Text Detection in Natural Scenes and Technical Diagrams with Convolutional Feature Learning and Cascaded Classification

by

Siyu Zhu

B.S. Shanghai Jiao Tong University
M.S. Rochester Institute of Technology

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology

May 12, 2016

Signature of the Author _____

Accepted by _____
Coordinator, Ph.D. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

COLLEGE OF SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

The Ph.D. Degree Dissertation of Siyu Zhu
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

_____

Dr. Richard Zanibbi, Dissertation Advisor

_____

Dr. Bo Yuan

_____

Dr. Carl Salvaggio

_____

Dr. Nathan Cahill

_____

Date

DISSERTATION RELEASE PERMISSION

ROCHESTER INSTITUTE OF TECHNOLOGY

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

Title of Dissertation:

Text Detection in Natural Scenes and Technical Diagrams with Convolutional Feature Learning and Cascaded Classification

I, Siyu Zhu, hereby grant permission to Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature _____

Date

Text Detection in Natural Scenes and Technical Diagrams with Convolutional Feature Learning and Cascaded Classification

by

Siyu Zhu

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

An enormous amount of digital images are being generated and stored every day. Understanding text in these images is an important challenge with large impacts for academic, industrial and domestic applications. Recent studies address the difficulty of separating text targets from noise and background, all of which vary greatly in natural scenes. To tackle this problem, we develop a text detection system to analyze and utilize visual information in a data driven, automatic and intelligent way.

The proposed method incorporates features learned from data, including patch-based coarse-to-fine detection (Text-Conv), connected component extraction using region growing, and graph-based word segmentation (Word-Graph). Text-Conv is a sliding window-based detector, with convolution masks learned using the Convolutional k-means algorithm (Coates et. al, 2011). Unlike convolutional neural networks (CNNs), a single vector/layer of convolution mask responses are used to classify patches. An initial coarse detection considers both local and neighboring patch responses, followed by refinement using varying aspect ratios and rotations for a smaller local detection window. Different levels of visual detail from ground truth are utilized in each step, first using constraints on bounding box intersections, and then a combination of bounding box and pixel intersections. Combining masks from different Convolutional k-means initializations, e.g., seeded using random vectors and then support vectors improves performance. The Word-Graph algorithm uses contextual information to improve word segmentation and prune false character detections based on visual features and spatial context. Our system obtains pixel, character, and word detection f-measures of 93.14%, 90.26%, and 86.77% respectively for the ICDAR 2015 Robust Reading Focused Scene Text

dataset, out-performing state-of-the-art systems, and producing highly accurate text detection masks at the pixel level.

To investigate the utility of our feature learning approach for other image types, we perform tests on 8-bit greyscale USPTO patent drawing diagram images. An ensemble of Ada-Boost classifiers with different convolutional features (MetaBoost) is used to classify patches as text or background. The Tesseract OCR system is used to recognize characters in detected labels and enhance performance. With appropriate pre-processing and post-processing, f-measures of 82% for part label location, and 73% for valid part label locations and strings are obtained, which are the best obtained to-date for the USPTO patent diagram data set used in our experiments.

To sum up, an intelligent refinement of convolutional k-means-based feature learning and novel automatic classification methods are proposed for text detection, which obtain state-of-the-art results without the need for strong prior knowledge. Different ground truth representations along with features including edges, color, shape and spatial relationships are used coherently to improve accuracy. Different variations of feature learning are explored, e.g. support vector-seeded clustering and MetaBoost, with results suggesting that increased diversity in learned features benefit convolution-based text detectors.

Acknowledgements

I would like to acknowledge my amazing advisor Dr. Richard Zanibbi for his selfless help and motivation. He has given me valuable advices and encouragement in both academic and life and has provided me with many opportunities professionally.

I would like to thank my thesis committee, Dr. Carl Salvaggio, Dr. Bo Yuan and Dr. Nathan Cahill, for their help. They have taught me in numerous ways and provided many important guidances through my research.

I would like to thank all my lab mates, including but not limited to Lei Hu, Kenny Davila, Manish Kanadje, Zachary Miller, Nidhin Pattaniyil, Christopher Sasarak, Awelemdy Orakwue, David Stalnaker, George Chen, Ben Miller-Jacobson, Kedarnath Calangutkar and Franscisco Alvaro. It was a very productive and exciting working experience in DPRL lab. And I am truly thankful for having the opportunity to work with those people.

I would like to thank my parents for their unconditional support. Most of all, I am grateful to my wife, Ning Ren, who has always been a tireless supporter of me and my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As machine learning and image processing algorithms are rapidly developed in recent years, complex image understanding techniques are being significantly improved. In this thesis, we will scrutinize a specific area of image understanding, focusing on text detection in natural scene images.

Since enormous amounts of digital images have been generated and stored throughout the internet, reading, processing and understanding these images is a great challenge and with great impact on the automation of industry and people's life. In 2012, the Google street view photos take about 20 petabytes of disk space, covering more than 5 million miles of roads, 3000 cities, and 39 countries. There are more than 2 billion photos are shared on Facebook alone in 2015.

Modern Graphical Processing Unit (GPU), Field Programmable Gate Array (FPGA) and other image processing devices have been developed significantly in the previous decade, allowing much-improved image processing speed with greatly lowered power consumption. GPU computation, while combining with suitable software, could fold the computation time of typical image processing by 10 to 20 times.

Recent development in machine learning algorithms provides people powerful tools to abstract, analyze and utilize image information in a data-driven, intelligent, and automatic way. Therefore, the rich demand of intelligent image processing algorithms and development of supporting computer science promotes more powerful and intelligent image understanding algorithms.

Scene understanding is difficult, especially for complexed natural scenes. As image processing algorithm reads pixel values as the raw input, a huge amount of data is needed to be processed, even for a few images with regular sizes. Besides sheer size problem, images captured from nature often contains

very complexed content, with interesting objects and many noise and backgrounds. As images are taken by cameras in a real world, lighting condition, scale, distance, and orientations are with very little control. To extract useful information from raw data needs strong abstraction, robust classification and intelligent adaptation. Texts in images often contain crucial information to understand the content. In picture labeling in social networks, geolocation tag registration, road sign recognition for autonomous driving and other applications, it is vital to correctly locate, segment, extract, and recognize texts from images.

Document image analysis and optical character recognition have been studied for many years. However, conventional systems provided very poor performance for text in complexed scenes. The difference between traditional document analysis and modern scene understanding is the great increase in variation of data in terms of appearance, location, structure and background. A crucial step in recognizing scene text is detection, which consists of locating, segmenting and extraction of the text from the image. The detection system should be intelligent enough, to extract text from all kinds of environment, regardless their differences in shape, size, orientation, font, colors and positions. In order to fulfill this task, modern machine learning algorithms along with specified designs for text target are used to form an accurate system.

In this thesis, we will discuss the methodology design and experiment for an intelligent text detection system. We divided the discussion into several segments, corresponding to different steps of the process, including detection, validation, and segmentation. The proposed system achieved state-of-the-art accuracy on detection task for public datasets. This discussion in text detection could be used and combined with segmentation and recognition algorithms to build an end-to-end text understanding system.

1.1 Thesis Statement and Research Questions

Image understanding is a combination of image processing and machine intelligence. The raw images as input are processed to create useful data for a machine learning system to extract, organize and understand information for a specified purpose. In this thesis, we propose to utilize machine learning algorithm to locate and detection text regions from very complexed images.

Our thesis statement is: **A simple algorithm for text detection in natural scenes can be produced to get higher accuracy than current systems without the use of image meta-data, OCR, or textual language models.**

Our experiments are designed to answer several research questions as listed below:

1. Are visual features alone sufficient for high-accuracy text detection in natural scenes?
2. How can features obtained via convolutional k-means be made more discriminating?
3. How can detection accuracy be improved by combining convolutional k-means detection with subsequent processing?
4. How can we use the different levels of ground truth information to improve detection accuracy?
5. How can convolutional k-means be used effectively to detect text in images other than natural scenes, e.g. diagram images?
6. How does detection in grayscale document images using convolutional k-means differ from detection in color images from natural scenes?

We need to point out that our system is applied to still images. For videos, the algorithm could be adapted and temporal information could be added. The images are captured by the conventional camera, therefore, no additional information besides the image pixel values are used. In a broader concept of scene understanding, depth information from LIDAR or infrared sensing array (range camera) might be used. Other information like GPS location might also be used to aid the accuracy of the detection and recognition. However, in the scope of this thesis discussion, pure image information is utilized. Therefore, one could save hardware expenses while developing real-world applications based on our proposed algorithm.

When processing text related images, a useful ancillary data is the lexicon data. By applying a language model with given grammar, vocabulary, and dictionary, one could enhance the detection and recognition by selecting reasonable recognition results from a predefined codebook. This thesis does not consider language model, and text detection is applied merely based on the image information of the object. Therefore, the performance that given by this system has potential to be further increased by fitting language model, if the character is predefined to be in a specified language (e. g. English). On the other hand, the system might be able to adapt to other languages without major modifications since no language model is required.

1.2 Contributions

In the thesis, we discuss the feature learning process for text detection, a comparison of using different convolutional kernels, different classification methods, how to convert from patch level detection map to pixel level and then bounding box level.

The key contribution of the thesis are:

- Feature learning method convolutional k-means is modified and improved. The features are learned automatically from the image data, and therefore adaptive to the dataset. The discussions are included in Chapter 3.

- Subsequent processing and detection method that utilizes the learned features are refined and improved. Coarse-to-fine detection is proposed which is similar to human visual attention. Contextual features are used in coarse scanning and differing rotation angles and aspect ratios are used in fine scanning. Innovative region growing and graph model based segmentation methods are introduced. Patch level prediction map is converted to CCs and precision is improved by using two stage of validation with spatial relationship information. The discussions are included in Chapter 4 and Chapter 5 for the natural scene and diagram images respectively.

- Different representations of ground truth, including bounding box and pixel levels, are combined to train and evaluation detection system. The discussions are included in Chapter 6.

- Detection systems with high accuracy are proposed. We tested our system using ICDAR 2015 focused scene text dataset and USPTO competition dataset. The proposed systems for natural scene text detection and document image part label recognition both get state-of-art performance. The discussions are included in Chapter 6.

- New analysis is included for text detection system. Different approaches to feature learning are discussed; Classification methods, including different cascaded classification models, are studied; The effects of using different sample equalization methods and classification methods are tested for word segmentation system, Word-Graph; The effects of different denoise methods are listed for document images.

1.3 Organization of This Document

This dissertation is organized as follows. In Chapter 2, Previous work about text detection is surveyed, including ground truth generation, evaluation, feature learning and classification methods. In Chapter 3, feature learning methods are introduced. Different feature learning techniques are proposed and analyzed. In Chapter 4, subsequent processing utilizing learned features for text detection in natural scenes is introduced. In Chapter 5, a different system for document image (engineer diagram) is introduced. In Chapter 6, the CC grouping, validation and segmentation methods are introduced to convert patch level detection into bounding box level. The final performance is compared with current state-of-art systems. Chapter 7 is the conclusion.

Chapter 2

Background

In this chapter, we discuss the background of text detection, from a broader perspective of computer vision to a narrower perspective of finding text location in images. The contents covered by this chapter includes: the impact of the technology, the problem definition, ground truth generation, evaluation metrics, feature generation and detection algorithm.

The following research questions as proposed in Section 1.1 are discussed. *Is visual feature alone sufficient for high-accuracy text detection in natural scenes?* Previous text detection systems are surveyed including ones used visual features only and ones used a combination of visual features and OCR and lexicon models.

How can features obtained via convolutional k-means be made more discriminating? and how can detection accuracy be improved by combining convolutional k-means detection with subsequent processing? Most conventional text detection systems utilized features with strong prior knowledge. Feature learning methods are proposed for text detection only in recent years. Therefore, systems using fixed features and learned features are surveyed and compared. The pros and cons are discussed for different types of systems.

How can we use different levels of ground truth information to improve detection accuracy? The ground truth generation and utilization methods are surveyed and discussed for previous work. Ground truth plays a very important role in both training and evaluation. Related to ground truth, evaluation metrics are also surveyed and discussed.

How can convolutional k-means be used effectively to detect text in images other than natural scenes, e.g. diagram images? and How does detection in grayscale document images using convolutional k-means

differ from detection in color images from natural scenes? For document images, very few previous systems utilized feature learning techniques. Strong prior knowledge and heuristic rule methods are commonly implemented for text detection in document and grayscale images. We survey the methods for document image analysis systems and compare their properties with text detection system for natural scene images.

This chapter begins with an overview in section 2.1, and then discuss different aspects of text detection systems. In section 2.2, ground truth and labeling methods are introduced, their effects on training and evaluation are discussed. In section 2.3, evaluation methods and metrics are introduced. In section 2.4, features used for detection are discussed. In section 2.5, recent text detection and localization algorithms are surveyed, including Document Image Analysis (DIA) systems and natural scene text detection systems. International Conference on Document Analysis and Recognition (ICDAR) competition is introduced [1] in section 2.6.

2.1 Overview

In this section, the impacts of text detection system are introduced briefly with a few examples. The challenges of text detection in document images and complex images are discussed. Finally, we provide definitions of terms that are commonly used in previous systems of text detection and clear some confusion where different terms refer to similar concepts.

2.1.1 Impacts

Computer vision application is ubiquitous, spreading from industrial robot controlling, vehicle navigation, visual surveillance to human-computer interaction (HCI) and electronic entertainment. In those applications, text (characters, words, phrases and sentences) is a interesting target for detection and recognition. Text detection/recognition serves many purposes in life and industrial practices:

- To increase accessibility and increase efficiency, a large amount of paper documents need to be converted to digital files for quick searching and editing. These documents include academic papers, books, newsletters, patents and spreadsheets. These tasks ask for fast but precise methods to convert scanned images into computer coded files automatically [2].

- For preserving, understanding and translating historical documents, enormous amount of paper documents need to be digitized and analyzed. Text localization, segmentation and recognition plays a vital role in these efforts to maintain and spread human culture and knowledge [3].

- The personal mobile computing devices (phones) are ubiquitous. Combined with accurate text detection algorithms, they are suitable for solving many problems, e. g. translation, math equation solver and note taking [4].

- Traffic control and autonomous driving requires highly accurate system for sign text detection and recognition. In the self-driving car scenario, cameras mounted on cars catch image frames from environment in high frequency, so computer embedded on cars need to quickly understand the text of road signs in order to guide driving correctly and timely [5].

- Social networks (Facebook) and search engines (Google) possess a huge amount of image/video data. They are also collecting more data each day with very high speed. In order to categorize and process these images/videos, texts within images could be used as hints, labels and indices for easier retrieval and grouping [6].

- Geolocation and geo-information systems (Google Maps) acquire huge amount of street views and natural scene images every day. In order to understand and register these images, traffic signs, business signs and street numbers are served as great cues. Therefore, powerful text detection systems are required to extract these cues from complex scenes accurately [7].

The text detection serves great purpose in document digitization/analysis, sign understanding, language translation, traffic control, autonomous driving and geo-tagging problems. Therefore, an accurate text detection would have a very broad application in those scenarios.

2.1.2 Challenges and Approaches

Text detection was previously resided in the realm of document analysis. However, documents analysis in strict format got very high accuracy in late 90s. On the other hand, unformatted texts in complex scene images were much more difficult and got very low detection rate.

Document Images (Simple Cases). Document Image Analysis (DIA) is a form of computer vision based document content digitization, understanding and management. The technology was built around the need to convert, manage and secure the escalating volume of documents (spreadsheets, word-processing documents, mails etc.) generated in industry organizations. Conventional DIA system usually contains preprocessing, binarization, segmentation and recognition (OCR) as shown in Figure 2.1.

In preprocessing, noise is reduced using bilateral filters or small Connected Component (CC) remover. Color is usually transformed from RGB space into more descriptive space, like CIEL*a*b* or HSV.

Document image is then binarized. Many binarization methods have been proposed to adapt to the variations of the illumination hence to increase accuracy.

Then images are segmented. The segmentation can divide the document into different regions, like text, background and graphs. Different level of segmentation produces paragraphs, text-lines, words and characters.

Finally, a recognizer converts image of detected text into text labels. After an initial recognition, lexicon or language models are usually used to refine the results by imposing constrains that enhance the accuracy. Recognition results sometimes are fed back to previous steps to improve accuracy.

Computer vision based optical character recognition (OCR) is the conversion of images of typed, hand-written or printed text into machine-encoded text. It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining. The term *OCR* was often used referring to recognition part of the document analysis or the entire procedure inter-changeably. Not long ago, the OCR was suggested by a number of researchers to be a 'solved' problem, as strictly formatted document images could be processed and recognized with extremely high accuracy [8].

Before 2000s, detection/localization of text is seen as a subtle component of an end-to-end system of document analysis. Text information is extracted from highly regularized formats, where the location of the text is clearly defined. Localization problem is of less interest than other problems, e. g. segmentation and recognition [9] [10] [11]. One survey has been conducted on camera-based document analysis [12]. Another survey has been conducted on text information extraction [13].

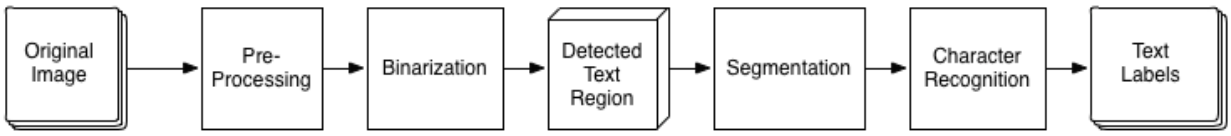


Figure 2.1: A conventional OCR system, which is usually composed of preprocessing, binarization, detection, segmentation, recognition

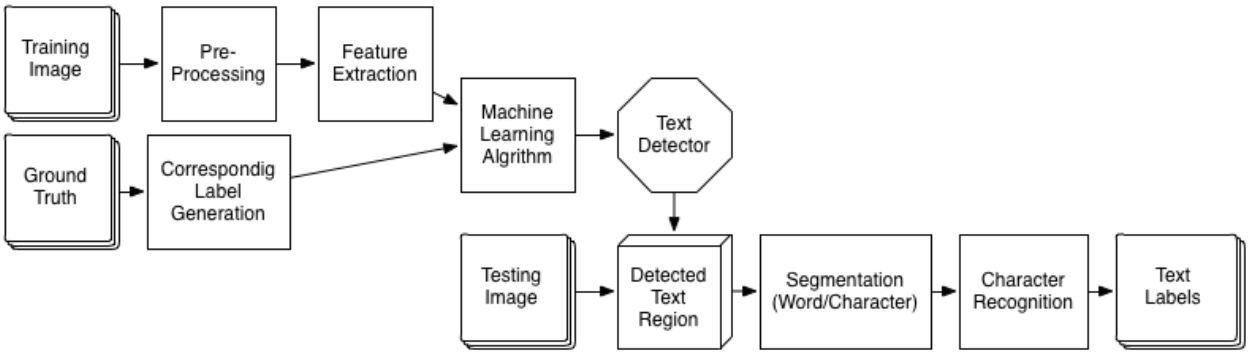


Figure 2.2: A sophisticated machine learning based OCR system for natural scenes, where features are learned automatically and adaptively from data.

Natural Scenes (Complex Cases). Recently, text detection/localization draw great attention in computer vision society. It is extensively studied in different circumstances for various targets: natural scene (camera based complex images), digital born (CGIs), street view, and video etc. Text can appear in forms of business signs, traffic signs, license plates, book covers, documents, monitor screens etc. Variations of foreground and background are much stronger, comparing to traditional document image analysis problems.

Text Detection system has many differences comparing to conventional DIA systems. The former problems usually can only be solved with adaptive models combining a large amount of features, while later one focuses more on utilizing human knowledge to process images based on heuristic rules. Conventional systems use features like geometric shapes, spatial relationships, textures to localize and segment texts, which requires strong prior knowledge from human experts and careful feature engineering. Thresholds are previously known, guessed or turned by human [14]. These limitations reduce the system robustness against variations, as they are more likely to be over-fitted to specified dataset. One could hardly achieve great accuracy for complex images using this technique. In Natural scenes, higher variation of data requires

systems to automatically adapt to data and to be more robust against different data types.

Text detection has been speedily developed in recent years. Large datasets are acquired for training and testing system for text detection. Advanced algorithms are proposed to generate features for text and to separate them from backgrounds, thanks to the development of machine learning algorithms.

In complex scenes, text detection is made difficult by large variations in color, font, size, and orientation of characters; Inhomogeneous lighting condition introduces highlight, shadow, reflection and color offset; Camera introduces additional noise, blurring and viewing angle distortion. Different types of noises and variations in different stage of image acquisition process are listed in Table 2.1.

As a result of progress in machine learning [15] and content based image retrieval (CBIR) [16], more generalized and automatic systems are developed for text detection. In most recent studies, machine learning methods are preferred over heuristic rules. With parameters and thresholds inferred automatically from training data, as in Figure 2.2, it requires less human intervention, generally increases the accuracy and robustness of text detection.

Previous systems used different text detection strategies, either step-wise or 'holistic', as mentioned by Ye et al. [14]. Step-wise text detection is composed of different components, including localization, verification, segmentation and sometimes recognition. 'Holistic' methods combine results from different stages, often applying OCR results for use in lexicon matching.

