Text Precision (%)* | *92,80* | *93,18* |
| *Text F-Measure (%)* | *94,07* | *93,8* |
| *Non-Text Recall (%)* | *80,93* | *85,28* |
| *Non-Text Precision (%)* | *91,56* | *83,22* |
| *Non-Text F-Measure (%)* | *85,92* | *84,19* |

*Table 4.3: Comparison of our method with [6] using the bounding box of all the extracted regions.*

| | Our Method | Tran et al [6] |
|---|---|---|
| *Text Recall (%)* | *97,35* | *94,46* |
| *Text Precision (%)* | *91,27* | *93,18* |
| *Text F-Measure (%)* | *94,21* | *93,8* |
| *Non-Text Recall (%)* | *80,93* | *85,28* |
| *Non-Text Precision (%)* | *91,56* | *83,22* |
| *Non-Text F-Measure (%)* | *85,92* | *84,19* |

*Table 4.4: Comparison of our method with [6] segmenting the text regions and using the bounding box of the other regions.*


*Figure 4.3: Example of test set documents. In the first row there are the original document, in the second row there are the ground truth of each image, in the third and in the fourth, there are respectively the text and the non-text extracted by our algorithm.*


*Figure 4.4: Example of overmerged paragraphs.*

## 4.5 Limits

In this section we explore, qualitatively, the limits, not only of our method, but also the learning limits of YOLO using our own dataset and the limits of a renowned open source OCR engine, Tesseract [44], apply to the extracted text regions.

Regarding our system the most relevant problem occurs in the segmentation phase, in fact, being based on variance, if we have long consecutive series of, black or white, pixels the document will not be divided. We can see an example of this problem in Figure 4.6.

We also noted some inaccuracy when the input document has close text lines but not belonging to the same paragraph, resulting in overmerged paragraphs, as the case of the identity card (Figure 4.4).

We also tried out to train YOLO with the same dataset used for the

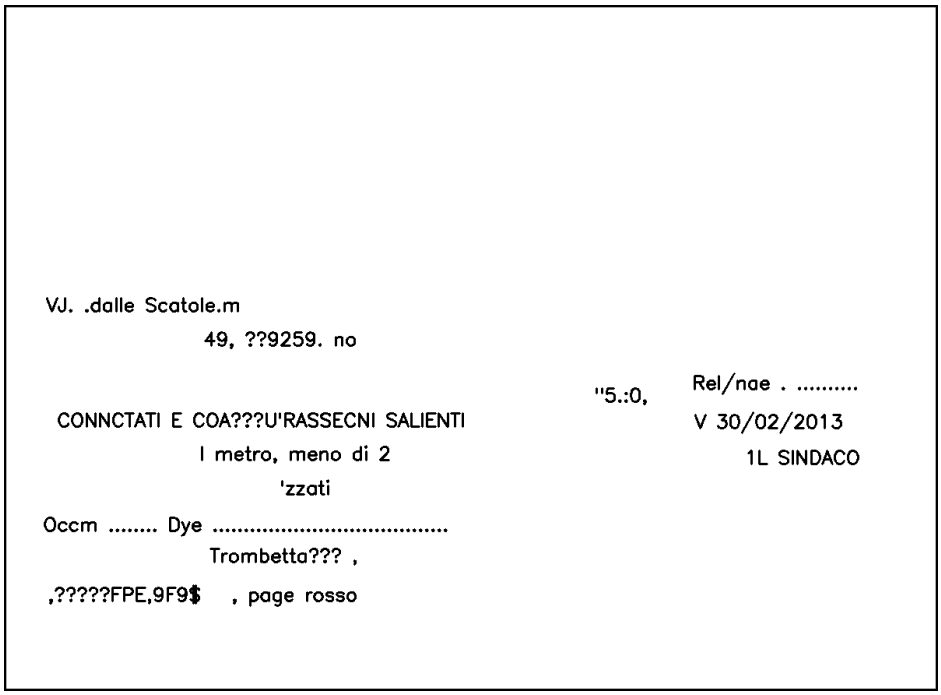


*Figure 4.5: The text extract with Tesseract OCR from the document in Figure 4.4*

training of AlexNet. We trained it with the same settings of the original paper, and also with different learning rate and batch size, but the result are almost the same. With none of the used settings we were able to get a loss of less than 14 and a confidence in a bounding box higher than 10

Finally we also explored the limits of Tesseract OCR [44] on some text regions extracted from our system: though the text of some standard documents, taken from the test set, have been correctly recognized, we notice many difficulties in the case of real world document in which the text is not well separated and clear such in Figure 4.5.