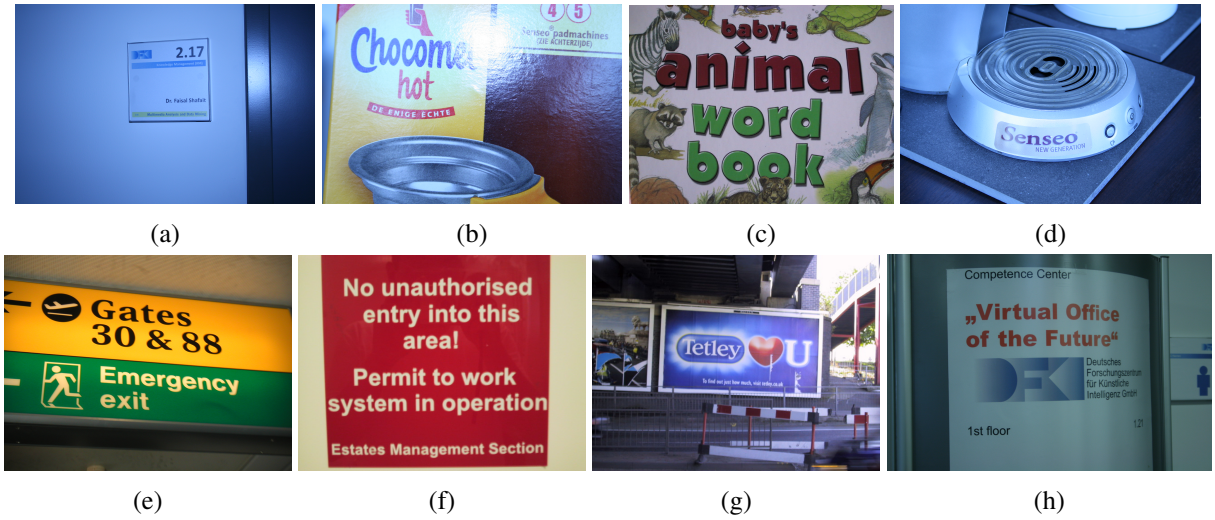
Examples of complex images from natural scenes are shown in Figure 2.3. Figure 2.3a shows that characters are in different colors and sizes, while some characters are surrounding other characters. Figure 2.3b shows that image contains occlusion and a single word is divided by background noise. Figure 2.3c shows where characters are unrecognizable due to high-lighting. Figure 2.3d illustrates a case where image contains complex background patterns. Recent researches found ample rooms for improvement in text detection and recognition for natural scene images. Current state-of-art systems obtain relatively low accuracy for complex cases (about 80% recognition rate). One survey on text detection and recognition has been conducted as in [14].

2.1.3 Terms Definition

In text detection systems, a standard pipeline is usually composed of several components, in order to extract information and convert to codes that computers understand. Starting from raw images or video frames,

Table 2.1: Common challenges faced by systems for text detection and recognition of natural scene images.

Category	sub-category
environment	scene complexity
	uneven lighting
image acquisition	blurring & degradation
	perspective distortion
text content	variation of aspect ratio
	multi-oriented & curved text
	variation of fonts
	multi-lingual environment



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

(e) (f) (g) (h)

Figure 2.3: Natural scene image samples from ICDAR 2015 robust reading competition. High variation of targets and background makes text detection problem challenging and interesting. The dataset contains a large variety of image samples. Camera based pictures are taken indoor and outdoor with texts from commercial signs, book covers, and brand names etc. Lighting condition creates artifacts like highlights and shadows which made the text harder to be detected. Different viewing angles introduce transformations, including rotation and skewing. Texts are with different colors, sizes, positions and fonts. Therefore, text detection in natural scene is a much more difficult problem than in document image.

preprocessing are usually used to rectify image from skewing and rotation, enhance contrast, sharpen edges and remove noises. The second step is *detection* of the position of the texts, including finding either the words/characters bounding boxes or pixels. The third is to convert information into computer code, aka. *recognition* of text.

In the literature, various stages of the tasks are referred to by different names, including *localization*, which aims to determine the position of candidate text; *detection*, which determine whether text exist in

images; text information extraction, which combines localization and binarization; text *enhancement*, which is used to rectify distorted text or to improve resolution prior to recognition; *scene text recognition* and *text recognition in the wild*, which focus on complex scene captured in nature. Although they are in different names focusing on different aspects and datasets of text detection, they are all lay in the scope of text detection.

Detection and *localization* are often used interchangeably for the same task in different cases. However, sometimes, they are referring to different problems even though techniques in the backend addressing these problems are similar. Text detection usually refers to finding texts in multiple images, where text may or may not appear in each image. Detector will classify images as containing text or containing no-text and then find the location of text in the earlier group. On the other hand, localization of text means finding the position of text on the premise that image contains text. The term *word spotting* is also used referring to finding the location of the word in a complex image. This term is likely to be confused with word retrieving problems and natural language processing (NLP).

In the next section, we start with methods for data acquisition and ground truth generation. As different datasets and ground truth heavily influence the methods used for detection, their training process, and evaluation methods. It defines the problem and provide foundations for algorithms to build upon.

2.2 Ground Truth

Text detection studies if image contains desired target text, if exits, then where the location is. Although rarely discussed, the quality of Ground Truth (GT) is important for training and testing a text detection system. Ground truth contains image and label pairs. Each image file has a corresponding label file indicating the location of the text. One of interesting research question is regarding to utilizing different level of ground truth, including bounding box level and pixel level to enhance training of detector.

2.2.1 Level Hierarchy

Bounding box level and pixel level ground truth labeling methods are mostly used for text detection systems. Example images of ICDAR robust reading competition ground truth are shown in Figure 2.4. As from the examples, bounding box ground truth could sometimes fail to provide sufficient information for text

detection system's training, while get low accuracy for text detection evaluation. Figure 2.4a shows if a character is contained by another character, the bounding-boxes of the two are overlapped. It is difficult to use the bounding box information to evaluation text localization performance correctly. If a word is not written in the up-right direction, a bounding box could also provide poor representation of the location of the text. As shown in Figure 2.4e, the rotated and tilted word 'UPGRADE' has a larger bounding box than other words, and a larger area inside the bounding box contains background pixels. It is also hard to label a word with characters with different sizes, as in Figure 2.4f, with bounding boxes. In those cases, pixel level ground truth could provide ampler information and more accurate location than bounding boxes.

Bounding Box Level. Bounding boxes file contains the x, y coordinates of corners and widths/heights of bounding boxes. The bounding box ground truth can be labeled in paragraph, text line, word and characters levels for different ground truth generation systems, as in Table 2.2. The finer region representation usually produces better ground truth. On the other hand, some label ground truth by categories, as text, graph, table, noise etc. Some systems just label the target objects as foreground and the rest as background.

Pixel Level. Pixel level ground truth are label maps, where 0 or 1 value indicates background or foreground. For multi-class labeling, colors are used to indicate class label. For pixel level ground truth, blurred edges brought ambiguous regions between text and background while no standard thresholding value is available. Papers [17] [18] [19] indicate that labeling these pixels (finding the right stroke width for text) plays a critical role in training and evaluation of text detection systems. Some systems utilize character skeletons to evaluate the text detection result. The advantage of skeleton based ground truth is that they are free of variation of character stroke width variations. The evaluation metrics based on skeletonize ground truth only consider whether the characters are located correctly, but not their total overlap area.

A major research question is if a combination of two level ground truths could lead to a better performance for text detection systems. In the case where both bounding box and pixel level ground truth available (e. g. ICDAR competition), how to combine these two type of information strategically is interesting. Not much previous work has been done using both kinds of ground truth. In our thesis, we design a two stage classification algorithm, using different ground truth in patch based and CC based classifiers, respectively. This design empowers us to detect text in complex scenes with very high accuracy, which is a novel approach.

(a) (b) (c)

(d) (e) (f)

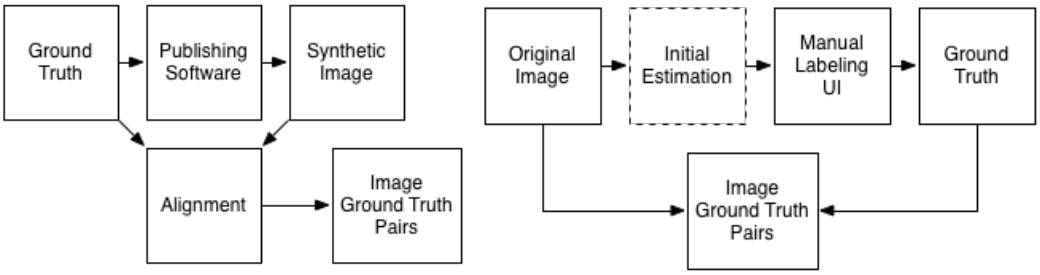
Figure 2.4: Examples of images from ICDAR competition 2013 with bounding box (a-c) and pixel level (d-f) ground truth.

2.2.2 Labeling Methods

Some work dedicated to producing high quality GT for different types of images, and made great examples of combining the power of computer automation and human guidance to increase accuracy of ground truth labeling. A good GT gives high evaluation result to a desirable detection, and low to the undesirable. A poor GT might do the opposite or just can't differentiate. Different systems might utilize different representations

Table 2.2: The ground truth labeling systems, include image synthesize systems using ground truth information and manually labeling systems for given image data.

Data Type	Generation System
Natural Scene	LabelMe [20]
	Nieuwenhuis [21]
	Zhou [22]
	Chen [23]
	Yao [24]
	Qu [25]
Google Street View	Wang(SVT) [26]
Document	PARC PixLabeler [27]
	Bunke (ALETHEIA) [28]



(a) Synthetic Image Using Ground Truth (b) Labeling Ground Truth Using Image

Figure 2.5: The Ground Truth generation methods: (a) Create synthetic images from ground truth; (b) labeling ground truth from images manually. Images and ground truths are in pairs where each ground truth file contains text location information for each corresponding image.

of ground truth that best suits their purposes. Usually, finer ground truth could provide better training and evaluation.

The GT generation have two main categories: *Synthetic method* utilizes label to generate data images; *manual Labeling* gets labels from the existing images, as in Figure 2.5. It creates images firstly (using camera), then generate ground truth by labeling each part of the images manually or semi-automatically. The *synthetic* method first creates the ground truth, then generate corresponding images. Synthetic and manual labeling methods are listed in Table 2.2 along with corresponding datatypes. Synthetic or manually labeling method has its own pros and cons, as discussed below.

Synthetic. For document, a lot of systems proposed to synthesize images from ground truth. Therefore, a large amount of GTs and corresponding images could be generated very quickly without human intervention. These images include newspapers, historical document and other types of documents.

Synthetic method is usually appropriate to generate print text with restricted formats. Hand-written text

or text in wild are rarely generated this way. In a synthetic dataset, one problem is to align the ground truth label and its rendered images [29], instead of to label ground truth itself. Some common challenges in synthetic methods are:

- How to generate text with required content, formats and layouts, including paragraph layouts, figure insertion, page layout etc.;
- How to build a system that correctly simulates print and scan process [30]. The problem can become harder for some specific types of images, like historical documents, which asks for simulation of historical degradation;
- How to align the ground truth with generated images, which usually contains scale, rotation and skew transformation;
- How to compensate character degradations. As noise is introduced into images, to reflect the changing in pixel values within characters becomes a problem. Some character are broken, touched or transformed in synthetic process, the ground truth in pixel level needs to reflect this change;
- How to combine automatic synthesization and user correction in a sensible way.

Some examples of GT generation systems are listed in Table 2.2. Some systems use publishing tools as their generator of image data. The publishing can be rendered on the fly (i.e. WYSIWYG word processor) or compiling afterwards (i.e. LaTeX).

Heroux et al. [31] proposed an approach for the automatic generation of synthesized document images and associated ground-truth based on a XML publishing tools called *TrueViz*. It generates rich information, including position and dimension, type of content, logical label, font attributes, and locations.

Zi et al. [32] proposed simulating the print and scanning for different kinds of languages. Any language supported by Microsoft Windows Enhanced Metafile Directives could be generated using this system. Newspaper ground truth generation method is introduced in Strecker et al.'s work [33]. The newspapers are generated in a way that mimics the real layouts. Print and scan process is simulated bringing noise and errors. The ground truth and scanned copy are aligned by feature alignment, and the position of each character is recorded in XML file. Yang et al. provide a system for semi-automatic generation of ground truth for chart images [34].

Manual Labeling. To detect more complexed targets, e.g. hand-written or natural scene symbols, synthetic method has difficulties modeling targets and backgrounds. Most of the labeling systems for this problem require user to label the pixels manually or semi-automatically. Besides synthetic methods, there are manually labeling methods for traditional document images, which provide alternative options. Text-labeling will be discussed with other objects labeling interface comprehensively in this section, as they always refer to each other. Common operations used by manual ground truth labeling user interfaces are: Click Selection, Brush, Rectangle Selection (Rubber band), Polygonal Selection (Lasso), Layer Locking, Highlighting Unlabeled Region and Undo/Redo. Systems that uses manually labeling method are listed in Table 2.2.

Ambiguous Region. The ambiguous region was described as Transition Regions by Ramirez-Ortegon et al. [35]. Transition method based Normal Threshold (NorT) is used to binarize historical document images. It shows that NorT method is robust against inaccurate estimation of transition proportion. Transition proportion is then used as a feature to detect region of interest and to estimate stroke width. The transition portion is ambiguous region: As nobody could define the pixels inside the transition region as foreground or background, using them for training or evaluation is meaningless.

Multi labels. Another approach to utilizing ground truth for image is to mark pixels with multiple labels rather than two. In Kopec et al.'s paper [36], regions that are labeled as confident text (Class 1) or background (Class 2). Blurred regions are labeled as ambiguous text (Class 3) and ambiguous background (Class 4). Therefore, the final evaluation is performed considering all types of regions. This fuzzy labeling method provides a flexible evaluation for images with blurs, noise and uneven illuminations. If a binarization ground truth is needed, a weighted sum and thresholding of the multilevel ground truth is computed. The weights are from a HMM classifier trained with binary ground truth.

Snyder Labeler. A pixel level ground truth labeling method is proposed by Snyder. The labeling method utilizes a color segmentation results and edge detection results. Firstly, the image is color segmented to form candidate regions, then people can label the text by clicking on the desired area [37].

The color segmentation is performed by K-means, while the number of clusters is tuned in real time to get the desired classification result. A smoothing algorithm is to remove small objects such as noises. It is a bilateral filter with varying window sizes and σ values. The user can tune the parameter of the bilateral filter until he found an acceptable noise removing result. Then he or she is able to see the color segmentation

results in real time by changing the number of the K-means. Different segments are indicated by different colors in segmentation maps.

Once the user thinks color segmentation result acceptable, he/she can select the regions of interest by clicking the candidate regions with his/her pointing device (mouse). The selected region will be marked as the text automatically. Then he/she will move to the next region and click again. After he/she finished all the text regions, one could click the finish button and a preview of the binary ground truth labeling map will be shown. One can modify ground truth or save the ground truth labels and move to the next image. Using this procedure, one could quickly label a large amount of images in pixel level and provide good training data for automatic detectors.

ALETHEIA. In Bunke et al.'s work [28], the ground truth generation method for layout analysis competition is introduced. This method is the base for competition of ICDAR 2003 and ICDAR 2005 layout analysis [38]. Objects are followed, such as accuracy, richness of information, efficiency of comparison, ease of understanding, ease of creation, and anticipation of future requirements. The user interface contains: individual selection, drag and resize operation (rubber band), polygonal operation (Lasso). The system is called *ALETHEIA* and designed for document and newspaper images.

LabelMe. Russell et al. [39] [20] proposed a user interface to label objects in scene images conveniently. An object labeling system is built and introduced called *LabelMe*. The system is web based, allowing people from all-of-the-world to label objects collaboratively. The people need to mark polygonal boundaries of objects and label the objects by name. A semi-automatic labeling method is included, where a Support Vector Machine (SVM) classifier is used to label objects automatically once user selected a rectangle region of interest. This system has already been widely used in many image understanding and retrieval systems [40] [41] [42] [43].

PARC PixLabeler interface is proposed by Saund, E. et al. [27], where a sophisticated but easy-to-use pixel labeling tool is introduced. The initial labeling could be achieved automatically or manually. The connected components are labeled with different colors. People could also label objects in brush mode or selection mode. The automatic structure group will detect horizontal and vertical lines, so people could select them later with a single click. The system also allows multiple users to label the same image and get higher accuracy using a voting mechanism. The system shows unlabeled pixels to let people label them more quickly. And layer locks are used to prevent errors. The system is implemented in *Java/Swing*. The

ground truth is stored in *PNG* and *XML* containing label group information.

Interactive. Interactive image segmentation systems are also capable of generating ground truth for object detection system or text detection systems. In Nieuwenhuis et al.'s work [21], joint distribution over color and spatial location are estimated. Parzen density estimator is applied to each user scribble. Bayesian Maximum A Posteriori (MAP) estimation approach is used alongside with convex relaxation techniques. Therefore, users can add simple scribbles to enhance the performance of the segmentation, until desired segmentation result is achieved. This kind of ground truth generation method will segment images into foreground and background objects with simple user scribbles, generating large quantities of images with labels.

A collaborative framework is designed by Zhou et al. [22]. The images are originally obtained with some of them labeled. To increase the dataset, a large amount of images need to be labeled automatically. The system computes the possibilities of the labels by analyzing the existing labels and visual features. The system could predict the labels if no label exists using visual features. It achieves much improved accuracy once a few labels are provided for the images. Chen et al. [23] proposed pixel accurate document image content extraction, in which the pixel accurate contents are extracted from document images automatically. The system utilizes color quantization, shape features and relationship between objects to classify objects as text, figures and background. People could fix the wrong segmentation regions manually to achieve close to perfect accuracy.

Another general purpose ground truth database which could benefit text detection is proposed by Yao et al. [24]. This system firstly divides the image into several meaningful regions and let users to mark each region with annotation. It combines bottom-up and top-down methods to accelerate the labeling process. Users could label small objects first which leads to the fully understand of the current image, then objects from labeled image could be used to guide segmentations of new incoming images in a top-down fashion. This process produces a hierarchical structure of the objects in images as ground truth, in an AND-OR graph knowledge base. They generated 636, 748 images and video frames using this method, including common scenes, aerial images, generic objects, faces, et al.

A Street View Text (SVT) dataset is generated by Wang et al. [26]. Images are harvested from Google Street View and only business signs are collected and marked in the ground truth. The ground truths are labeled with bounding box information. The bounding box is not exactly around the text regions. Labeled

bounding box areas are usually larger than optimal.

With ground truth generated using synthetic, manual and interactive methods, they can be used to train, test and evaluate the performance of text detection systems. In the next section, we will discuss the evaluation methods for text detection and corresponding metrics. Along with ground truth, evaluation methods provide bases for text detection algorithms, and influence how one should attack these problems.

2.3 Evaluation Methods

As discussed in previous section, bounding box and pixel level ground truth are commonly used by different text detection systems to evaluate their performance from different aspects. Different metrics are proposed for evaluation using different ground truth.

2.3.1 Level Hierarchy

Similar as ground truth labeling, the performance of the system can be evaluated in different levels. The evaluation level is chosen by different systems, based on kinds of ground truth available, application and other factors. Common levels of evaluation include bounding box level and pixel level.

Bounding Box Level. Bounding box level evaluation is widely adapted by a major portion of text detection systems, due to its simplicity.

In Mao's [44] empirical study, they choose University of Washington Dataset. The bounding box for the each textline is defined. They have proposed a five-step performance evaluation methodology for evaluating page segmentation algorithms.

Pixel Level. In Stathis et al.'s paper [45], local vs. global binarization methods are compared. The evaluation metrics are Pixel Error Rate (PERR) and PSNR. The values of these metrics indicate system accuracy, and variations indicate stability. Global methods include Otsu, Histogram peaks, K-means, Pun's, Kittler and Illingworth's method. Local methods include Kohonen SOM, Niblack, Abutaleb, Palumbo, Li and Liu's method et al. The ground truth is well-defined black and white document images, and detection target is noisy images generated using ground truth by adding random noise.

Sarkar et al.'s work [46], the classifier used pixel labels. The document image is firstly binarized into CCs. And a segmenter is used to separate the pixels into different CCs. Two-stage classifier is used to

classify the CCs into different categories, print, handwritten, background and noise. The evaluation is done in terms of precision, recall and error rate of pixel level.

The pixel accurate ground truth is used by Shafait et al. [47] to evaluate the page segmentation performance. Each label in segmentation result or ground truth is considered as a node, the edge weight is computed using the overlapped foreground pixels inside the nodes. Total over/under-segmentations, over-segmented components, under-segmented components, missed components and false alarms are used as evaluation metrics.

Sezgin et al. [48] compared image thresholding techniques, using pixel level ground truth. Text detection algorithms are compared, including histogram based, clustering based, entropy based, object attribute based, spatial correlation based, and local information based methods.

In Stathis et al.'s paper [45], miss-classification error, edge mismatch, relative foreground area error, modified Hausdorff distance, region non-uniformity is used for evaluation. Four evaluation metrics are used in Smith et al.'s paper [18]: F-measure, negative rate metric (NRM), Peak Signal to Noise Ratio NR (PSNR) and normalized cross correlation (NCC).

2.3.2 Metrics

Usually, text detections are evaluated in terms of *precision*, *recall*, *F-measure* and *Area Under Curve* (AUC): The *precision* is defined as the percentage that retrieved instances that are relevant; The *recall* is defined as the percentage that relevant instances that are retrieved; And *F-measure* is the harmonic mean of the precision and recall, , as in equations below:

$$\text{Precision} = \frac{\text{relevant instances} \cap \text{retrieved instances}}{\text{retrieved instances}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{relevant instances} \cap \text{retrieved instances}}{\text{relevant instances}} \tag{2.2}$$

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.3}$$

The AUC is defined as the area under the precision vs. recall plot, while changing the prediction thresh-

Table 2.3: Text detection evaluation systems, datasets used, ground truth, and corresponding evaluation metrics. (BB, Bounding Box; UW, University of Washington Dataset; NIST, National Institute of Standards and Technology; MARG, Medical Article Records Ground-truth; NTI, Non-destructive Testing Image; SVHN, Street View House Numbers; Over/Under-Segmentation includes total over-or-under segmentation, over-or-under segmented components; ME, Mis-classification Error; EMM, Edge Miss-Match; RAE, Relative Foreground Area Error; MHD, Modified Hausdorff Distance; NU, Region Non-Uniformity; PERR, Pixel Error Rate; PSNR, Peak Signal Noise Ratio; NRM, Negative Rate Metric; NCC, Normalized Cross-Correlation.)

Ground Truth Level	**Systems**	**Dataset**	**Evaluation Metrics**
Word BB	ICDAR [49] [50] [51] [52] [53]	ICDAR Dataset	Precision, Recall, F-measure
	Zhu [54]	US Patent Diagram Images	Precision, Recall, F-measure
Textline BB	Mao [44]	UW	Error Rate
Pixel	Sarkar [46]	NIST, MARG	Precision, Recall, Error Rate
	Shafait [47]	UW III	Over/Under-Segmentation, Miss, False Alarm
	Sezgin [48]	NTI, Document	ME, EMM, RAE, MHD, NU
	Stathis [45]	Document, Noisy	PERR, PSNR
	Moll [17]	Document, Line Art	PERR, Precision, Recall
	Smith [18]	DIPCO [55]	F-measure, NRM, PSNR, NCC
Character Labels	GoodFellow [56]	SVHN, reCAPTCHA	transcription accuracy

old. The AUC is a comprehensive measurement of system performance, considering both aspects of the system whenever a high precision or a high recall is preferred. It represents an average detection performance throughout entire dynamic range of classification threshold.

Other metrics, such as *Correlation based*, *shape based*, *information based* and *skeleton based* metrics are usually used by different systems. A comprehensive list of text detection systems, their target datasets and corresponding evaluation metrics are shown in Table 2.3.

Correlation Based Metrics. This kind of evaluation shows how many pixels (or how much bounding box areas) overlaps between detector's prediction and ground truth. The correlation based metric is straight forward: the more agreement between prediction and ground truth, the better performance. However, as no structure information is considered in those metrics, it could not separate different kinds of errors. For example, system I contains more false positive pixels only because it assumes wider stroke width of the text; system II contains less false positive pixels, but all of them are far away from text regions (noise objects). So which system should we choose? A meaningful choice should be system I, but system II will be suggested by correlation based metrics. Common correlated based metrics includes *Normalized Cross-Correlation (NCC)*, *Edge Mismatch* and *Relative Ground Area Error*.

Usually NCC is used to represent the correlation magnitude between prediction and true labels. It is defined as:

$$NCC = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M}(I_1(x,y)-\bar{I_1})(I_2(x,y)-\bar{I_2})}{\sqrt{\sum_{x=1}^{N}(I_1(x,y)-\bar{I_1})^2\sum_{y=1}^{M}(I_2(x,y)-\bar{I_2})^2}} \tag{2.4}$$

where higher NCC indicate a better match. Mis-classification Error is defined as,

$$ME = 1 - \frac{|B_0 \cap B| + |F_0 \cap F|}{|B_0| + |F_0|} \tag{2.5}$$

where B_0 and F_0 denote the background and foreground in ground truth. B and F denote the background and foreground in the detectors prediction. The negative rate metric (NRM) is defined as:

$$NRM = \frac{NR_{FN}NR_{FP}}{2} \tag{2.6}$$

where NR_{FN} and NR_{FP} are false negative rate and false positive rate respectively.

Edge Mismatch only focuses on edge pixels instead of all region, which is defined as,

$$EMM = 1 - \frac{CE}{CE + \omega[\sum_{k\in EO}\delta(k) + \alpha\sum_{k\in ET}\delta(k)]} \tag{2.7}$$

$$\delta(k) = \begin{cases} |d_k| & \text{if } d_k < \text{max distance} \\ D_{max} & \text{otherwise} \end{cases} \tag{2.8}$$

where CE is the number of Common Edge pixels between ground truth and prediction. EO is the missing pixel; ET is the false alarm pixel. ω and α are penalty ratios. d_k is the Euclidean distance of excess edge pixel to the nearest complementary edge pixel. Instead of measuring the entire region, this metric measures the edge difference between prediction and ground truth.

Relative Ground Area Error is defined as:

$$RAE = \begin{cases} \frac{A_0 - A}{A_0} \text{ if } A < A_0 \\ \frac{A - A_0}{A} \text{ if } A \geq A_0 \end{cases} \tag{2.9}$$

where A and A_0 indicate the foreground region of prediction and ground truth respectively. This mea-

surement focus only on foreground area, which is suitable for 2-Class problem. This metric bias towards foreground, because for detection problems, foreground regions usually occupy much less areas than background but with much higher interest.

Shape Based Metrics. On the other hand, shape based metric consider the shapes and uniformity of the detected region. The systems who produce detection regions more likely to be text are awarded, and systems who produce a lot of noise regions distributed across the image are punished. Usually, shape based will be used along with correlation base metrics to form a complete representation of detection performance. Common shaped based metrics includes Region Non-uniformity and Modified Hausdorff Distance.

Region non-uniformity is defined as,

$$NU = \frac{|F|}{|F+B|}\frac{\sigma_f^2}{\sigma^2} \tag{2.10}$$

where σ and σ_f denote the variance of the whole image and variance of the foreground region respectively, $|F|$ and $|F+B|$ denote the total pixels in foreground and foreground/background combined, respectively. This metric indicates how uniform the detected foreground regions are.

Modified Hausdorff distance is defined as,

$$MHD(F_0, F) = \frac{1}{F_0}\sum_{f_0 \in F_0} d(f_0, F) \tag{2.11}$$

where $d(f_0, F)$ is defined as $max_{f_0 \in F_0}(min_{f \in F}||f_0 - f||)$, which is the maximum of minimum distances. This is a metric measuring the shape different between prediction and ground truth objects.

Information Based Metrics. Some metrics utilize the information theory understanding of the image and ground truth. The detection is considered as signals containing noise. Therefore, metrics like *Peak Signal-to-Noise Ratio (PSNR)* and *Entropy* are used to evaluate the detection performance. The PSNR is defined as:

$$PSNR = 10\log_{10}\frac{C^2}{MSE} \tag{2.12}$$

where C is the maximum fluctuation in the input image data type. Mean Square Error MSE is defined as

$$MSE = \sum_{x=1}^{N}\sum_{y=1}^{M}\frac{(I_1(x,y) - I_2(x,y))^2}{MN} \tag{2.13}$$

where M and N are image dimensions. And C is the difference between foreground and background colors. Information based metrics evaluate similar aspects of the detection results as correlation based metrics. It is usually a different view of the same problem.

Skeleton Ground Truth Metrics. When no 'perfect' ground truth could be achieved, the evaluation algorithm plays a major role to compensate difference in ground truth labeling and help find the best detection system in a sensible way. As precision could be decreased either by a falsely detect noise object or by edge disagreement of a correct object, we certainly want to punish the previous case heavier than the later one. Similarly, a lower recall could be caused by either a missing object, or thinner stroke width of text. We want the evaluation system to consider the previous false negative while later a meaningful detection.

Some systems are introduced to utilize imperfect ground truth. Ntirogiannis et al. [57] proposed an evaluation method of text detection. The skeletonized ground truth is introduced to enhance evaluation robustness. The original ground truth image is skeletonized, and user will need to modify the skeleton manually to remove spurs and noise. Then the precision, recall, and false alarms are redefined based on the skeletonized ground truth and its corresponding estimated ground truth edges. Recall is computed only using skeletonized ground truth and target binary result image. Precision is computed using estimated ground truth and edge image. False alarm (FA) is computed using the skeletonized ground truth. If the result image CC is not overlapped with any skeletonized GT, then the number of pixels will be added to the FA rate. Finally, FA rate is normalized by the total number of detected pixels. The stroke width is estimated using Canny edge detector. The method is used to evaluate different binarization methods, which are pixel level text detections for document images.

The ground truth and metrics are with great interest for text detection. A well labeled ground truth helps both training and evaluation. Usually pixel level could help get better accuracy, thanks to its detailed information. Different evaluation metrics are discussed, including correlation based, shape based and information based. These metrics can be used for either bounding box or pixel level ground truth for evaluation. Correlation and information based metrics evaluate similar aspect of the prediction from different point of view. Shape based evaluation adds shape uniformity explicitly to regularize the detection map. In our experiments, ICDAR 2015 evaluation method is used, which will be discussed in more detail in Section 2.6. Precision, recall and F-measure is computed in bounding box level. One-to-multi and multi-to-one matching are also modeled with penalties. This penalty in bounding box level is served as a regularization factor

similar to shape based metrics.

2.4 Features

For text detection, color, edge, texture and shape features are used in conventional systems. Stroke, point, region and appearance features have been recently explored. Some examples of previous systems are listed in Table 2.4 along with features and classification algorithms used.

The information that used for text detection could be categorized into two groups: visual features and lexicon models. The first group represents the appearance of the image and the later one includes additional knowledge of grammar, spell, and other language statistical information. Our research is interested about if using visual features alone is sufficient for high-accuracy text detection? There are many systems proposed for text detection using only visual features [58, 59, 60]. But there are also systems adding lexicon models upon visual feature detectors to enhance performance [61, 62]. The latter group usually tends to get better performance, as more information has been used and extra efforts have been put for a higher accuracy. However, in many cases, the lexicon information is not available (i. e. foreign language, data with only visual ground truth). Therefore, a system dependent to visual features alone is interesting, because it can be applied to a wider range of applications. Therefore, in this thesis, we only consider visual features for text detection.

2.4.1 Feature Design

Multiple features are proposed for text detection including colors, textures, edge/gradients, stroke width and Maximal Stable Extremal Region (MSER). These features are to capture different facets of target's characteristics and often used together to build a stronger classifier.

Colors. Text is often produced in a uniform but distinguishable color so that it can be perceived by human eyes easily. Color features are widely adapted in many text localization methods. In complexed images, using color information alone is not accurate or robust enough to correctly find the texts. In cases of uneven lighting or text in multiple colors, the color features performance is degraded seriously. In order to increase robustness and adaptive to human perception, color space transformation is often performed. Different color spaces, like CIEL*a*b* or HSV could also be mixed to enhance accuracy.

Table 2.4: A partial list of text detection systems proposed in the past decade. The detection systems could be categorized into two groups. One group uses simple classification method, like single threshold and heuristic rules. Another group use intelligent and adaptive methods, based on machine learning algorithms, like SVM and AdaBoost. To enhance performance, spatial relationships, recognition results and lexicon models are often used. Features designed specifically for texts are also listed, like SWT or MSER. Features commonly used for generic object detection systems, like HoG and Haar features are also used by some systems to locate texts.

Algorithm	System	Features	Year
Heuristic Rules	H. Chen [63]	MSER objects	2011
	L. Neumann [62]	MSER objects	2012
	Epshtein [58]	SWT	2010
	Gao [64]	edge, color	2001
	Clark [65]	edge, texture	2002
	D. Chen [66]	edge	2003
	Sharma [67]	edge, texture, orientation	2012
	Yi [68]	gradient, color	2011
	Halima [69]	projection, shape	2013
	Thillou [70]	orientation (Log-Gabor)	2007
	Leon [71]	hierarchical region geometry information	2013
SVM	Kim [59]	texture	2003
	X. Wang [72]	contrast, HoG	2010
	Petter [73]	edge	2011
	Coates [60]	unsupervised learned features	2013
AdaBoost	X. Chen [74]	Haar wavelet, intensity, edge	2004
	Zhu [75]	geometry, shape	2005
	Hanif [76]	texture (SD), HoG	2008
	Neumann [77, 78]	MSER	2013
CNN	He [79]	MSER	2015
MRF	Shen [80]	edge	2006
NMS	K. Wang [81]	HoG	2011
WaldBoost, CRF	Pan [82]	HoG	2011
MLP	Chowdhury [83]	Sparse Stroke Width	2012

SVM, Support Vector Machine; MRF, Markov Random Field; SD, Standard Deviation; HoG, Histogram of Gradient; SWT, Stroke Width Transform; MSER, Maximally Stable Extremal Region; NMS, non-Maximal Suppression filtering; MLP, Multi Layer Perception; CNN, Convolutional Neural Network.

Texture. As characters in text lines usually appear to have fixed intervals, texture are used to address this characteristic. Texts are considered as textures and analyzed in frequency domain. This includes features based on Fourier Transform [76], Discrete Cosine Transform [59], Wavelet Transform, Local Binary Patterns (LBP) and Histogram of Gradient (HoG) [81, 82]. It is often used with multi-scale sliding windows to detect and localize texts. Texture are useful for textline and sentences, but not as effective against isolated

characters.

Edge/Gradients. Edge or gradient features are also used to find text under the assumption that characters exhibit strong and symmetric edges against background. In practice, edge features usually combined with color, size and positions. Comparing to color, edge features are less sensitive to luminance inhomogeneity or different colors of characters. However, for complex scenes, where many other objects can also exhibit strong edges (e. g. windows, fences, leaves, graphs), edge features may have difficulties to find correct targets.

Stroke Width Transform. Epshtein etc. proposed a famous feature for text detection in natural scenes in 2010 [58]. The Stroke Width Transform (SWT) measures the stroke width of different objects based on an edge detection results and try to filter out object with non-uniform stroke width. It assumes that text in natural scene, no matter their colors, orientations, sizes or fonts, usually maintain relatively uniform stroke width. On the other hand, background objects, like human, cars, sky and grass, usually have large or small stroke width, or have stroke width varies severely inside its boundaries. Especially for tree foliage, where other information could fall short to distinguish them from text as they usually get similar textures, stroke width value could separate them very efficiently and accurately.

The stroke width transform starts with an edge detection whose performance affects the accuracy of stroke width measurement significantly. Epshtein used Canny edge detector in this process. Then ray tracing is performed begin with each edge pixels and along the orthogonal direction of the edges. The initial value of stroke width is set to ∞. Then for each pixel p who lies on edges, a ray perpendicular to the orientation of the edge is formed and followed. The angle of the current edge direction is computed by the ratio of the local gray scale gradient value.

$$\theta = arctan(\frac{\delta_y}{\delta_x}) \tag{2.14}$$

where θ is the angle between y axis and the edge direction, δ_x and δ_y are the local gradient of x and y directions. And then ray grows in the perpendicular direction, as shown in the vector equation below:

$$\mathbf{r} = \mathbf{p} + n \cdot \mathbf{d_p}, n > 0 \tag{2.15}$$

, until another edge pixel $\mathbf{q}$ is found with roughly opposite direction. Where $\mathbf{r}$ represents the newest pixel

(a) (b)

Figure 2.6: Stroke width measurement using ray tracing [58]. (a) The pixel's stroke width value is set to be the minimum distance between two edges, as two different rays are intersected, the pixel value is chosen to be the length of the horizontal one. (b) The first pass of the ray tracing will set stroke width value to incorrectly large for pixel marked as red in the corner, the second pass will assign this pixel stroke width value to the smaller median value of the two rays, which is more close to the true stroke width value.

adding to the ray, $\mathbf{p}$ represents the original edge pixel, n is the ray current length and $\mathbf{d_p}$ is the unit vector point perpendicular to $\mathbf{p}$'s gradient direction. By roughly opposite, it means the direction of the two edges should be larger than $\frac{\pi}{6}$.

The distance between the two edges is then assigned to all the pixels along the ray as their stroke width, if it is less than their current values. If the ray does not find an opposite edge, the ray is discarded and no change will be made for the stroke width values of the pixels along the ray.

To counter problems near the corners, the SWT algorithm asks for another pass along the non-discarded ray again. Therefore, along each ray in the first pass, the median of the SWT value would be computed. Then all pixels along the current ray will be set to the median value if they are originally larger than it. This second pass is used to address miss-measurement of the stroke width having a situation like in Figure 2.6, where the first pass stroke width value is much larger than its real width value.

Maximal Stable Extremal Region (MSER). MSER based text localization has been widely explored. The main advantage of this representation is its effectiveness of extracting candidate region/components for text classification. As text components usually have significant color contrast comparing to backgrounds and tend to form homogenous color regions, MSER is used to find all these regions and use other features to eliminate false positives.

By searching for the most stable region, it could extract objects with uniform colors and clean edges. By looking for the minimum gradient value of area change when varying threshold, one could avoid leakage problem as in other segmentation algorithms, like watershed. Therefore, the region candidates are more

meaningful and robust. It was originally designed to extract any objects that have similar looking after affine or perspective transformations. However, nowadays, it is most commonly used for text detection due to the property of the text in natural scenes. Many systems in the ICDAR robust reading competitions utilize MSER or modified algorithms to generate candidates for regions of texts.

In our system, we used a flood-filling based region growing method to group pixels into CCs in the region of interest. This method is related to MSER, as both methods use the edge and color difference as references for region extraction. However, the cost function of our method and MSER is differed. While MSER considers the relative area change in each iterations, we focus on the color difference and edge intensity.

In practice, single feature might never achieve acceptable performance for text detection/localization. Usually multiple features are combined to retrieve texts from image. Some texts might have different colors, but with constant intervals. While some texts might have different sizes, but with similar stroke widths. Machine learning algorithms provide powerful tools for combining and weighting features based on their discriminative power.

2.4.2 Feature Learning

Unsupervised feature learning method, along with current widely studied deep learning algorithms provide a powerful approach allowing us to construct classifiers for text detection automatically through training. Feature learning provides many advantages that conventional knowledge and heuristic rule based systems could not offer:

- Data-driven, so the system is born adaptive to the target dataset. It weights the importance of each feature automatically to adapt to the target object and background noise through training process.
- Generate abstract and representative features automatically, while these kinds of features are hard to be designed by human.
- Easy to generate a very large amount of features allowing a modern machine learning classifier to be trained adequately, which is also hard if possible for a conventional prior-knowledge based system.

Popular feature learning methods are listed in Table 2.5, where they can be roughly categorized into probabilistic model, auto-encoders, graph model and clustering method.

Table 2.5: Feature learning methods [84] that was popularly used. Feature learning are usually unsupervised algorithm where features are extracted from the training data with abstraction. Features learned is representation of generalized data, where learning algorithm tends to generate highly averaged features that have low correlation between each other. PCA, Principle Components Analysis; MRF, Markov Random Field; rBM, restricted Boltzmann Machine; DAE, Denoising Auto-Encoders; CAE, Contractive Auto-Encoders.

Category	Methods
Probabilistic model	
Directed Graphical Models	PCA, Sparse Coding
Undirected Graphical Models, MRF	rBM
Auto-Encoders	
Regularized Auto-Encoders	Sparse Auto-Encoders
	DAE
	CAE
Manifold Learning	Neighborhood graph
	Nonlinear Manifolds
Clustering methods	Convolutional K-means

Unlike task specific assumptions often used in conventional text detection systems, Bengio [84] etc. proposed some common assumptions for deep learning machines, in order to perform efficiently. These assumptions are important for learning good abstract features which is more powerful towards difficult classification tasks. These assumptions are not only for deep learnings, but suitable for any feature learning algorithm to be discriminative and effective.

- Smoothness, as the low level feature values changed in a small amount, the higher level feature should also change in small amount accordingly. This ensures that features learned are consistent towards different sample values and will not alter classification result in a undesirable way;
- Multiple explanatory factors, means a reasonably sized learned representation can capture a huge number of possible input configurations. For easily separated samples, there might be a few simple features could be used to discriminate them into foreground and background; For hard samples, more explanatory features are needed, This way, samples can be separated using a limited number of features. Otherwise, the computation in execution can be highly complex and large quantities of features are required to achieve acceptable performance. Speed will be slow.
- Sparsity, the learned features should be a combination of several low level features. The sparsity ensures that the extracted features are independent of each other, as non-sensitive to small value

changes of other representations.

- Simplicity. The learning algorithm should be simple. The complexity of the learned system should come from data structure itself, not the learning scheme.

Deep Learning. One category of feature learning method, called Deep Learning, is prosperously developed in recent years. Although the initial concept of deep learning comes from Neural Networks, it could be simply understood as unsupervised learning of hierarchical features, where different stages of features represent different level of abstraction and composition. Thanks to the development of deep learning, many unsupervised feature learning methods are proposed for text [60] or other [85] object detection problems, which made possible for automatically generating adaptive features.

Auto-encoder. An auto-encoder is an artificial neural network based feature learning process. The aim of auto-encoders is to learn a set of representations that have minimum reconstruction error from the original data. Architecturally, auto-encoder is very similar to a multilayer perceptron (MLP) network. The difference is that instead of training to predict output labels, auto-encoders is trained to reconstruct its own inputs.

Auto-encoders often have two phases, encoder and decoder, which are defined as transitions ϕ and ψ. Therefore, we have,

$$\phi : \mathcal{X} \to \mathcal{F} \tag{2.16}$$

$$\psi : \mathcal{F} \to \mathcal{X} \tag{2.17}$$

$$\arg\min_{\phi,\psi} \|X - (\Psi \circ \Phi)X\|^2 \tag{2.18}$$

where ϕ and ψ denote the transition from sample vectors to basis of feature space (learning target) and the transition backward, respectively. $\circ$ indicates a combination of transitions. If the transition ϕ and ψ are linear and can be represented with matrix multiplications, then $\circ$ is the product of two transformation matrices.

If the feature space $\mathcal{F}$ has less dimensionality than input space $\mathcal{X}$, then the feature vector $\phi(x)$ can be regarded as a compressed representation of the input x.

Restricted Boltzmann Machine. Similarly, restricted Boltzmann machine (rBM) are often used to learn features from data. An rBM is represented by an undirected bipartite graph consisting of a binary hidden layer, a visible layer. Restricted means the model has no intra-node connections within the same layer, comparing to a more general Boltzmann machine.

The energy of a configuration (pair of boolean vectors) (v, h) is defined as

$$E(v, h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \tag{2.19}$$

where $W = (w_{i,j})$ (size mn) is a matrix of weights associated with the connection between hidden unit h_j and visible unit v_i. And a_i is bias weights (offsets) for the visible units, while b_j is for the hidden units.

In unsupervised feature learning, rBM is viewed as single layer architecture, where the visible variables correspond to input data and hidden variables correspond to feature detectors. The weights can be trained by maximizing the probability of visible variables using the contrastive divergence (CD) algorithm. Restricted Boltzmann machines are a special case of Boltzmann machines and Markov random fields.

Convolutional Neural Network. A convolutional neural network is a type of feed-forward artificial neural network, but individual neurons are tiled so that they are corresponds to pixel values in a perceptual plane. It is inspired by biological process of cat's retina and cortex, and then widely adapted by image recognition and detection algorithms.

Convolutional Neural Networks was established based on the so called neocognitron [86]. LeCun et al. improved the previous design and built the famous LeNet-5 [87]. CNN achieved great accuracy improvement towards handwriting digits, character and image recognition problems. CNN usually contains different layers to model the process of the function of biology eyes, including a convolutional layer for pattern matching, a Rectified Linear Units layer for non-linearity and a spatial pooling layer for translation compensation etc., as in Figure 2.7. CNN could be built so it contains multiple replications of composition of layers and make the system deep.

When used for image based classification, convolutional neural networks (CNNs) consist of multiple layers of small neuron collections which process portions of the input image, called receptive fields. The outputs of these collections are then tiled so that they overlap, allowing CNNs to tolerate small translation of the input image.