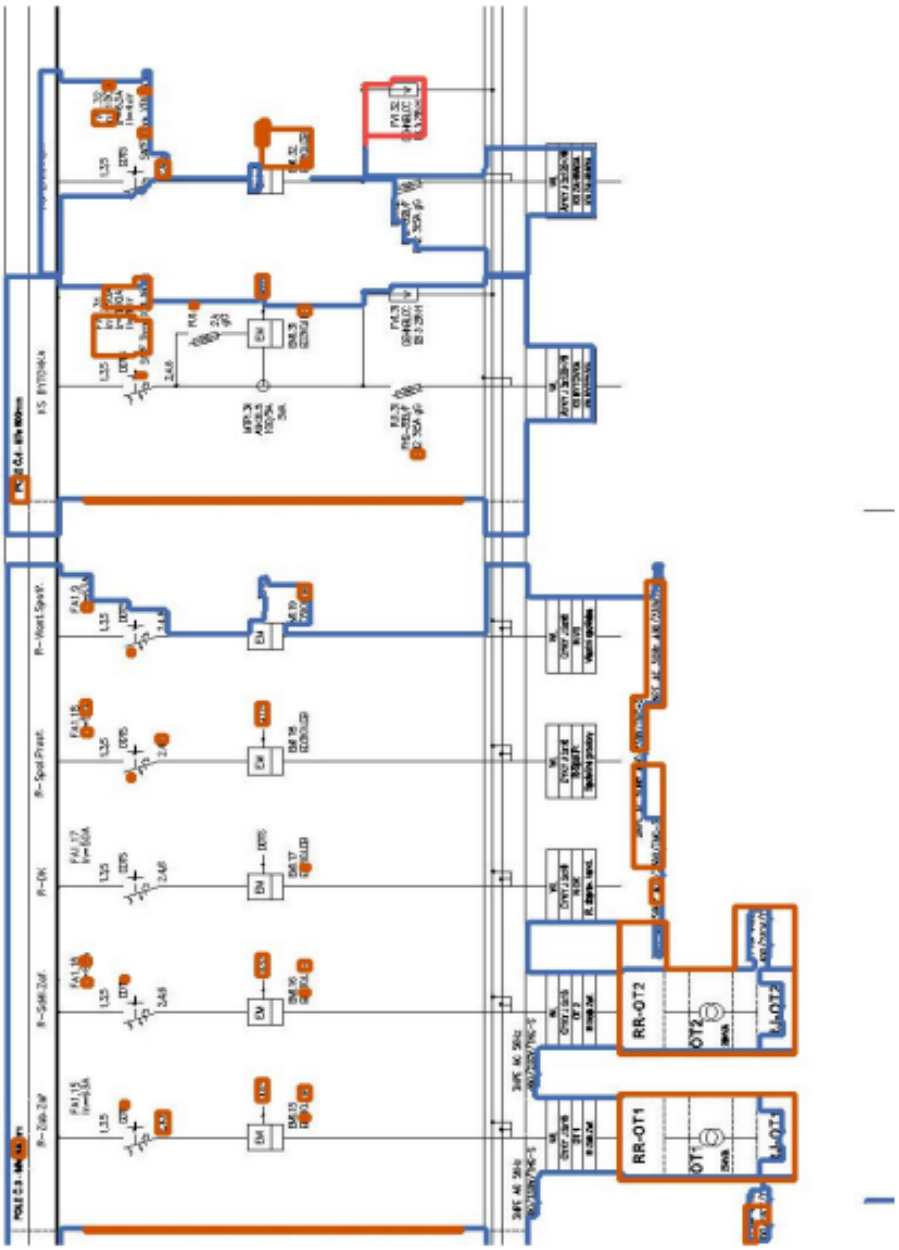


*Figure 4.6: Example of long consecutive series of black pixels.*

# Chapter 5

# Conclusions and future work

This thesis was aimed at improve an existing document segmentation system to advance the state of the art for that concerning the parsing of a document.

The thesis started from a document parsing system dedicated to the segmentation and the classification of the extracted zones relying only on heuristics. Such a system showed improvable performance in primarily in the classification phase.

The resulting parsing system has proved his effectiveness improving the overall performance. In particular the used convolutional neural network has highlighted the possibility to learn easily the proposed classes (text, image, table, graph and noise) and leads the system to best results both in the identification of text, and non-text, areas.

However, our system has also shown some limits and weaknesses moving from test documents to real-world documents.

The major deficiency has been found in the segmentation phase which is

considerably influenced by the variance; this can lead to totally wrong results in the classification and in the overall system.