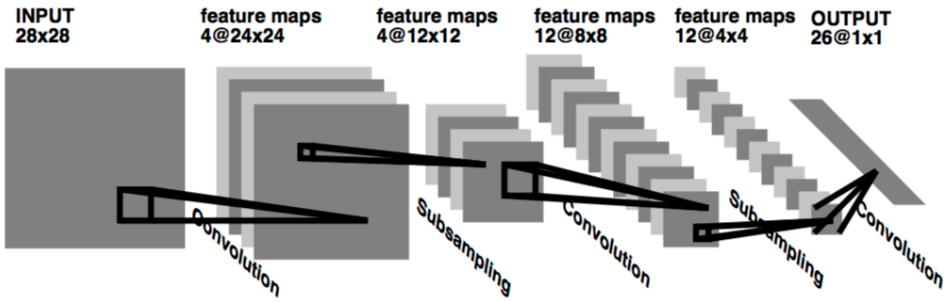


Figure 2.7: An illustration of Convolutional Neural Network from LeCun's website. The original image is convolved with codebooks (filters) to generate features. The pooling stage (subsampling) reduce the resolution of the convolution map, provide stability against small translation of characters. Multiple stages of convolution can be used in CNN and finally produce fully connected feature maps.

One major advantage of convolutional networks is the use of shared weights in convolutional layers, which means that the same filter (weights bank) is used for each pixel in the layer; this both reduces memory usage and accelerate performance.

Compared to other image classification algorithms, convolutional neural networks use relatively little pre-processing. This means that the network is responsible for learning the filters that in traditional algorithms were hand-engineered. The lack of dependence on prior knowledge and human effort in designing features is a major advantage.

Convolutional K-means Closely related to our own work, Coates et al. [60] proposed convolutional feature-based algorithms for text detection and recognition, and obtain improved recognitions rate compared to previous state-of-art systems. Wang et al. [61] extends this convolution-based approach by adding a lexicon model, which further increases text detection and recognition accuracy. Our system is partially based on their design, but with a more dedicated patch shape and procedure to enhance accuracy specifically for texts. In our thesis, we adapt Coates' method for feature learning. Similar convolution K-means algorithm is used. More details of convolutional K-means are discussed in Chapter 3.

Convolutional K-means, rBMs and auto encoders are algorithms that learn representative features from data. Those features are averages of sample data and distribute around the center of data clusters. However, one could learn another type of feature, called discriminative features. These features utilize label information, and distribute around the boundary between different classes. Combining both representative and discriminative features, stronger classifiers can be trained. This answers our third research question: Can

features obtained via convolutional k-means be made more discriminating? In experiment, we combine it with discriminative feature derived from support vectors, as shown in Chapter 3.

2.5 Classification Methods

The features for text detection are often used in two ways, heuristics or machine learning methods. Either way, the features can be used isolated or combined. Almost all systems introduced previously used multiple features for text detection, as single feature might never powerful enough for even a simple case detection problem.

Heuristics often used rule of thumb, educated guess or common senses to design threshold values and weights based on human expert knowledges. Fine tuning is needed to grid search best parameters for particular datasets. Therefore, the systems based on heuristics are very prone to overfitting. On the other hand, machine learning techniques allow classifiers to be built intelligently and adaptively. The automation saves enormous human labor while achieving greater robustness against varying image properties.

A large amount of systems has been proposed either using a heuristic method or machine learning algorithms, for document image analysis or text detection in wild scenes.

2.5.1 Heuristic Rules

A heuristic rules technique, often referred as heuristic, is an approach that employs a practical method not guaranteed to be optimal, but sufficient for immediate goals. Where finding optimal is impractical or not necessary, heuristics can be used to speed up and to find acceptable solutions. Strong prior knowledge and human expert opinions are often required for designing good heuristics.

In Sarkar et al.'s work [46], document images are analyzed using heuristic rules. Two stage classifier is used to classify CCs into different categories including print, handwritten, background and noise.

Sometimes, heuristic rule methods are applied to calibrate the power of newly proposed features for text detection. For example, L. Neumann et. al. [62] used heuristic rules for text detection tasks using their proposed MSER features. In Epshtein et al.'s work [58], heuristic rules are applied to test the ability of proposed SWT features.

Many other systems also used heuristic for complex scenes. Chen et al. [63] proposed to combine MSER

with SWT and other features for text detection of ICDAR images. They used heuristic and add features in different steps through a filtering pipeline. Gao et al. [64] used heuristic to detect text from natural scenes combining several features, including colors, layouts and language models. Focus-of-attention method is used to enhance the overall performance beyond previous systems. This system is used to translate languages into English with video cameras. Clark et al. [65] combines a heuristic based line searching method with neural network based texture feature classifiers to find text in complex scenes. The high level and low level cues are combined to rectify text from skewing introduced by perspective transformation. Chen et al. [66] proposed a text detection and recognition system in complex images and video frames. It is a two-step approach that combines speed of heuristic localization step with machine learning verification step. The experiments were conducted on large databases of real broadcast documents. Sharma et al. [67] used primarily edge features along with heuristic method to find text lines in different datasets. Yi et al. [68] proposed a system consists of two major steps: Image partition and character grouping. In partition, they used gradient features along with colors to find character candidates and then group them into words based on their spatial structure. They focus on textline and used heuristic to extract them from complex images. Mancas-Thillo et al. [70] used color features along with Log-Gabor filter features to find text using heuristic. Leon et al. [71] used texture and geometric features to detection captions.

2.5.2 Machine Learning

Machine learning are statistical algorithms that deal with pattern recognition, artificial intelligence and computational learning theory. They are computer programs that can learn from and make predictions on data. It often involves two steps, training and testing: In training, the data is used as guidance for algorithm to organize according to the current performance and make predictions; In testing, the prediction is compared with ground truth and errors are analyzed to instruct next iteration or step of learning process. It contains three major categories: supervised learning, unsupervised learning and reinforcement learning. These learning methods are often used for tasks like classification, regression and clustering. For text detection, features could be used by unsupervised learning. However, in this section, we focus on supervised learning methods that build the classifiers which separate image into foreground and background regions.

A variety of machine learning techniques have been used, including Convolutional Neural Networks (CNN) [87, 79], deformable part-based models [88], belief propagation and Conditional Random Fields

[89, 90, 82, 80], Neural Networks [65, 83], Support Vector Machines [59, 72, 73, 60], and AdaBoost [74, 76, 77, 78]. A summary of machine learning algorithms that was used by text detection systems is shown in Table 2.4.

Conditional Random Field. Conditional random field (CRF) is a class of statistical modeling method in machine learning. Ordinary classifiers predict labels without neighboring samples, a CRF takes context into account. CRF is a type of discriminative undirected probabilistic graphical model. In computer vision, CRF is widely used for object recognition and image segmentation.

A CRF is defined on observations $\boldsymbol{X}$ and random variables $\boldsymbol{Y}$ as follows:

Assuming $G = (V, E)$ as a graph, $\boldsymbol{Y} = (\boldsymbol{Y}_v)_{v \in V}$, so that $\boldsymbol{Y}$ is indexed by the vertices of G. Then $(\boldsymbol{X}, \boldsymbol{Y})$ is a conditional random field when the random variables $\boldsymbol{Y}_v$, conditioned on $\boldsymbol{X}$, obey the Markov property with respect to the graph:

$$p(\boldsymbol{Y}_v | \boldsymbol{X}, \boldsymbol{Y}_w, w \neq v) = p(\boldsymbol{Y}_v | \boldsymbol{X}, \boldsymbol{Y}_w, w \sim v) \tag{2.20}$$

where $w \sim v$ means that w and v are neighbors in G. This means CRF is an undirected graphical model whose nodes can be divided into exactly two disjoint sets $\boldsymbol{X}$ and $\boldsymbol{Y}$, the observed and output variables, respectively; the conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X})$ is then modeled. Markov property ensures the current iteration status $\boldsymbol{Y}_v$ is only decided by input (observed) $\boldsymbol{X}$ and previous iteration $\boldsymbol{Y}_w$ (neighboring nodes).

Felzenszwalb et al. [89] used belief propagation methods to solve early vision problems, such as stereo, optical flow and image restoration. They managed to increase the speed of the algorithm form quadratic to linear time, therefore, it can be applied to a large dataset practically. Shen et al. [80] used graph model based spatial relationship features to filter background objects using an MRF. Pan et al. [90] used CRF method to find text locations in ICDAR 2003 dataset. Both unary character properties and binary neighboring component relationship are used for accurate detection. Then texts are grouped into lines using an energy minimization approach. Another experiment is done towards ICDAR 2005 dataset and also shows great accuracy [82].

Support Vector Machine. Support Vector Machine (SVM) is a class of supervised machine learning algorithm that analyze data used for classification and regression. Given a set of training samples, each marked for belonging to one of two classes, an SVM training algorithm builds a model that assigns new

examples into one class or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the samples as points in space, mapped so that two classes are divided by a clear gap that is as wide as possible.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using which is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

We are given a training dataset of n points of the form $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the 'maximum-margin hyperplane' that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying:

$$\vec{w} \cdot \vec{x} - b = 0, \tag{2.21}$$

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors. where $\vec{w}$ is the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$.

For hard margin, we can put this together to get the optimization problem: Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$, for $i = 1, \ldots, n$. The $\vec{w}$ and b that solve this problem determine our classifier, $\vec{x} \mapsto \text{sgn}(\vec{w} \cdot \vec{x} + b)$.

For soft margin, we introduce the *hinge loss* function, $\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} + b))$. We then minimize:

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))\right] + \lambda\|w\|^2, \tag{2.22}$$

where the parameter λ determines the tradeoff between increasing the margin-size and ensuring that the $\vec{x}_i$ lie on the correct side of the margin. Thus, for sufficiently small values of λ, the soft-margin SVM will behave identically to the hard-margin SVM if the input data are linearly classifiable, but will still learn a viable classification rule if not.

Felzenszwalb et al. [88] used deformable part-based models to find location of text in complex dataset *PASCAL*. Latent SVM algorithm is used with discriminative training and partially labeled data to train the system. Kim et al. [59] proposed to use SVM classifier for text detection. They introduced Continuous Adaptive Mean-Shift (CAMSHIFT) algorithm to move the focus within the image by growing or shrinking the region of interest. Texture features are analyzed to extract text from images and video frames. Wang et al. [72] designed a system to find text in videos. The system firstly categorize the video frames into three categories: easy, median and hard. The easy text is located using stroke width information; conditional morphology is incorporated for median data; For hard data, SVM classifier is trained after previous two steps to reduce false alarms. Petter et al. [73] proposed a augmented reality (AR) system for language translation using text detection. They firstly used SVM to find single characters. And then use its location as cues to find whole words and sentences. Coates et al. [60] used convolutional K-means to learn features from ICDAR 2013 dataset. Then they used SVM classifier to locate and recognize texts. Their method of feature learning has a great influence on our system. And will be discussed in finer details in Chapter 3.

AdaBoost. AdaBoost, short for 'Adaptive Boosting', is a machine learning algorithm proposed by Freund and Schapire [91]. It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the basic learning algorithms, 'weak learners', is combined into a weighted sum that represents the final output of the boosted classifier. It is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances that was misclassified by previous iteration. In most of problems, it is less susceptible to the overfitting problem than other learning algorithms. The weak learners should be at least slightly better than random guessing, the final model can be proven to converge to a strong learner. AdaBoost is often referred to as the best out-of-the-box classifier.

More details about AdaBoost algorithm, its training and testing method, and its variations are introduced in Chapter 4.

Chen et al. [74] proposed a system for helping blind or visually impaired people read text in the city scenes. They combine weak features using AdaBoost algorithms. A cascaded architecture with 79 features and 4 strong classifiers is trained. Commercial OCR engine is used to recognize texts once detected. They achieved 90% accuracy on their own dataset acquired by blind people. Hanif et al. [76] used 39 features with AdaBoost classifiers to detect text in ICDAR 2003 dataset. Two weak learners, linear and log-likelihood ratio under Gaussian Assumption are used. This algorithm is fairly quick, which can process an image with

640 by 480 pixels in 2 seconds. Neumann et al. [77, 78] used AdaBoost for text detection in ICDAR 2003 dataset. The system has three improvements: Instead of a feedforward pipeline, they verify multiple textline hypothesis simultaneously; Synthetic fonts are used to train the classifiers; MSER features are used to extract candidate CCs.

Convolutional Neural Network. As CNN can both produce features from data and classify samples based on the feature it generates, many paper used CNN to detect texts in scene images automatically. Lecun et al. [87] proposed CNN to recognize handwritten characters, bank checks and other images. He et al. [79] used CNN along with MSER features to find text location in ICDAR 2011 and 2013 datasets. More details of CNN are discussed in Section 2.4.

As discussed above, many algorithms in machine learning are applied in text detection to achieve robust and accurate performance. We surveyed only part of a large amount of text detection algorithms that incorporate machine learnings. An accurate, adaptive and fast text detection system needs carefully selection of features learning and machine learning algorithms that best fits the problem and cooperate well with each other.

A research question is: Can word detection accuracy be improved by combining convolutional k-means detection with subsequent processing? In our experiment, we used a combination of different algorithms. Convolutional K-means is used to learn features from data. A modification of single layer convolutional neural network is combined with AdaBoost classifier for patch based detection. While, random forest is used for classification of character relationships. Another important factor is the utilization of contextual information, which is critical for the success of extracting region of interest for text.

In the next section, we discuss the famous competition in text detection area - ICDAR Competition. It has been hold in 2003, 2005, 2011, 2013 and 2015. It introduced very helpful datasets for text detection task in the wild and inspired many state-of-the-art text detection systems throughout the decade.

2.6 ICDAR Competitions

Over the last two decades, a number of text recognition competitions have been held as part of the International Conference of Document Analysis and Recognition (ICDAR) [1]. ICDAR Robust reading competitions have been hold in 2003 [49], 2005 [50], 2011 [52][53], 2013 and 2015 [1] and remarkably pushed

Table 2.6: Word Detection Metrics (ICDAR Robust Reading Task). We compare our system's performance with state-of-the-art system from 2013 and 2015 competitions. The top 3 systems for each competition are listed. We achieved higher recall, precision and F-metric values comparing to the best systems. No lexicon model or OCR is used to achieve this accuracy.

Method Name	Recall(%)	Precision (%)	F-score (%)
Our System	**81.02**	**93.39**	**86.77**
2015 Competition			
StradVision	80.15	90.93	85.20
Text-CNN [79]	76.95	92.78	84.13
VGGMaxNet [92]	77.32	92.18	84.10
2013 Competition			
USTB_TexStar [93, 94, 95]	66.45	88.47	75.89
Text Spotter [77, 96, 97, 78]	64.84	87.51	74.49
CASIA_NLPR [98, 99]	68.24	78.89	73.18

forward this area. Ashida, Becker, Kim, Yin and StradVision achieved best performance in text detection in 2003, 2005, 2011, 2013 and 2015 respectively.

In 2015, data sets are categorized into: Digital Born, Focused Scene, Text in Videos and Incidental Scene Texts. Each group contains three tasks: localization, segmentation and recognition, and end-to-end performance is also measured. An full summary of top 3 system performance from ICDAR 2015 and 2013 are illustrated in Table 2.6, along with our best results.

In ICDAR text detection competitions [49] [50] [51] [52] [53], bounding box information is provided to train, evaluate and compare participant systems. Participants are trained using the bounding box ground truth and evaluated in terms of precision, recall and F-measure.

The evaluation method using by the competition holder is computing the overlapping area of the predict region and ground truth region, then use them to compute true positive rate, false positive rate, etc. and then compute precision, recall and F-measure. The evaluation method has been improved in 2015 [1]. It checks the overlapping area for each detection with ground truth and compute the final precision, recall and f-measure based on the total number of words that are correctly detected. The bounding box of prediction should at least have a precision of 40% and a recall of 80%. As mentioned in the paper [1], many-to-one and one-to-many matching is also implemented for partially detected words or wrong segmentation, with a 20% accuracy discount as penalty.

At ICDAR 2015, the StradVision corporation obtained the strongest detection results for the Incidental Scene Text task. This system is closed, but is apparently based upon an active learning model.

He et al. [79] placed second, using two main improvements over earlier MSER-based text detection methods. First, they introduce 'Text-CNN', where a multi-purpose classifier is defined instead of a conventional binary (text/non-text) classifier. In each layer of a Convolutional Neural Network, specific labels and locations for text pixels define targets alongside binary foreground/background labels. The trained classifier is adapted to specific text types, and achieves higher accuracy as a result. Objects such as bricks, windows and bars that are easily confused with text are filtered, as they tend not to have a high classification confidence for a character class. Second, Contrast-Enhanced MSER is proposed to find text regions using a data-driven method. Text regions can be extracted in complex backgrounds and uneven lighting conditions.

Jaderberg et al. [100] placed third. To train their system, a very large corpus of labeled text images is generated using a font rendering engine. Noise and variations are added, including border, color, composition, distortion, and background blending, mimicking texts in 'wild' natural scenes. For detection, they use a deep Neural Network with three different encoding diagrams, including dictionary, character sequence and bag-of-N-gram encodings. The dictionary encoding methods shows the best performance, as lexicon constraints are used to improve accuracy (by pruning detections for invalid words).

For the 2013 ICDAR Robust Reading task, USTB_TexStar developed by Yin et al. [95] obtained 1st place [94, 93]. This system proposed an MSER pruning algorithm to improve precision, uses single-link clustering to group candidate regions instead of empirically selecting a threshold, and uses character classification to filter non-text candidates at last.

In second place, Neumann et al. used an MSER-based algorithm [77]. In [78], they compared filtering methods considering isolated CCs, CC pairs, and CC triples, and find the additional context available in features extracted from CC triples provide the best performance. In follow-on work [96] they propose cascaded filtering of MSER regions. Simple features are computed beforehand to eliminate easy backgrounds, and then more complex features are used. with an AdaBoost classifier. In [97], multiple candidates are kept after recognition and sequence selection to improve recall, and a Gaussian scale-space pyramid to increase accuracy.

2.7 Summary

In this chapter, we discussed the background of text detection systems. Detection is a part of image understanding pipeline, containing detection, localization, segmentation and recognition. Detection techniques have a great impact on industrial and academic uses. Recent researches focus on text detection in complex scenes, like camera based natural images, where text sizes, colors, positions, and orientations are with much larger variations. Complex backgrounds, noises and deformations make this kind of problem challenging and interesting.

Ground truth generation methods are discussed in both pixel level and bounding box level. The generation methods can be categorized into two major groups: synthetic and manual labeling. Most previous system used either bounding box or pixel level ground truth. In our experiment, both ground truth and pixel level features are used together in different stages to form a system with enhanced accuracy. Details are discussed in Chapter 4, 5 and 6.

Evaluation methods, in pixel level and bounding box level, are discussed and metrics are introduced. Precision, recall and F-measure are commonly used to evaluate detection systems. Correlation based, information based and shape based evaluation metrics describe the system performance from different aspects. In our thesis, we adapt ICDAR 2015 evaluation method [1], which is a bounding box level metric. It considers the one-to-multi and multi-to-one box matching and used precision, recall and F-measure. Although, the metric adapted by ICDAR is not explicitly correlation based. It shares many common properties. They are both subject to the overlapping area between detection and ground truth. They are also normalized based on the total area. Unlike shape based metrics, ICDAR metric does not explicitly penalize regions with small uniformity. Unlike entropy based metrics, they are not seeing the detection as signal processing or model the detection error as noise.

Features that used by text detection systems are surveyed. Conventional features include color, edge, texture, stroke width and regions. However, modern systems can create adaptive features using self-learning technique. Related algorithms are also introduced. In our system, convolutional K-means is used. An alternative feature learning based on support vectors are also introduced to compensate the representative features with discriminative features (as in Chapter 3).

Heuristic rules and machine learning methods are used to utilize those features and combine different

features for accurate detection. Machine learning algorithms use less strong prior knowledges or human interventions, therefore, they are more suitable for complex datasets. Different sets of machine learning algorithms are surveyed, and the proposed system utilize a combination of them. A structure similar to single layer of CNN is combined with AdaBoost for patch based detection. Contextual information is used and to enhance the coarse detection accuracy (see Chapter 4). Random Forest is used to learn relationship classifiers to segment characters into words. By utilizing different machine learning algorithms intelligently and properly, we built a system with state-of-art accuracy.

ICDAR competition is introduced, their dataset, evaluation metrics, and participant top players are discussed. Though for conventional document image, the state-of-art systems achieve near perfect performance, the text detection in complex scenes performance still has abundant room for improvements. The following chapters of our thesis discuss our approaches and difficulties designing such a detection system, and hope can contribute to the efforts of solving this interesting problem.

Chapter 3

Feature Learning

In our thesis, a state-of-art text detection system is built using features learned from convolutional K-means. From the experiment, combining with appropriate searching algorithms, feature learned from convolutional K-means is sufficient for building a classifier that separates complex targets and backgrounds effectively. Through this chapter, research question as stated in Chapter 1 is to be answered:

Can features obtained via convolutional K-means be made more discriminating?

In this chapter, the procedure we have done to learn adaptive features for text detection system is introduced. Besides convolutional K-means, we utilize SVM to learn features more close to classification boundary. These features are called discriminative, comparing to representative features that were learned by convolutional K-means.

As discussed in Chapter 2, features used for text detection are categorized into two major groups: designed features through careful engineering and strong prior knowledge and learned features that are computed from training data.

Although the earlier group is easier to produce and achieved success against document images, it has difficulties in text detection problem for natural scenes where target variations are large. On the other hand, feature learning utilizes simple algorithms to form complicated features from data. Those features do not require expert domain knowledge about the data and generalize better than hand-crafted.

In section 3.1, we introduce the ICDAR 2015 dataset, which is the main dataset for training and testing our text detection system in Chapter 3, 4 and 6. In section 3.2, we discuss feature learning using simple and efficient convolutional K-means. This method is so simple and powerful, an accurate text detection

system based on these features can be built without using a lexicon model. In section 3.3, the effort has been made to compensate the disadvantages of the convolutional K-means based algorithms with discriminative features. Support Vectors are used to compose this new feature group.

3.1 Data

We tested our system on ICDAR 2015 focused scene image dataset, containing 258 training images and 251 testing images. The ICDAR 2015 images are camera images randomly taken for different scenes, including book covers, street views, indoor signs and objects, and commercial products, etc. The texts within those images are mostly scene texts (texts within the original scenes), but not graphic texts (texts embedded after acquisition).

The text colors, positions, sizes, and fonts are with great variation, and very hard to be detected by conventional OCR systems. Although, the images contain handwritten texts, a large portion of the targets are typed fonts. Graphic Calligraphy is contained in some of the images, including commercial brands (coca-cola, sony, etc.), computer screen digits, book titles. These unorthodox text fonts make the detection problem even harder. Poor lighting conditions and reflections make some of the targets harder to recognize, even by human eyes.

Backgrounds in those images are also very complex, as shown in Figure 3.1. Fences, bricks, and leaves in the natural environment are most likely to be miss-classified as texts, as they possess similar stroke width, spatial pattern, and uniform colors. As pictures are taken in totally different environments and lighting conditions, backgrounds are very hard to model.

The image sizes in the dataset vary greatly as well. For training data, the smallest height is only 102 pixels, and largest height is 2592 pixels. The smallest width of the image is 422 pixels while the largest is 3888 pixels. The smallest area of the image is 43,044 square pixels while the largest is 10,077,696.

For test data, the smallest height is 200 pixels, while the largest is 2592. The smallest width is 350 pixels while the largest is 3888. The smallest area of the image is 70,000 square pixels while the largest is 10,077,696.

As the image sizes change significantly, the text sizes within those images also change. In this dataset, some small images could contain larger text than large images. This characteristic prohibits scale normaliza-

Figure 3.1: Natural scene image samples from ICDAR 2015 robust reading competition. ICDAR 2015 competition uses the same group of images for text detection.

tion based on image sizes. Therefore, the text size inside those images are unknown and need to be searched exhaustively. This slowness in speed is a common disadvantage of patch-scanning-based methods.

Figure 3.2 shows the histogram of the characters contained in training and testing datasets. The histogram shows most of the symbols are numerical and English characters in the dataset, accompanied by a small number of punctuation symbols and math symbols. The difference between training and testing symbol probability distribution could cause problems for our system. For example, symbol '.', ',' and '|' only appear in training sample, and symbol '[' and ']' only appear in the testing sample. One way to illustrate this difference is to normalize the distribution probability function and compute the difference between the two, which is shown in Figure 3.3. The figure shows that difference is small, and should have a subtle effect on the final performance. The most significant difference is occurred for symbol 'e' and 't', where testing set contains more samples, as well as for 'L' and 'a', where training set contains more samples.

3.2 Convolutional K-means Algorithm

Feature learning algorithms were developed using restricted Boltzmann machine (rBM) [101] or auto-encoder [102] etc. Coates et al. [60] proposed convolutional K-means feature learning algorithm, utilized simple k-means clustering to learn presentation banks. Convolutional K-means examines the angle distance (dot product) between training patch samples and generates cluster center vectors through iterated learning

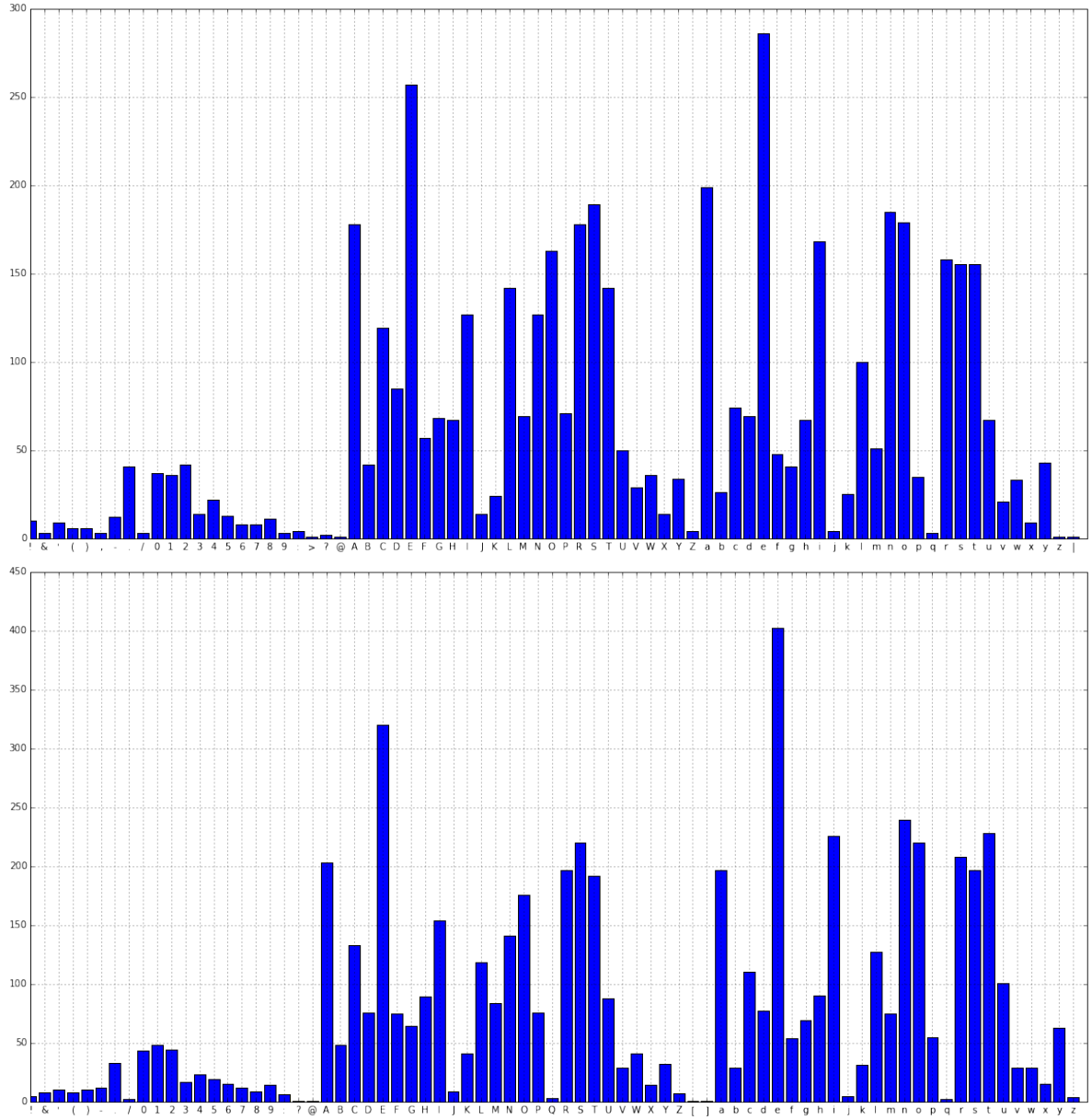


Figure 3.2: The histogram of character labels in ICDAR 2015 dataset. The training and testing dataset contain different symbols. For example, '.', '>' and '|' are contained by training images while do not appear in testing data. '[' and ']' are contained in the testing sample but not in the training sample. Both datasets contain numerical and English characters. The total number of appearance of each character is not equal. Although, we are using a non-statistical model for text detection, and our system does not rely on a lexicon model, the difference between training and testing character distribution will have an effect on the final accuracy. However, the effect is small, as the distribution difference is trivial for popular symbols.

process.

They [60] claimed that it is more desirable to use fundamentally adaptive methods (convolutional K-

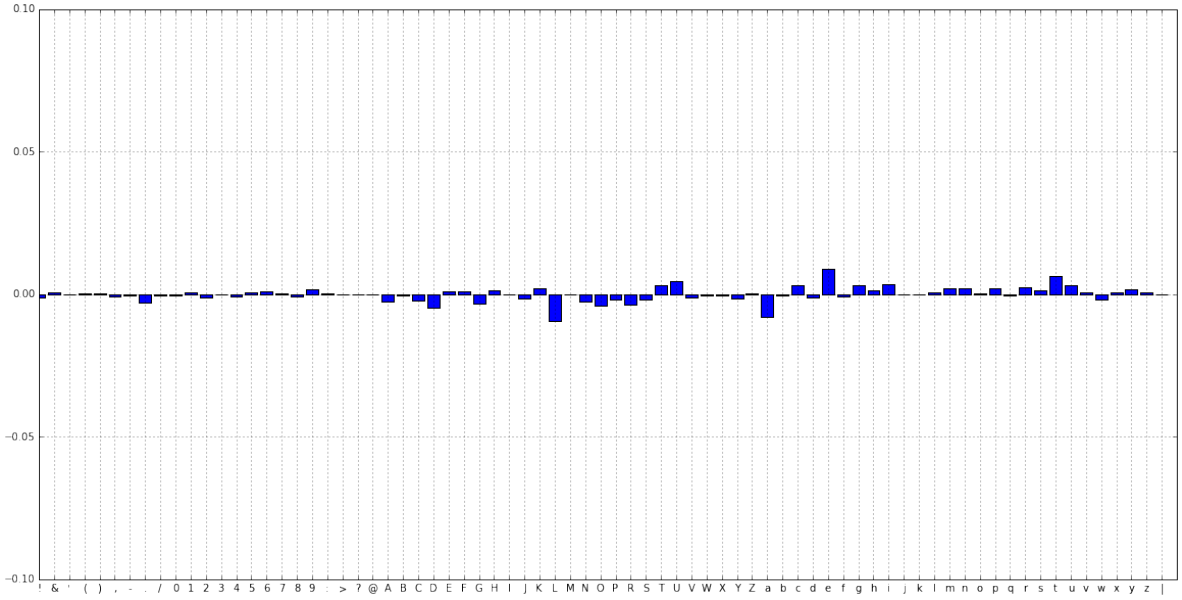


Figure 3.3: The Difference of histogram of character labels in ICDAR 2015 dataset after normalization. The histogram for training and testing is normalized, therefore accumulation of the histogram adds up to 1. Then the difference between two histograms is computed. The difference of distribution between training and testing is small. The positive value indicates increasing of possibility in testing data, while negative value indicates decreasing. The largest two increments are at character 'e' and 't'. The largest two decrements are at character 'L' and 'a'.

means) to train and test image data than manually designing new features. Manually designing for different datasets is difficult and slow to progress.

The cluster centers represent typical patterns, including horizontal/vertical bars, corners, tilted bars with different angles, zebra textures etc. These patterns are learned automatically from data, without elaborate modeling. Some examples Coates learned from natural scene images are shown in Figure 3.4. The learned features are mostly abstractions of line and texture patterns.

The only hyperparameter that needs tuning is the number of clusters. These learned features are very similar to those acquired by an autoencoder or rBM according to Coates [101].

Bengio etc. [84] reasoned that why Coates could learn great performance using relatively simple K-means based methods is partially due to relatively low dimensions of those targets. So one could use learning methods with fewer layers to achieve acceptable performance. Coates learned some basic image patch dictionaries, mostly lines, corners, and curves. However, they are sufficient for text detection, because texts are mostly combinations of these basic strokes.

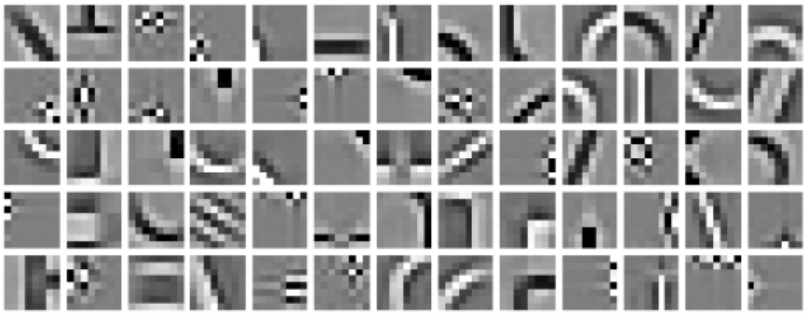


Figure 3.4: Some example of feature dictionary patches learned using convolutional K-means from natural scene images generated by Coates[60].

On the other hand, in more complicated cases, like face detections, sophisticated feature representations are used. In a typical convolutional feature learning neural network, multiple hierarchies of features are learned within different layers of higher and higher abstraction. The first layer could represent basic patterns; the second layer combines basic patterns and represents some common cues with higher abstraction, like partial faces, eyes, noses and cheeks with different viewing angles [103].

3.2.1 Methodology

Our system proceeds in several stages:

- Apply an unsupervised or supervised learning algorithm to a set of image patches harvested from the training data to learn a bank of codebooks;
- Evaluate the features convolutionally over the training images;
- Train a classifier for text detection based on convolutional results as inputs.

In this chapter, we discuss the method for the first step: learning features to form a bank of codebooks. The second and third steps are introduced in next chapter.

Like many feature learning schemes, the convolutional K-means based feature learning is composed of following steps:

- Harvest a set of small patches (8×8), x_i from training images, $x_i \in \mathbb{R}^{64}$.
- Normalize the training vectors to have standard length 1;

- Run unsupervised learning algorithm to build a mapping from input patches to a feature vector bank.

First, given a set of training images, we extract 8-by-8 patches to yield vectors of pixels x_i. Then, each vector is normalized to have length equals to 1, $x_1^2 + x_2^2 + ... + x_{64}^2 = 1$. Given this normalized bank of input vectors, we are now ready to learn a set of features as cluster centers from those patches.

The Convolutional K-means algorithm is similar to standard K-means, updating sample cluster labels and shifting cluster centers iteratively to maximize inter-cluster distance and minimize intra-cluster distance. However, instead of using Euclidean distance, convolutional K-means uses inner product distance as a similarity measurement. For sample dataset containing m samples with n features, we make the matrix size as $n \times m$, where each column stands for a sample vector and each row corresponds to a feature scalar, $X \in \mathbb{R}^{n \times m}$.

For initialization, we randomly pick k samples from the sample matrix X, normalize each vector, so cluster center matrix $D \in \mathbb{R}^{n \times k}$. k is the number of clusters. The goal here is to minimize the equation:

$$\sum_i ||Ds_i - x_i||^2 \tag{3.1}$$

as in [60]. Each column of D is the normalized basis vector (i. e. cluster center). s_i is the hot encodings vectors with only one non-zero element, denoting which column of D that current sample x_i is belonging to. And its magnitude is the dot product distance between the current sample and cluster center. x_i is the corresponding training sample. To search for matrix D with minimum overall distances from samples to cluster centers, we alternatively minimize D and s_i.

Having D fixed, we solve for s_i by letting $s_i[k] = D^{(k)\top} x_i$ for

$$k = \arg\max_j D^{(j)\top} x_i \tag{3.2}$$

where $s_i[k]$ means kth elements of s_i, $D^{(j)\top}$ means the transpose of jth column of D. And other elements in s_i except $s_i[k]$ are set to zero.

Having s_i fixed, we solve the minimum of equation (3.1) for D in closed form. Notice that each column of D could be solved independently. So for each column, we solve for:

$$\sum_{p} ||ya_p - x_p||^2 \tag{3.3}$$

where y is a column of D, which is a center vector of a cluster. p is the indices of samples that are assigned to column/cluster y. a_p is the single non-zero element in s_p. x_p are samples that are classified to y. Compute the first derivative and set it to zero, we have:

$$\sum_{p} (2a_p^2 y - 2a_p x_p) = 0 \tag{3.4}$$

This way, we can calculate center vector y in terms of current samples x_p and sample weights a_p.

$$y \sum_{p} 2a_p^2 - \sum_{p} 2a_p x_p = 0 \tag{3.5}$$

$$y = \frac{\sum_p a_p x_p}{\sum_p a_p^2} \tag{3.6}$$

We now could compute each column y in matrix D using the classified samples x_p and their corresponding hot encodings s_p. The second derivative of (3.3) is given as,

$$\frac{d^2 \sum_p ||ya_p - x_p||^2}{dy^2} \tag{3.7}$$

$$= \frac{d \sum_p (2a_p^2 y - 2a_p x_p)}{dy} \tag{3.8}$$

$$= 2a_p^2 \tag{3.9}$$

Notice this value is always greater than or equal to zero, so the extremum we computed in Equation (3.6) is the minimum value. We update D and s alternatively, and after a certain number of iterations or the overall distance is less than a threshold, we stop the learning process. Columns of D are then used as a dictionary of quantized features. These quantized features, unlike features designed with prior knowledge, are directly learned from the data. In the experiment, 1000 features are learned for detectors.

3.2.2 Experiments

First, 32 by 32 patches are extracted from training images. The patches can be categorized into foreground and background as labels of those patches are known or derived from bounding box labels. According to Coates' labeling method, the foreground patches should have at least 80% overlapping area with known character bounding boxes. Meanwhile, at least one of the width or height of the patch should be within 30% of the width or height of the character bounding box.

However, different from Coates. We define foreground patches in a more meaningful way. We found that patch labeling is dependent to bounding box ground truth, the patch size and step size of patch extraction.

Let's assume a case where patch sizes are 32 by 32 and step sizes are 16, the neighboring scales have a ratio factor of 0.9. When the patch scans across the foreground character bounding box, the minimum overlap ratio in one direction is 0.75. For a 2-D image, the minimum area overlap ratio is $0.75^2 = 0.56$.

Considering this worst case where a patch is partially overlapped with a foreground character, the definition for foreground patches is: if a patch is less than 10% different from the width or height of a character bounding box, and the overlapping area is greater than $0.75^2 = 0.56$, then it is considered a foreground patch.

In order to extract foreground samples, one should raster scan images with multiple scales (30 different scales in our experiment) and extract all qualified foreground patches, meanwhile, get corresponding background samples as well. Because foreground samples can be in different sizes, depending on the sizes of text characters, they need to be resized to 32 by 32 afterward. The resize is done by bilinear interpolation. As background samples are much more frequent than foreground samples, we subsample the background to equalize the sample numbers using random sampling. In practice, the equalization is done within each image, and within each scale. Therefore, fewer numbers of background samples need to be kept, saving an enormous amount of memory and disk space.

After extracting 32 by 32 image patches, we then extract 8 by 8 patches from larger patches, as shown in Figure 3.5. This is the second stage of patch extraction. For each 32 by 32 image patch, four 8 by 8 patches are extracted, the location of the smaller patch within the larger patch is randomized. This randomness is very important, which ensures good performance where convolution is performed. If the extraction is not random enough, convolution results will degrade as excitations at different locations are biased. Totally

50,000 large patches are used for feature learning, containing 25,000 foregrounds and 25,000 backgrounds. Therefore, 200,000 small patches (8 by 8) are used for feature learning over convolutional K-means. That is a combination of 100,000 foregrounds and 100,000 background samples.

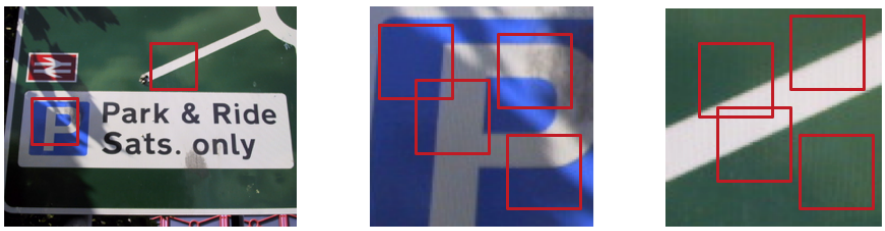


Figure 3.5: Example of 8 by 8 patches extraction. We first generate 32 by 32 patches from images. The region overlapping area with the ground truth is computed to label the region as foreground and background. Then foreground/background samples are equalized for each image. Then from extracted 32 by 32 patches, 8 by 8 patches are extracted randomly and used to learn features. The original image is shown at left, two examples of foreground and background 32 by 32 patches are marked by red rectangles. Then two patches are shown on the left and randomly selected 8 by 8 patches are shown inside each 32 by 32 patch. This process is same as in Coates work [60].

To initialize, the cluster centers are uniformly randomly selected from 8 by 8 patches. Then convolution K-means algorithm is performed over all training samples iteratively.

To evaluate the discrimination power of different numbers of features, one can apply those features to a simple AdaBoost classifier and compare the classification accuracy. Using this method, we train a classifier with 1000 iterations and compute AUC and F-measure in patch level, as described below.

To compare our system with Coates' work [60], we compute the area-under-curve of precision/recall (AUC) in patch level. By varying the threshold and using the ICDAR bounding boxes as labels, one can sweep out a precision-recall curve for the detector. The area under this curve (AUC) is used as the performance measure by Coates. The precision and recall are computed in patch level using testing patches generated from natural scene images. For 50,000 patches generated from the image, 80% are used for training and 20% are used for testing. Therefore, 10,000 patches containing equal numbers of foreground and background samples are used for generating AUC curve.

Meanwhile, we also evaluate the performance in terms of F-measure in patch level. It is performed using same patches as in AUC computation. Different from AUC measurement, which evaluates an average

performance across the entire threshold range, f-measure is computed at one threshold which maximizes the performance.

Finally, one can test the performance of the entire system. Additional components in the text detection pipeline are added, including patch based detection (coarse and fine), region growing, word segmentation classification etc, which are introduced in later Chapters. Precision, recall, and F-measure for the final output is computed in bounding box level.

The patch level measurement, including AUC and F-measure, are computed on a cleaner dataset, where foreground and background patches are categorized, randomized, and equalized. On the other hand, overall performance is more complicated where the effect of the initial detection is altered by later processing. Both kinds of evaluations are interesting. Former one shows a direct relationship between discrimination power and features, later one shows its effect when applied to the entire system.

3.2.3 Results and Discussion

The convolutional K-means is trained on 200,000 samples of 8 by 8 patches. Each patch is serialized into 64 element long vectors. And 1000 iterations of convolutional K-means is performed. The total movement of centers at each iteration is recorded and shown in Figure 3.6. The center movement distance converges to a small value near zero indicates that the algorithm has met an equilibrium. Usually, some centers will be fixed at certain locations, other centers would jump between different locations with small distances. We stop the iteration at 1000 as the overall movement is negligibly small.

An example of feature codebooks learned for different stages of detectors (coarse and fine) are shown in Figure 5.4. The coarse and fine detection will be described in detail in Chapter 4. These features are patterns, including bars, stripes, corners and blobs, which are not designed by hand, but learned directly from training data, representing basic shape elements of texts and backgrounds.

The number of features learned is a hyperparameter that affects the performance and speed of the system. More features could help to get higher accuracy for subsequent processing and the detection. On the other hand, more features mean a longer learning time of convolutional K-means and a longer time for classification. The time complexity of classification with respect to the number of features n is linear $O(n)$. We tested the AUC, F-measure in patch level and performance of the overall system on bounding box level using different numbers of features to prove our hypothesis. 500, 1000, 2000 features are used.