Future works in the context of our thesis may follow two main directions:

- The first one is to improve the segmentation phase developing other algorithms in order to be less sensitive to variance and extract less fragmented regions.
- The second is to use an end-to-end detection and classification system to avoid extracting regions with algorithms.

  Even if in our research it did not give positive results, this may be due to the increase in the network complexity, YOLO, with respect to AlexNet and so to a need of a different dataset.

# Bibliography

[1] Lawrence O'Gorman and Rangachar Kasturi. *Document image analysis*, volume 39. IEEE Computer Society Press Los Alamitos, 1995.

[2] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms: a literature survey. In *Document Recognition and Retrieval X*, volume 5010, pages 197–208. International Society for Optics and Photonics, 2003.

[3] David Doermann. The retrieval of document images: a brief survey. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 2, pages 945–949. IEEE, 1997.

[4] Kai Chen, Fei Yin, and Cheng-Lin Liu. Hybrid page segmentation with efficient whitespace rectangles extraction and grouping. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 958–962. IEEE, 2013.

[5] Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. A realistic dataset for performance evaluation of document layout analysis. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 296–300. IEEE, 2009.

[6] Tuan-Anh Tran, In-Seop Na, and Soo-Hyung Kim. Separation of text and non-text in document layout analysis using a recursive filter. *KSII Transactions on Internet and Information Systems (TIIS)*, 9(10):4072–4091, 2015.

[7] Harold Borko and Myrna Bernick. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162, 1963.

[8] GBG IDscan. www.idscan.com, identity validation — protecting your identity in a digital world.

[9] Swapnil A Vaidya and Balaji R Bombade. A novel approach of handwritten character recognition using positional feature extraction. *International Journal of Computer Science and Mobile Computing*, 2(6):179–186, 2013.

[10] Jamileh Yousefi. Image binarization using otsu thresholding algorithm. 2011.

[11] Oeivind Due Trier and Anil K. Jain. Goal-directed evaluation of binarization methods. *IEEE Transactions on pattern analysis and machine intelligence*, 17(12):1191–1201, 1995.

[12] Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung, et al. A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17(5):713–727, 2001.

[13] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[14] Kwan Y. Wong, Richard G. Casey, and Friedrich M. Wahl. Document analysis system. *IBM journal of research and development*, 26(6):647–656, 1982.

[15] Lawrence O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

[16] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

[17] Apostolos Antonacopoulos. Page segmentation using the description of the background. *Computer Vision and Image Understanding*, 70(3):350–369, 1998.

[18] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. Recursive xy cut using bounding boxes of connected components. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 952–955. IEEE, 1995.

[19] Raymond W Smith. Hybrid page layout analysis via tab-stop detection. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 241–245. IEEE, 2009.

[20] Antoine Azokly. Une approche générique pour la reconnaissance de la structure physique de documents composites. *IIUF-University of Fribourg*, 1995.

[21] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[22] Dacheng Wang and Sargur N Srihari. Classification of newspaper image blocks using texture analysis. *Computer Vision, Graphics, and Image Processing*, 47(3):327–352, 1989.

[23] Luigi Cinque, Luca Lombardi, and G Manzini. A multiresolution approach for page segmentation. *Pattern Recognition Letters*, 19(2):217–225, 1998.

[24] Claes Strannegård, Olle Häggström, Johan Wessberg, and Christian Balkenius. Transparent neural networks. In *International Conference on Artificial General Intelligence*, pages 302–311. Springer, 2012.

[25] Pierre Heroux, Eric Trupin, and Yves Lecourtier. Modélisation et classification pour la rétroconversion des documents. *CIFED'00*, pages 413–421, 2000.

[26] www.iapr.org/archives/icdar2015/index.html, icdar 2015 — 13th iapr international conference on document analysis and recognition.

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

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[29] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[30] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[31] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

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

[34] Emilio Soria Olivas, Jose David Martin Guerrero, Marcelino Martinez Sober, Jose Rafael Magdalena Benedito, and Antonio Jose Serrano Lopez. Handbook of research on machine learning applications and trends: Algorithms, methods and techniques-2 volumes. 2009.

[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[36] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[37] www.numpy.org, numpy - numpy.

[38] www.opencv.org, opencv library.

[39] tzutalin. github.com/tzutalin/labelimg, labelimg.

[40] Google Brain Team. www.tensorflow.org, tensorflow.

[41] UNT Libraries' Digital Projects Unit. digital.library.unt.edu, digital library.

[42] www.cs.toronto.edu, alexnet implementation + weights in tensorflow.

[43] Z Reitermanova. Data splitting. In *WDS*, volume 10, pages 31–36, 2010.

[44] Ray Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.