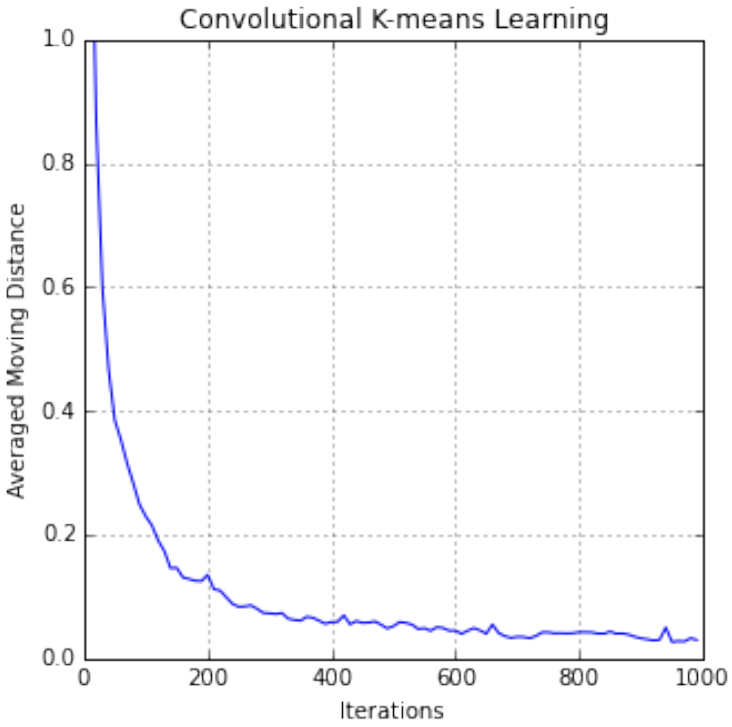


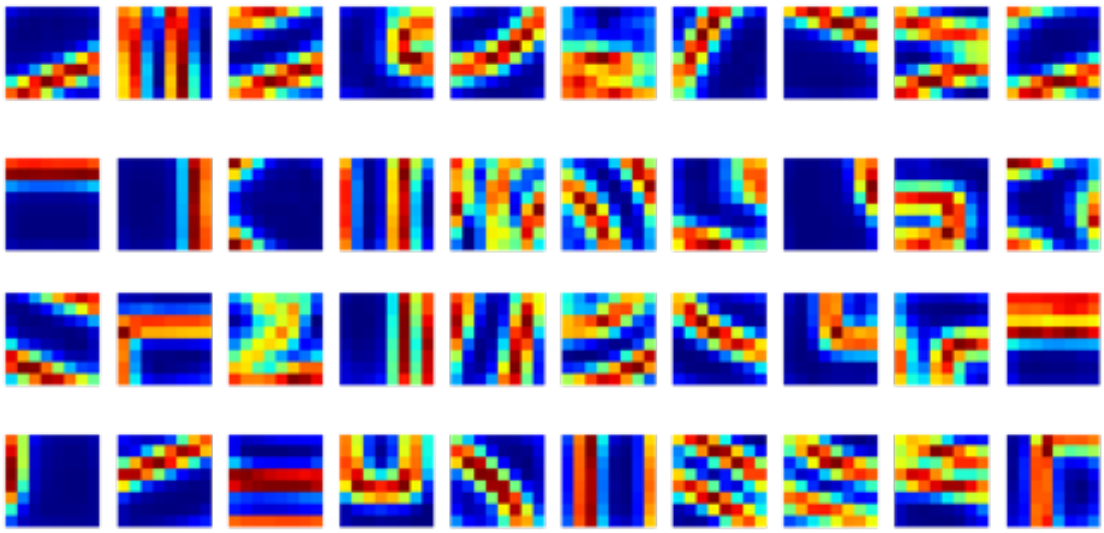
Figure 3.6: The total movement of centers at each iteration of convolutional K-means. The movement gets smaller through iterations and converges at about 1000. The center stops shifting indicates that the algorithm has achieved equilibrium: the inner cluster distance is minimized and inter cluster distance is maximized under the definition of Convolutional K-means.

The experiment results is shown in Table 3.1. The fewer features we use, the lower performance it is. However, when using 1000 and 2000 features, the performance incrementation is trivial. Considering the time complexity is $O(n)$, we use only 1000 features for our systems for a faster processing time.

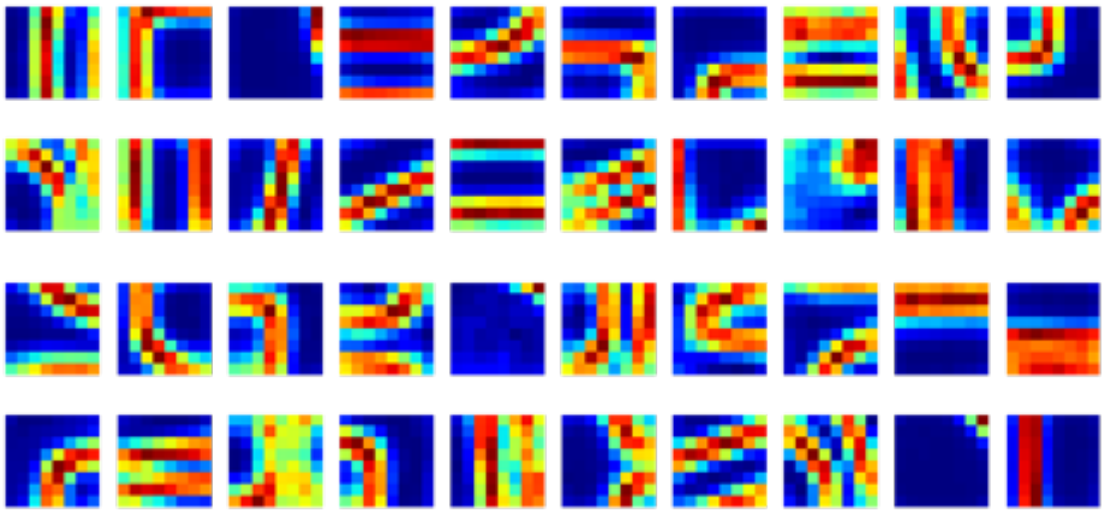
To compare our performance with Coates, we show the AUC measurement in Table 3.1, where we achieved similar performance using 1000 features. Our AUC is 61.49% comparing to Coates' 62%. By adding more clusters, one can even improve the performance to 61.86% at 2000 features. This is not a surprise as we implement very similar architecture to Coates for feature learning.

The F-measure in patch level is shown in Table 3.1. The F-measure increases with No. of features. Also, notice that this evaluation measurement is very high, as it is applied to equalized foreground and background samples. One need to achieve very accurate classification on this testing dataset in order to get acceptable performance for the overall system.

The performance of the entire system is also shown with different numbers of features, in Table 3.1. The



(a)



(b)

Figure 3.7: Features learned using convolutional K-means algorithm. (a) Top 40 features with highest weights in coarse AdaBoost detector; (b) Top 40 features with highest weights in fine AdaBoost detector. The feature intensity is plotted in 'jet' colormap. The learned features are convolutional K-means centers, containing bars in horizontal, vertical, and slanted. They also contain corners, curves and zebra patterns.

two kinds of evaluations are differed mainly based on two reasons.

First, in practice, background samples are much more frequent than foreground samples. Therefore, missing is more expensive than false alarms. To address this problem, coarse and fine detection are introduced to establish a filtering pipeline to enhance precision in each stage. Furthermore, verification classifiers

Table 3.1: Performance comparison using different number of features. The performance is evaluated using overall system, where only number of features are changed for the coarse detector. The evaluation is performed using word-level bounding box ground truth and ICDAR 2015 metrics. The performance is monotonically increasing with the number of features, which means more features will benefit the system accuracy. However, the difference between 2000 and 1000 features are not large.

No. of Features	AUC	Patch (%)			Overall (%)		
		Precision	Recall	F-measure	Precision	Recall	F-measure
500	55.38	90.31	86.04	88.12	90.77	74.81	82.02
1000	61.49	98.35	95.88	97.10	93.38	81.02	86.76
2000	61.86	98.69	96.72	97.70	93.39	81.63	87.07
Coates	62	-	-	-	-	-	-

are used to prune remaining false positives in CC level. These processes are discussed in more details in later chapters.

Second, the patches are extracted with different scales. Therefore, converting the patch level detection back to a meaningful hotmap (pixel level) or bounding box coordinates (Bounding box level) requires merging different scales of detections. The merging process could amplify the negative impact of the classification errors if patches are in large scales and minify the impact if patches are small. Meanwhile, merging can reduce some negative effects of missing if patches in the same location are found in neighboring scales.

Therefore, two kinds of evaluations are both meaningful. The first one indicates the direct relationship between classification correctness with features while the second one shows a more comprehensive effect of features when applying to the wild scenes.

Convolutional K-means generate representative features. However, the label information is not utilized yet. To detect text from complex scene images, one would like to address the difference between text patches and background patches. Therefore, discriminative features are important for training a good classifier. As training set already provided the label information, a simple Support Vector Machine is can be trained therefore support vectors can be used as another set of feature codebooks. Our experiments show that cluster center vectors and support vectors, being representative and discriminative features respectively, can get stronger classification results than pure convolutional K-means learning.

3.3 Combining Feature Learning with Support Vectors

For feature learning based methods, features and the original training data lives in the same feature space. In our experiment, they are all 8 by 8 image patches, i. e. 64-dimensional. Most feature learning methods, including auto-encoder, rBM, and convolutional K-means, aim to generalize the representation using certain average computations, according to the *Sparsity* assumption of Bengio [84]. *Sparsity* requires features are independent of others, therefore maximizing the inter-cluster angle distance (minimizing dot product) does that for us. It is called a representative feature because it is intended to be generalized representations for a partial group (cluster) of training data.

Another category of features is called *discriminative*. Instead of staying at the center of clusters, discriminative features are close to the boundaries of different classes. The advantage of discriminative features is that they can utilize the information of labels, if available, from the training data. Meanwhile, representative features could only catch the inertial structural characteristics of data, but label information is discarded.

As discuss above, the convolutional K-means learn representative features. Using the label information from the training data, one can learn discriminative features with a supervised learning method.

3.3.1 Methodology

A very intuitive method to learn discriminative feature is to utilize the support vectors in the SVM algorithm. The SVM finds a decision boundary that best separate two group of data by maximizing the margin between data points and boundaries. The closest points to the decision boundary are called support vectors, meaning they are supporting the maximum-margin hyperplane. To separate data points that are not linearly separable, kernel tricks could be used to transform data from one feature space into another.

The support vector is utilized in our system based on several reasons:

- It is effective in high dimensional feature space;
- It uses a subset of training points in decision function, which is a vital property allowing to use it as a feature selector;
- It selects vectors that best separate classes (maximizing margin), which indicates optimal discriminative power of those features;

- It provides weights for each support vectors, which are proportional to the margin sizes of the decision boundary hyperplane.

These properties of SVM algorithm make it a very desirable approach for training discriminative features from labeled data.

SV Kernel Selection For feature learning, one does not need to classify the samples, but to learn features abstracted from the samples. Therefore, this algorithm is not applied directly for classification.

In order to extract optimal discriminative features, different kernels are tested. Because those features are with high dimensions, a visual observation is not feasible to be used for helping select kernels. One way to compare the performance of different kernel methods is to evaluate the classification results on a testing dataset.

This comparison contains the linear kernel, which performs no feature space transformation; polynomial kernel, which measures the distances of samples using dot products; radial basis function (RBF) kernel, which measures the distances of samples using Gaussian functions.

The SVM classifier is trained and tested to classify testing sample data to find out the most discriminative kernel. There are several parameters need to be grid-searched:

- C, penalty parameter for error term. The C parameter trades off misclassification of training examples against the simplicity of the decision surface. A low C makes the decision surface smooth while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors (more prone to overfitting).

- γ, kernel coefficient. The γ parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The γ parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

- Degree of the polynomial kernel function. The degree of the dot product of x_i and x_j. The degrees are usually chosen from 1, 2, 3, ...

- Zero Coefficient. If this coefficient is 0, then the polynomial kernel is homogeneous. Otherwise, it is inhomogeneous, and the zero coefficient is a constant value added to the dot product.

3.3.2 Experiments

The experiments are done using the same dataset as described in section 3.2. There are totally 200,000 samples of 8×8 image patches with labels for SVM classification training and testing, containing 100,000 foregrounds and 100,000 backgrounds. 80% are used for training and 20% are used for testing.

For grid search, we choose different C values: 1, 10, 100, 1000; gamma values: 0.001, 0.01, 0.1; degree of polynomials: 1, 2, 3; Zero coefficient: 0. 1000 support vectors are generated. The results of the grid search are shown in Table 3.2, where the Linear kernel gets best result.

Then, different numbers of support vectors are learned using linear kernel. The numbers are chosen to be 500, 1000, 2000 to match the cluster center method of feature learning. To pick a specific number, one can randomly sample from all support vectors. The stochastic distribution of selection probability is based on the weights of each support vectors. The weights of all support vectors are normalized and then random picking (one at a time) is performed 500, 1000, 2000 times respectively from all support vectors subjected to this probability distribution.

We then perform convolutional K-means with 1000 iterations using these support vectors as seeds to produce discriminative features. The convolutional K-means is critical because it averages the support vectors to remove singleton examples and make the learned feature more generalized.

We then compare the performances among detection classifiers using convolutional K-means, using support vectors and a combination of the two. In our hypothesis, the discriminative features (support vectors learned) could not serve well alone as a base for text detection problem, as it lacks the generalization of representation for all samples. However, it should be a good supplement for cluster centers. It addresses the information that was missed by using convolutional K-means alone. And it utilizes the label information that was ignored by other types of feature learning methods.

3.3.3 Results and Discussion

As in Table 3.2, the linear kernel produces the best result when comparing to polynomial and RBF kernels. It indicates that although the images are very complex, features are with high dimensions, they are mostly linearly separable. The best results are at C equals 10.

The performance is shown in Table 3.3 supports the correctness of the hypothesis: The combination of

Table 3.2: Grid search for support vectors. The support vector machine is tested using linear, polynomial and rbf kernels. For linear kernel, C value 1, 10, 100 and 1000 are tested; For rbf kernel, besides C value, gamma value 0.001, 0.01 and 0.1 are tested; For polynomial kernel, besides C and gamma, degree value 1, 2 and 3 are tested. The error rates on testing dataset are shown.

Kernel	γ	C = 1	C = 10	C = 100	C = 1000
Linear	-	21.54	**20.36**	25.86	35.01
RBF	0.001	22.35	20.85	26.71	38.92
	0.01	21.93	20.86	26.10	35.44
	0.1	21.47	21.33	26.84	35.28
Poly 1	0.001	25.45	23.13	22.27	29.71
	0.01	26.72	23.58	23.39	30.83
	0.1	38.93	39.20	40.41	39.21
Poly 2	0.001	24.39	22.75	26.75	36.20
	0.01	25.31	22.84	28.10	37.17
	0.1	39.28	42.45	45.77	43.02
Poly 3	0.001	25.85	24.09	23.73	32.66
	0.01	27.28	25.47	28.91	34.48
	0.1	29.71	26.95	31.99	36.20

representative feature (convolutional K-means) and discriminative feature (support vectors) serves better as the basis for our text detection classifier. On the other hand, if only the discriminative features are used, the performance degrades significantly. Different numbers of features are also tested. More features can enhance performance. On the other hand, increasing in feature number means slower computation.

Although the combination of discriminative and representative features improves performance, the increase is small. The representative features certainly play a more important role in the classification, as using them alone could achieve comparable accuracy. On the other hand, discriminative features could not achieve similar results, the decrease in accuracy using only those features is large.

In practice, the algorithm that combines both feature extraction methods is valuable. Because the performance improvement can be achieved not by using more features, but by changing the feature compositions, the improvement in accuracy does not require sacrifice in speed.

Table 3.3: Performance comparison between feature learning methods, where representative feature and discriminative features are learned. Convolutional K-means are used to learn representative features and Support Vectors are used to learn discriminative features. The Convolutional K-means based features have much stronger classification power than SV based features according to the experiment. By combining the two kinds of feature, one could make the overall performance even higher. Precision, Recall and F-measure are listed to make full comparison.

Support Vectors			
No. of Features	Recall	Precision	F-measure
500	35.02	70.32	46.76
1000	37.29	74.48	49.70
2000	38.78	75.61	51.27
Random Selection			
No. of Features	Recall	Precision	F-measure
500	74.81	90.77	82.02
1000	81.02	93.38	86.76
2000	81.63	93.39	87.07
Combined			
No. of Features	Recall	Precision	F-measure
500	76.12	91.43	83.08
1000	81.33	93.50	86.99
2000	81.64	93.40	87.13

3.4 Summary

To our research question, *can features obtained via convolutional K-means be made more discriminating?* The answer is yes. Although the convolutional K-means can be made more useful by simply increasing the total number of features learned, the improvement is small when changing from 1000 to 2000 features. However, as convolutional K-means only produce representative features that address the inertial structure of the samples, one can add discriminative features to enhance the performance. The label information is utilized through the supervised learning process (SVM in our case). Combining different kinds of features gets better performance without sacrificing execution speed.

The combination of representative and discriminative features provide slightly improved performance for our text detection system. This combination of features is an approach to enhance performance while preserving computation speed. The performance is tested using the overall system, replacing only feature learning part for different cases. The evaluations are done in word bounding box level with ICDAR 2015 images.

Although only some specific implementations on text detection are shown in this chapter, the answers above are generic and can be adapted to a broader scope of pattern recognition problems.

First, the features learned from data shows great effectiveness against complex scenes. This procedure avoids strong prior knowledge and domain specified engineering labor. Although the mathematical model is simple (Convolutional K-means or SVM), the learned features are powerful. For different types of images, or for other types of data, e. g. economical, biological and chemical, where features are hard to be engineered or designed, they can be learned directly from data itself.

Second, besides representative features, label information are utilized to learn features to enhance overall performance. An intuitive way to create features close to class boundaries is to utilize support vectors. This hypothesis is supported by our experiment, where a combination of representative and discriminative features can get improved accuracy. Notice that this method makes no assumption about data types, therefore, it can also be applied to other types of datasets and problems.

The feature learning process generates feature codebooks from training data automatically. The next question is how to use them to form an accurate classifier for detection. In next chapter, we introduce methods that utilize those features through a simple convolution algorithm and use the convolution results as inputs for an AdaBoost classification problem.

Chapter 4

Word Detector

There are two approaches for text detection, which are patch scanning based word detector and CC based character detector [14]. The first method searches the image in a raster scan manner; The second method extracts CCs as candidates for further processing.

In our system, a combination of both raster scan and CC based methods is used. It increases the overall performance by compensating disadvantages of each individual method. An illustration of our text detection pipeline is shown in Figure 4.1. The raster scan for word detection is utilized first, finding candidate image patches with coarse and fine grids; The CC based method for character detection is used then, where CCs are formed and filtered to enhance precision.

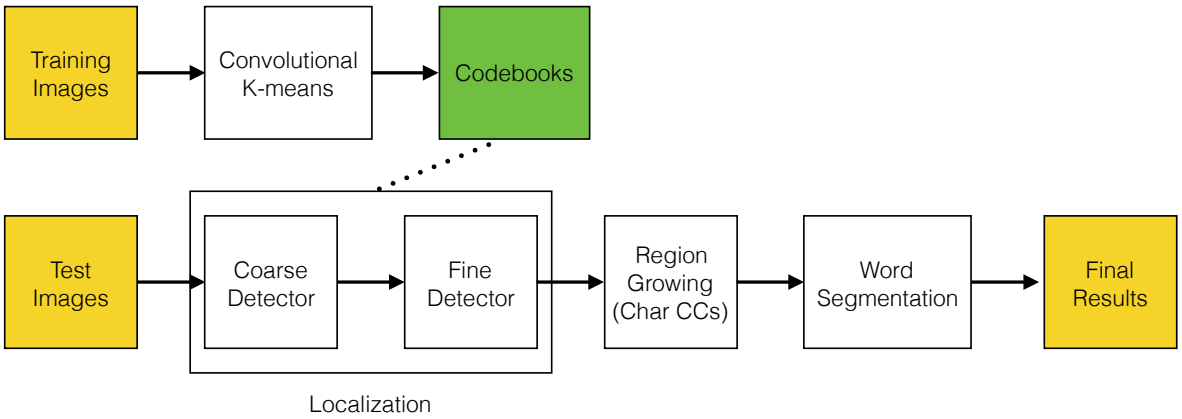


Figure 4.1: System Architecture. The main stages localize text pixels, generate and verify connected components as characters, and finally segment words

The patch based classification produces hotmaps as text location predictions. This is not pixel level

accurate and can not be used directly for ICDAR 2015 evaluation. To utilize these predictions, one need to convert patch level prediction into pixel and word bounding box levels. The following chapters discuss the methods of utilizing patch based detector and combine its prediction with CC leveled detections.

Patch Based Detector. The advantage of the *patch-based model* is its capability of extracting texts with different shapes and colors. The disadvantages are its slowness and computational expensiveness. Because texts are in different sizes, aspect ratios and angles, detection usually requires scanning an image multiple times. Furthermore, a patch-based method is incapable of getting pixel level predictions.

The patch based detector is called *Text-Conv*, with a two-step scanning including coarse and fine, to achieve the balance between speed and accuracy. The coarse detector is used to estimate text location, which also scans the image in different scales. Then fine detector is applied to pinpoint the center location of the text patches.

CC based Detector. The advantage of *CC based model* is its speed because only a limited number of candidates are considered. CCs can be seen as super-pixels pre-segmented from the image directly. Once CCs are extracted, a classifier can filter background from foreground samples. But the accuracy is radically dependent on the CC extraction. An inadequate extraction algorithm degrades the overall performance.

The CC based character detector is called *Word-Graph* (Chapter 6), the center pixels found by fine detection are used as seeds in a region growing algorithm to form CC candidates. The region growing algorithm is flood-filling fashioned, which adding new pixels into the foreground by checking their gradient values and color differences repeatedly. The formed CCs are fed into verification classifiers for the further pruning.

In this chapter, the patch based word detector Text-Conv is discussed, regarding the research questions:

- Is visual feature alone sufficient for high-accuracy text detection in natural scenes?
- How can features obtained via convolutional k-means be made more discriminating?
- How can detection accuracy be improved by combining convolutional k-means detection with subsequent processing?

The Text-Conv is evaluated and shows high accuracy on patch level. Different methods are implemented with learned features to achieve this high accuracy, including two-stage classification with coarse and fine step sizes, contextual features and a combination of different rotational angles and aspect ratios.

4.1 Methodology

The codebooks learned from convolutional K-means (Chapter 3) are utilized and combined with a single convolutional feature generation layer. And the output of this architecture is used to train an AdaBoost classifier for detection. The classifier produces real-valued confidence predictions. Those results are then merged into a saliency map indicating the possibility of containing text symbols.

Our word detector is composed of two stages: coarse and fine. Coarse detection provides an initial estimation of locations of texts while fine detection prunes the detected regions with closer examinations. Both of them utilize learned codebooks from convolutional K-means (see Chapter 3).

The architectures are similar for coarse and fine detection, i.e. combining single layer network with AdaBoost. But they also differ in many aspects:

- The coarse detector has a larger stride size than fine detector, therefore, the names coarse and fine;
- The coarse detector utilizes contextual information to improve accuracy while fine detector used local information only;
- The coarse detector is applied to the entire image, while fine detector is applied after coarse detection and only to regions of interest;
- The coarse detection is applied across 30 different scales to search for texts with different sizes. On the other hand, the fine detector is applied to scale of interest and its neighboring scales;
- The coarse detector used patches extracted from ICDAR image for training while fine detector used extra synthetic data to enhance its performance.

4.1.1 Coarse Detection

The coarse detector is a combination of convolutional k-means, convolutional layer, and AdaBoost classification. The training procedure of detecting texts in natural scene images with the proposed detector includes:

1. Acquire codebook banks learned from convolutional K-means;
2. Convert the original image to gradient map using edge detection;

3. Raster scan image with 30 different window sizes;

4. Resize all extracted patches to standard size (32 by 32) with bilinear interpolation;

5. Use a convolution layer to generate features for each location for each convolutional mask;

6. Generate labels using bounding box ground truth for each patch;

7. Utilize features and labels to train a confidence rated AdaBoost classifier.

For testing, one need to:

1. Convert test image to gradient map;

2. Raster scan image with 30 scales;

3. Resize patches;

4. Convolve the patches with codebooks as in training;

5. Utilize the trained classifier to predict the patches as foreground/background with confidence scores;

6. Build a hotmap for detection prediction using classification results and patch locations;

7. Combine different scales of detection hotmaps into a final hotmap using maximal pooling.

The proposed detector is a combination of a feature learning process, a convolutional layer, and an AdaBoost classifier. The detection is modeled as a classification problem, where each location inside the image is classified as foreground or background. Details are provided below.

Comparing to CNN. The architecture differs from the original CNN based on several reasons, including the feature learning methods (convolutional K-means), contextual information, aspect ratio and rotation and relatively smaller size of the training data:

- Due to its simplicity and good performance, convolutional K-means algorithm is chosen for feature learning. So filters are not learned (update filter weights through iterations) inside the CNN with back-propagation;

- Different from image classification or recognition problems, where the entire image is classified and labeled, detection scans the image to produce a saliency map.
- Deep CNN is designed for so-called *Big Data*. An enormous amount of data is required for sufficient training. For small training data, deep networks are usually prone to overfitting.

Based on the limitations and problems listed above, our convolutional layer is different from CNN's architecture:

- A convolution is utilized to evaluate each codebook using learned features. The supervised learning is then done by AdaBoost. No back-propagations are needed;
- Coarse and fine scans are applied. While coarse detector quickly scans the image with different scales, fine detector checks each region of interest with small stride width;
- Neighboring blocks are used to train AdaBoost classifier for an improved accuracy in coarse detection;
- Small stride width, different rotational angles and aspect ratios are used to enhance accuracy in fine detection;
- A single layer convolution is implemented instead of a deep network.

As listed above, our system is different from the generic CNN and capable of addressing the unique problems of text detection problem in natural scenes. Patch shapes and scanning methodologies are designed to address context information and variations in shape and orientations. And classifier structure and training procedures are also different to accommodate relatively smaller training dataset. Coates [101] has shown that a shallow network can achieve state-of-art performance for text detection. So we can train an accurate detector with relatively small data size.

Despite the differences, the convolution layer of our system shares two common features from original CNN architecture:

- The convolutional layer convolves image with the feature banks to generate new features (codebook evaluation);

- The convolutional layer is spatial pooled to reduce the number of features while reducing sensitivity to small translations of targets;

Scale Searching. In order to catch texts in different sizes, the coarse detector scans multiple scales. The scaling can be done by resizing images and scan with fix-sized patches, or by scanning images with different patch sizes and resize patches into 32 by 32 pixels. Both approaches acquire the same results, as the order does not affect the final patch values.

The first approach is selected for our system due to its speed, where the image sizes are shrunk down for each iteration, and the patch sizes are fixed. On the other hand, the second approach extracts larger and larger patches in each iteration, bringing about slower patch extraction. Meanwhile, it resizes many patches separately instead of a single image, introducing longer overhead computation time.

In our system, for both training and testing, images are resized by a ratio of 0.9 for 30 different scales. It covers targets with about 23.59 times size differences. The image resize is done by bilinear interpolation. 32 by 32 patches are used to scan the image in different scales.

Convolution. To detect the target, the patch scanning method requires raster scanning the entire image from left to right and top to bottom. At each location, a fix-sized window is used to crop an image patch for classification. In our system, each image patch is firstly convolved with feature codebooks and then classified using AdaBoost algorithm.

The convolution for image patch can be performed with different boundary strategies, called 'same', 'valid' and 'full'. For the same mode, the result is the same size as the input image patch; For valid mode, it only computes the positions that require no padding, and the result is smaller than the original image; For full mode, it computes all positions where filter overlaps image, and the result is larger than the input image. For same and full mode, padding method is used to create values for surrounding pixels. For example, zero padding, which adds zero valued pixels, and mirror padding, which adds mirroring values around image patch boundaries are commonly used methods.

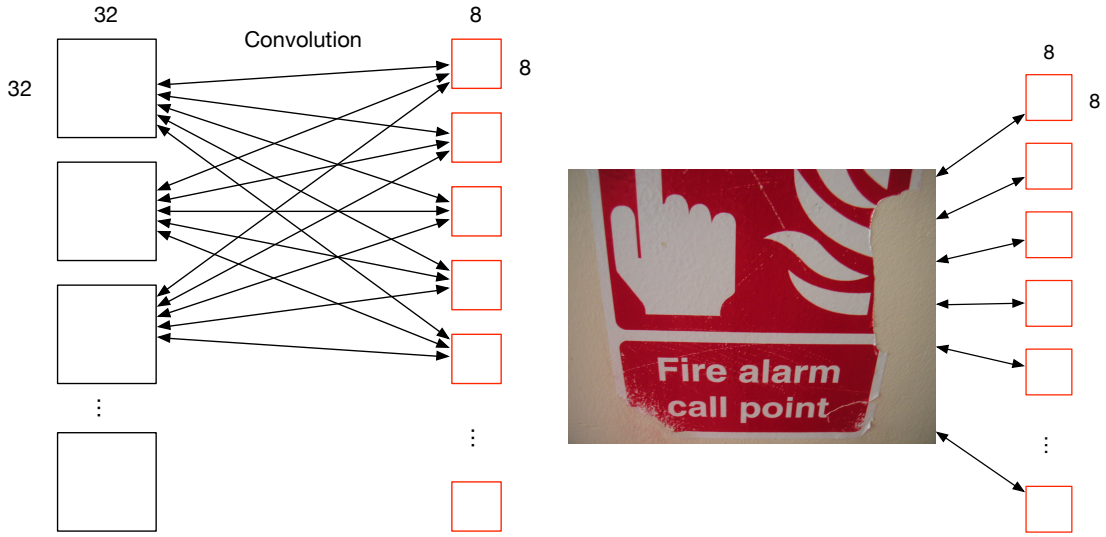
For image size of $m \times m$ and filter size of $n \times n$, 'same' mode convolution produces a matrix with size $m \times m$; 'valid' mode produces a matrix with size $(m-n+1) \times (m-n+1)$; 'full' mode produces a matrix with size $(m+n-1) \times (m+n-1)$. Here we assume that image sizes are always bigger than filter sizes ($m > n$).

In our system, 'valid' mode is chosen for convolution computation. Therefore, the number of generated

features is smaller comparing to ‘same’ and ‘full’, and the performance of feature generation is independent of padding methods.

For detection, one can extract image patches first and then perform convolution for individual patches, or one can perform convolution first and then extract patches. The second option is usually faster than the first in practice, as shown in Figure 4.2. For example, the computer usually performs the convolution using two Fast Fourier Transformations (FFTs) to convert image and filter into the frequency domain, multiplies the results, and then transforms it back to space domain. The first option requires hundreds of FFTs for a normal sized image, while and the second option requires only three [87].

As ‘valid’ boundary strategy is chosen, the result of convolution is fully determined by pixel values inside each patch, so the order of convolution and patch extraction does not affect the final results. Therefore, the FFT-based option is chosen, due to its speediness.



(a) (b)

Figure 4.2: Illustration of two convolution options. (a) The first option, extracting patches first and then compute convolutions; (b) The second option, performing convolutions before patch extraction. The first option requires nm convolutions, where n is the number of patches extracted, and m is the number of codebooks; The second option requires only m convolutions.

Gradient Image. Illumination variance can decrease the detection accuracy. Imagine a pair of identical characters, when exposure is made dark and light respectively, their convolutional responses are different. When patch extraction is performed before convolution, pixel values are normalized before convolution to compensate the difference in illumination. However, one can not normalize illuminance when convolving

image before patch extraction.

Figure 4.3: Illustration of gradient images produced by Sobel filtering. (a)Original image from ICDAR training dataset; (b)The corresponding gradient image; (c) The first two lines are original image patches and their corresponding edge patches from Tao Wang's dataset. The second two lines are corresponding background patches extracted from ICDAR images.

One way to avoid this problem is to perform convolution on the gradient image, as shown in Figure 4.3. By computing the first derivative of the image along x and y-axis, we preserve the structure of the target, but removes the constant amplitude, avoiding illuminance inhomogeneity problem.

To compute the gradient map, 3 by 3 Sobel filter is used, as shown below.

$$\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

Sobel filter in horizontal direction.

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{pmatrix}$$

Sobel filter in vertical direction. Applying Sobel filtering in x, y direction will generate horizontal and vertical edges correspondingly. The final edge map is then the addition of the edge maps along two orthogonal directions. Unlike Canny edge detector, these gradient maps are with real numbered values. The values at each pixel is corresponding to the local gradient of the grayscales.

The gradient map is computed once for each image, and then the convolution is computed. For 1000 features in the codebook dictionary, one need to perform 1000 times convolution for each scale, that is 30,000 convolutions for 30 different scales. Still, this is much faster than performing convolution after patch extractions.

Patch Extraction. After convolution, patches are extracted. For a 32 by 32 patch convolving with a 8 by 8 feature code, the resulting image is 25 by 25 pixels using 'valid' boundary condition, as shown in Figure 4.4. The step size for coarse detection is 16 pixels, which is the half of the patch size. This step size is large, allowing relatively quick searching, but would produce low precision of the detection.

Maximal spatial pooling with 3 by 3 blocks is then applied to each patch. Spatial pooling decreases the number of features and reduce the sensitivity to small translations.

For a CNN, the convolution layer is usually composed of three steps: convolution, spatial pooling, and Rectified Linear Unit (ReLU) [104]. ReLU is a common activation function used in neural networks (NNs), as shown in Figure 4.5, and it usually serves for the following purposes:

- For a network deeper than a single hidden layer, ReLU introduces non-linearity to the system, allowing the system to form non-linear decision boundaries;
- Because ReLU converts all negative inputs to zero, it only activates about 50% of the hidden units (having non-zero output) to achieve the desired property called sparse activation for deep network;
- For a deep network, ReLU gets very efficient gradient propagation, reducing vanishing or exploding gradient effect.

However, in our system, convolved feature values are used directly, ReLU is discarded for simplicity. By discarding ReLU, one can utilize all units responses for classification. Meanwhile, as most advantages of ReLU relates to a deep network, this modification does not degrade our classification. Another reason that ReLU can be discarded is that the classification is done by succeeding AdaBoost classifier, so no activation function is required.

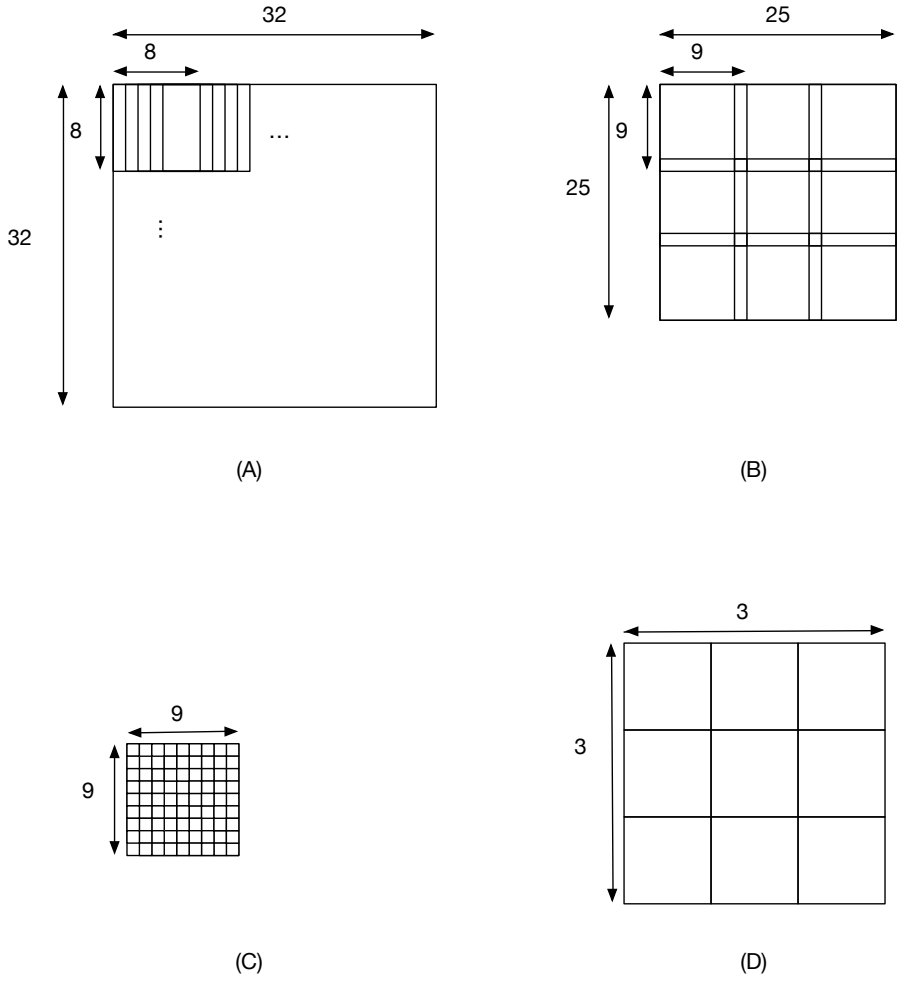


Figure 4.4: The illustration of the patch used for the convolutional layer. (A) The 8-by-8 codebook convolves with 32-by-32 image patches with stride size equals to 1. The result is (B) 25-by-25 convolutional map. To perform spatial pooling, we use 3-by-3 blocks. Therefore, each (C) 9-by-9 block from the 25-by-25 matrix is selected, the maximum number is used as block value. The blocks are overlapped by 1 element. The final result is a (D) 3-by-3 matrix.

Contextual Blocks. We found that using a standard window such as shown in Figure 4.6(a) to capture local information gives poor performance, as many false positives are generated. However, neighboring

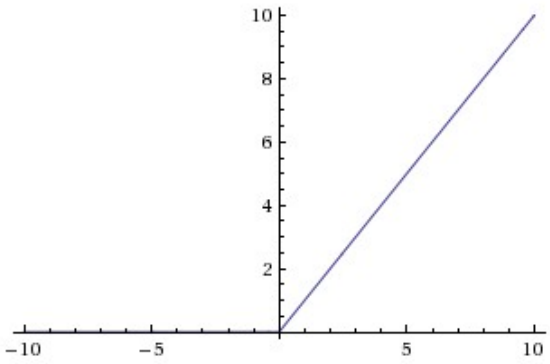


Figure 4.5: An activation function ReLU used for regular CNNs to introducing non-linearity. The function for ReLU is $f(x) = \max(0, \alpha x)$, where α is a constant coefficient. In our system, ReLU is removed for simplicity and better performance.

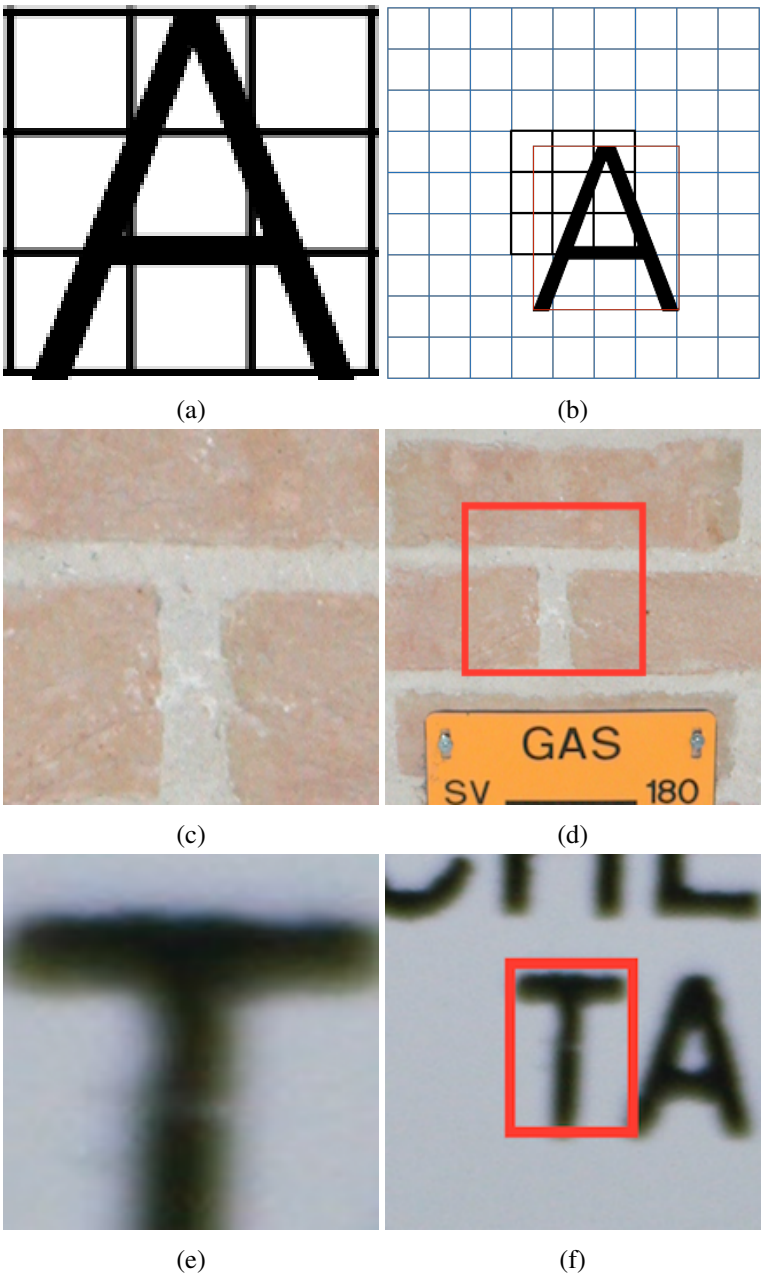
pixels provide discriminative information (see Figure 4.6 (c)-(f)). For example, the local information is not sufficient for classifying a pattern as a character 'I' or a vertical bar. On the other hand, the neighboring information provides enough features for a human to classify the image patch as foreground or background.

To consider neighboring patches during detection, we design the image patch as shown in Figure 4.6(b). The center block contains a 3×3 grids containing the target region at center. The eight neighboring blocks around the center block provide contextual information. Coarse detection produces a hotmap representing the likelihood of text. Examples of coarse detection hotmaps with and without contextual features are shown in Figure 4.11. As seen in the example, a substantial increase in discriminative power is provided by the surrounding blocks.

4.1.2 Fine Detection

Coarse-to-fine scanning divides text detection into two stages. In coarse stage, the image is scanned with a larger step size, therefore, with lower location accuracy but faster computation time. The saliency map produced by the coarse detection is then used to crop the region of interests. The scales of detection map are also used for filtering. The fine stage uses a smaller step size (1 pixel). It also searches for different aspect ratio and rotational angles to improve detection accuracy. The region and corresponding scales located by coarse detection are used to reduce the extent of searching.

In natural scene images, the text can be with different orientations and aspect ratios, as in Figure 4.7. The difference in aspect ratio is caused by perspective transformation with different camera viewing angles.



(a) (b) (c) (d) (e) (f)

Figure 4.6: Local 3x3 (a) and Contextual 9x9 (b) detection windows. Contextual windows are a standard 3x3 window surrounded by features from an 8 neighborhood. Panels (c)-(f) illustrate resolving ambiguous local features by context.

Meanwhile, symbols like ‘i’ and ‘j’ are usually with very small widths, which are different from other characters. To detect transformed aspect ratio and special symbols, like ‘i’ and ‘j’, the aspect ratio grid search is performed. The final output is a maximal pooling result.

Similarly, natural scenes are often taken with small rotational angles, due to perspective transformation.

(a) (b) (c)

Figure 4.7: Examples of characters with different aspect ratios, where red bounding boxes marked the characters (words) of interest. The variations in aspect ratios are partly because of variation of viewing angles of cameras, but also because of different characters, like 'i' and 'j'. (a) The aspect ratio variation caused by camera viewing angles; (b) The aspect ratio variation caused by different characters; (c) The aspect ratio variation caused by the curve of the surface.

Some characters are written in curved lines, which also introduce rotations, as in Figure 4.8. Therefore, rotational angles are also grid searched and maximal pooled.

(a) (b) (c)

Figure 4.8: Examples of characters with different rotational angles, where red bounding boxes marked the characters (words) of interest. The variations in rotational angles are caused by changing of camera viewing angles and changing of orientations of text lines. (a) The angle variation caused by camera rotation; (b) The angle variation caused by curved text lines; (c) The angle variation caused by the curve of the surface.

For fine detection, local information is utilized alone classifying patches as foreground or background. We compute aspect ratios from 0.6 to 1.4 with step size as 0.2. We also compute rotational angles from -6 to 6 degrees with step size equals 2 degrees.

4.1.3 Classification

The goal of AdaBoost is to construct a strong learner by combining a list of weak learners.

The weak learners are built by decision stumps using convolutional mask response features. A weak

classifier is selected in each iteration based on its error rate on the training data. The samples are then re-weighed, the wrongly classified samples are weighted heavier, and the correctly classified samples are weighted lighter. Then next weak classifier is selected according to performances on newly re-weighed samples. After many iterations, there are a handful collection of weak classifiers, and the final strong classifier is the linear combination of weak classifiers weighted on their error rates in each iteration.

This simple scheme has several advantages. First, it is less prone to over-fitting on most kinds of datasets. Second, the simple structure allows modifying the training process to build cascaded classifiers.

Viola and Jones [105] proposed a method of combining multi-stages of AdaBoost classifiers for face detection and achieved a great improvement in speed. One major advantage of this method is that it discards a lot of negative samples in early stages allowing a very fast execution. The AdaBoost algorithm in each stage makes this possible.

Third, AdaBoost utilizes simple decision stumps as weak learners, allowing tuning precision and recalls by adjusting the threshold. Therefore, V-J system manages to control the precisions and recalls in each cascaded stage. The final performance can then be tuned in this way.

In our system, the idea of cascaded filtering is utilized. For example, the coarse-to-fine architecture is actually a filtering method to increase accuracy while reducing computation time. In later procedures, we also turn our verification classifiers based on V-J's scheme.

Although the Viola-Jones method provides a good way of predicting final performance, an early mis-prediction of decision boundary would result in a bad performance on testing data in later stages [106].

AdaBoost is a boosting algorithm proposed by Yoav Freund and Robert Schapire [91], which is called a meta-algorithm, which combines other algorithms to improve the performances. Early meta-algorithms, including bootstrapping, bagging and boosting, are based on randomly data sampling.

The Bootstrapping method repeatedly draws n samples from total samples N. For each set, it estimates a set of statistic parameters. The average of the individual estimates is called the bootstrap estimate.

The bagging method repeatedly draws n samples from the total number of samples N to find a classifier for each set. The final classifier is a vote of the individual classifiers.

The boosting method is developed based on the bagging method. First, bagging is performed to obtain classifier C_1. Second, the second set of samples are drawn with half of them are misclassified samples using C_1, and used to form classifier C_2. The third classifier C_3 is trained using the samples which C_1 and C_2

disagree on. The final classifier is a vote of C_1, C_2 and C_3.

AdaBoost. The difference between AdaBoost to all other meta-algorithms is that AdaBoost re-weights the data instead of sampling the data. AdaBoost is adaptive to data by biasing towards samples which are misclassified. The purpose of AdaBoost is to combine a set of weak classifiers into a strong classifier based on their performances on the training data. The weak classifiers are required to be at least slightly better than random guesses, then the performance of the strong classifier can be improved through iterations. The final classifier is a weighted vote of the weak classifiers. The original Discrete AdaBoost is listed as Algorithm 1.

Algorithm 1 AdaBoost

Initialize sample weights $\omega_i = 1/N, i = 1, ..., N$.
for $m = 1, 2, ..., M$ **do**:
    Find best weak classifier $f_m(x) = -1, 1$, using current weights ω_i on the training data;
    Compute error rate of the weak classifier $err_m = E_\omega[1_{y \neq f_m(x)}]$;
    Compute classifier's weight $\alpha_m = log((1 - err_m))/err_m$;
    for $i = 1, 2, ..., N$ **do**:
        Update sample weights $\omega_i \leftarrow \omega_i / exp[\alpha_m 1_{y \neq f_m(x)}]$.
    end for
    Re-normalize so that $\Sigma_i \omega_i = 1$.
end for
Build final classifier $y(x) = sign[\sum_{m=1}^{M} \alpha_m f_m(x)]$.

There are also many variations of AdaBoost, addressing different problems and trying to generate enhanced performance.

Friedman etc. [107] viewed boosting as an approximation to additive modeling on the logistic scale using maximum Bernoulli likelihood as a criterion. They proposed modifications to boosting to reduce computation and achieve better performance. For discrete AdaBoost, the proposed modification is called LogitBoost, where classification results of weak classifiers remain as $-1, 1$, but using an adaptive Newton algorithm for fitting an additive logistic regression model.

The Gentle-Boost is a modified Real-Boost (Algorithm 2), where Newton stepping method is used as in LogitBoost. They claimed that LogitBoost and GentleBoost could achieve comparable performances to Real-Boost but have lower computational complexity.

Real AdaBoost. AdaBoost with confidence-rated predictions, or called Real AdaBoost is proposed by Schapire etc. in 1999 [108]. This method gets self-rated confidence scores which estimate the reliability of

each prediction. By introducing confidence rate, a more generalized AdaBoost is proposed, where α can be tuned to improve performance. Real AdaBoost usually converges to a smaller error rate while faster to train in many cases.

Confidence-rated Real AdaBoost is used in our text detector to classify patches into foreground and background. Unlike original AdaBoost which produces hard labels as -1 and 1, confidence-rated AdaBoost classifier could produce not only label but also confidence that a sample belongs to that label. The higher number indicates a higher likelihood that a sample belongs to the foreground and vice versa. The pseudocode of the algorithm is shown below.

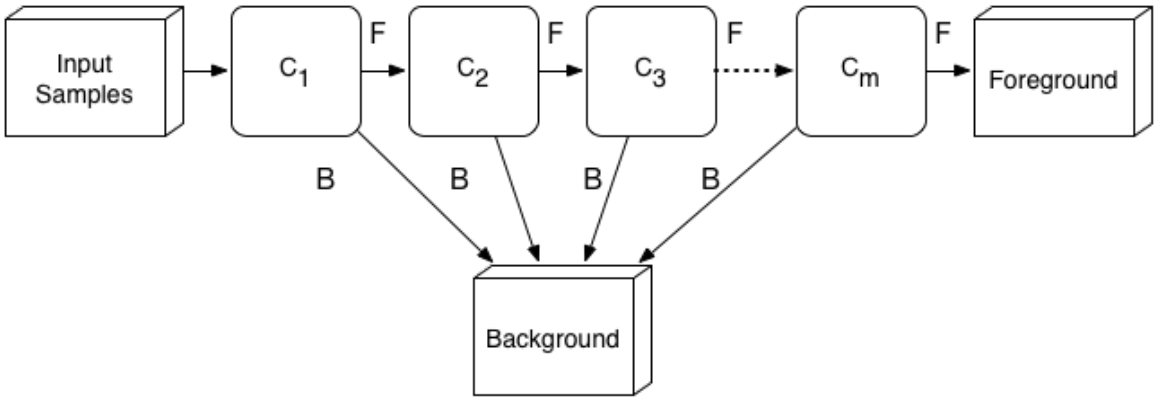


Figure 4.9: VJ Cascaded AdaBoost classifier for object detection. The classifier filters easy negatives in early stages using relatively simple classifiers and deal with hard samples in later stages. Each stage is a filter, where only positive predictions are used in later stages.

Cascaded AdaBoost. To increase speed while preserving accuracy, a Viola-Jones fashioned cascaded classifier is tested and compared with conventional Real-AdaBoost. The cascaded classifier is designed to remove easy negative samples in early stages while preserving high recall value. Using multiple steps of filtering, the classifier removes negative samples to achieve the desired precision and recall, as shown in Figure 4.9.

The text detection systems need to deal with similar speed problems as in face detection: A lot of samples need to be processed from images (raster scanned samples in different scales).

In a standard Viola-Jones model, the precision and recall value are predefined and fixed. The AdaBoost classifier is trained using only a small number of features in early stages (two features for the first stage), which removes a lot of negatives. The recall is fixed at a very high value while precision is allowed to be as

Algorithm 2 Real-Boost

1: Select 2N training samples, with N foreground samples and N background samples. Each sample is composed of T features;
2: Initialize the sample weights to be 1/2N for each sample;
3: **repeat**
4: Build decision stump classifiers, and find the best threshold values with the lowest weighted errors.
5: For all decision stump classifiers, compute the following possibilities:

$$\begin{array}{l} P_r^+(j) = \sum p_i \text{with summation is over examples, } x_i \text{ satisfying } h_j(x_i) = 1, y_i = 1 \\ P_r^-(j) = \sum p_i \text{with summation is over examples, } x_i \text{ satisfying } h_j(x_i) = -1, y_i = -1 \\ P_w^+(j) = \sum p_i \text{with summation is over examples, } x_i \text{ satisfying } h_j(x_i) = -1, y_i = 1 \\ P_w^-(j) = \sum p_i \text{with summation is over examples, } x_i \text{ satisfying } h_j(x_i) = 1, y_i = -1 \end{array} \tag{4.1}$$

$$G(j) = \sqrt{P_r^+(j)P_w^-(j)} + \sqrt{P_w^+(j)P_r^-(j)} \tag{4.2}$$

6: Choose the decision stump classifier h_j that minimize $G(j)$. Here i indicates the ith iteration, and j indicates the jth classifier.
7: Calculate weights c_t^+ and c_t^- and the function g_t as follows:

$$c_t^+ = \frac{1}{2}\log\frac{P_r^+(t)+\epsilon}{P_w^-(t)+\epsilon} \tag{4.3}$$

$$c_t^- = \frac{1}{2}\log\frac{P_w^+(t)+\epsilon}{P_r^-(t)+\epsilon} \tag{4.4}$$

$$g_t(x) = \begin{cases} c_t^+ & h_t(x) = 1; \\ c_t^- & h_t(x) = -1. \end{cases} \tag{4.5}$$

where the small positive value ϵ is 1/(8N).
8: Update the sample weights, $w_i = w_i e^{-y_i g_t(x_i)}$;
9: Normalize the sample weights to have summation to 1;
10: **until** The error rate is lower than a target or maximum number of iterations is reached.
11: The final classification is given by

$$f_T(x) = \sum_{t=1}^{T} g_t(x) \tag{4.6}$$

where t is the iteration number, T = 1000 for our experiment.

low as slightly higher than 50%. After multiple stages, the final overall recall value is,

$$R = r_1 \times r_2 \times r_3 \times ... \times r_n \tag{4.7}$$

where R is the overall recall value and r_i is the actual recall for each stage. And final precision value is got by

$$P = 1 - (1 - p_1)(1 - p_2)(1 - p_3)...(1 - p_n) \tag{4.8}$$

where P is the overall precision value and p_i is the actual precision for each stage.

For a system containing more than 10 stages, the recall value must be kept fairly close to 100% in each stage to get an acceptable overall recall. In our experiments, it is fixed at 99% acrossing all stages. On the other hand, the overall precision gets improved through stages. A high precision of overall system can always be achieved by adding enough stages.

It is easy to tune the stage precisions. However, it is very unlikely that one can adjust the stage recalls since they are very close to 100%. If stage precisions are tuned higher, fewer stages are required for a certain overall performance. However, more features are needed for each stage. We tested different precision thresholds through different stages and compare their influences to the overall performance on the testing data.

The experiments are done in three different cases:

- Increasing threshold from low to high values through 30 stages. For a 30 stages cascaded system, the precision is 50% for the first stage and increased by 1% in each successive stage. So the last stage precision is 80%;
- Decreasing threshold from high to low. The precision is 80% for the first stage, decreased by 1% each stage and ends at 50% for the last stage.
- Using zero gradient of f-measure in each stage;

The precision values are tuned in the training process. Once the threshold is fixed for each stage, testing samples usually get different precision and recall values for each stage classifier. So the final performance could be different.

Another method to control the precision threshold is to find the zero gradient point of overall F-measure. The precision and recall values change along with the threshold values. For very small thresholds, precision is low, since the probability for a sample being classified as the foreground is high. For very large threshold values, recall is low, since strict criterion removes most of the samples from the foreground. By tuning threshold from negative to positive values, precisions are changed from 0 to 1 and recall values are changed from 1 to 0.

The precisions are monotonically decreasing w.r.t. recall values. In each stage, we would like to find the a threshold that maintains a high recall but also has the highest precision value.

The goal is to maximize the overall F-metric of the system, which is given by the equation 4.9.

$$F = \frac{2PR}{P+R} \tag{4.9}$$

where P represents overall precision and R represents overall recall. To find the maximum value, one could simply compute the gradient of the equation and make it equal to 0. This only ensures an extremum in common cases. However, as P or R is always greater than 0 for different thresholds, so maximum values are given by this equation. Here two calculus equations are used to compute derivatives:

$$(fg)' = f'g + fg' \tag{4.10}$$

$$\frac{f}{g} = \frac{f'g - fg'}{g^2} \tag{4.11}$$

Where f' and g' are generic terms for functions. If extremum value needs to be found for overall F-measure, one only needs to set the derivative of equation 4.9 to zero, which is

$$\frac{(2P'R + 2PR')(P+R) - 2PR(P'+R')}{(P+R)^2} = 0 \tag{4.12}$$

where P' and R' are derivative of overall precision and recall with respect to threshold. As for P and R could not be 0 at the same time for any threshold,

$$(P+R)^2 > 0 \tag{4.13}$$

therefore, we look for

$$(2P'R + 2PR')(P + R) - 2PR(P' + R') = 0 \tag{4.14}$$

$$PP'R + P^2R' + P'R^2 + PRR' - PP'R - PRR' = 0 \tag{4.15}$$

$$P^2R' = -P'R^2 \tag{4.16}$$

$$\frac{P^2}{P'} = -R^2R' \tag{4.17}$$

Notice that the P and R represent values in overall system, they can be converted to stage-level values using equation 4.7 and 4.8.

Using this simplification, equation 4.7 and 4.8 becomes

$$R = (r)^n \tag{4.18}$$

$$P = 1 - (1 - p)^n \tag{4.19}$$

where r and p are recall and precision of current stage classifier. Substitute this equation into 4.17, we have

$$\frac{(1 - (1 - p)^n)^2}{n(1 - p)^{n-1}p'} = -\frac{(r^n)^2}{nr^{n-1}r'} \tag{4.20}$$

$$\frac{1 - 2(1 - p)^n + (1 - p)^{2n}}{np'(1 - p)^{n-1}} = -\frac{r^{n+1}}{nr'} \tag{4.21}$$

$$\frac{1}{np'(1 - p)^{n-1}} - \frac{2(1 - p)}{np'} + \frac{(1 - p)^{n+1}}{np'} + \frac{r^{n+1}}{nr'} = 0 \tag{4.22}$$

This is the most simplified closed form that we could get, and the threshold for this equation is then searched

numerically in experiment.

Actually, this equation is very easy to be computed as once the function of (1-p), p', r and r' is known, one can quickly compute the values within the searching range. Practically, the precision value should be more than 50%, otherwise, the overall performance will be decreased instead of increased by current stage. On the other hand, we will limit the precision values to be lower than 80%, therefore, a single stage can not use too many decision stumps to slow down the process. By limiting the searching range from 50% to 80% precision, one could search for the numerical solution of the equation very quickly for each stage. And this solution is related to the number of stages used for the entire cascaded architecture n.

4.2 Experiments

We tested our system on ICDAR 2015 focused scene image dataset, containing 258 training images and 251 testing images.

First, 32 by 32 patches are extracted from training images. The patches can be categorized into foreground and background as labels of those patches are known or derived from bounding box labels. Patch labeling is dependent to bounding box ground truth, patch size and step size of patch extraction.

For coarse detection, patch sizes are 32 by 32 and step sizes are 16, the neighboring scales have a ratio factor of 0.9. When the patch scans across the foreground character bounding box, the minimum overlap ratio along one dimension is 0.75. So for two dimensions, the worse case overlapping ratio is $0.75^2 = 0.56$. Therefore, the definition for foreground patches is: if a patch is less than 10% different from the width or height of a character bounding box, and the overlapping area is greater than $0.75^2 = 0.56$, then it is considered a foreground patch.

In order to extract foreground samples, one should raster scan images with multiple scales (30 different scales in our experiment) and extract all qualified foreground patches, meanwhile, get corresponding background samples as well. In our experiment, the images are resized, and patches are extracted with constant 32-by-32 pixels.

Background samples are more frequently extracted than foreground samples, as background area is usually much larger than foreground area in natural scene images. We subsample the background to equalize the sample numbers using random sampling. In practice, the equalization is done within each image, and

within each scale. Therefore, fewer numbers of background samples need to be kept, saving memory and disk space.

In our experiment, 72588 samples are extracted from the training images, with 36294 positive and 36294 negative samples. We divide the samples into 80% training and 20% testing using random selection. Therefore, 58070 samples are used for training the classifier, and 14518 samples are used for evaluation. The training and testing samples are all generated from ICDAR 2015 training dataset, and the testing images are not used. The testing images are saved for the final evaluation of our systems to avoid overfitting to the data.

To train the fine detector, 50,000 randomly selected foreground images from Wang's synthetic dataset [61] are used, along with 50,000 randomly selected background samples generated from ICDAR images. Increasing numbers of samples enhance the training of the classifier. Although the fine detector is not trained using contextual features, the additional training sample could help it get comparable performance.

After training the coarse and fine detector, the detector is applied to ICDAR 2015 testing dataset for evaluation. For coarse detector, 30 different scale patches are extracted and fed into the AdaBoost classifier. The prediction in different scales are shown in Figure 4.10, indicating the likelihood of being foreground in different scales. The final hotmap is a maximal pooling combination of all 30 different scale hotmaps.

Contextual information is used. An comparison of detection hotmap from classifiers with contextual feature and without is shown in Figure 4.11 , where contextual feature based classifier generate hotmap separate foreground and backgrounds more clearly and with less false detection. The fine classifier is trained using fewer features (local blocks only) but with more training samples. And the performance for both coarse and fine detectors are shown in Figure 4.12 and 4.13.

The precision, recall, and F-measure is plotted in Figure 4.12 for both coarse and fine classifiers. The F-measure maximum point is near the equilibrium where recall and precision have same value. However, that is not necessary the case and mostly F-measure peaks where recall is higher than precision.

The Receiver Operating Characteristic curve is shown in Figure 4.13, which represents the classifier's performance in the entire performance envelope. It shows the power of a classifier separating two class of samples, and closer the curve to perfect point means better separation. Both coarse and fine detector has great accuracy and ROC curve very close to the perfect point. However, fine classifier shows higher accuracy.

The cascaded classifier is then tested to increase speed while maintaining the accuracy. The standard

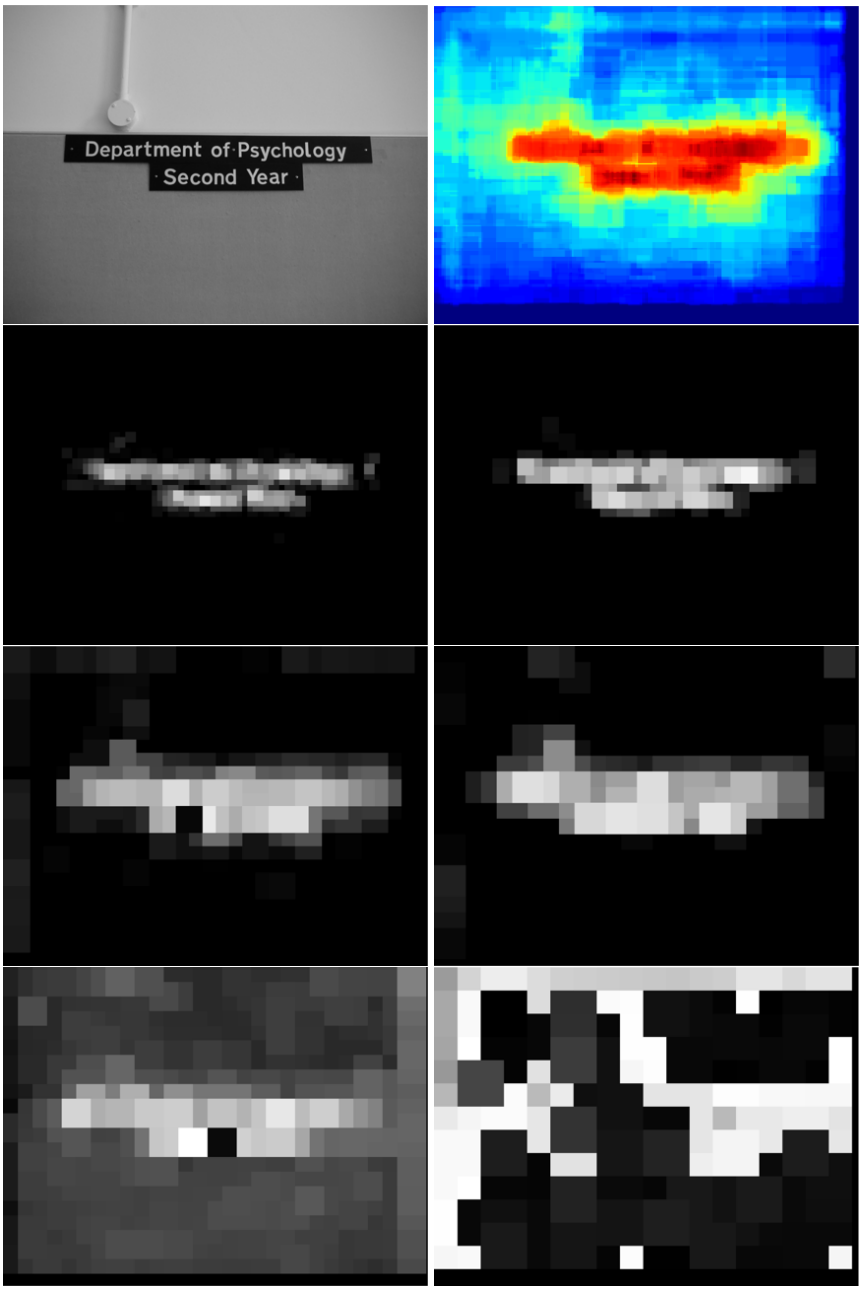
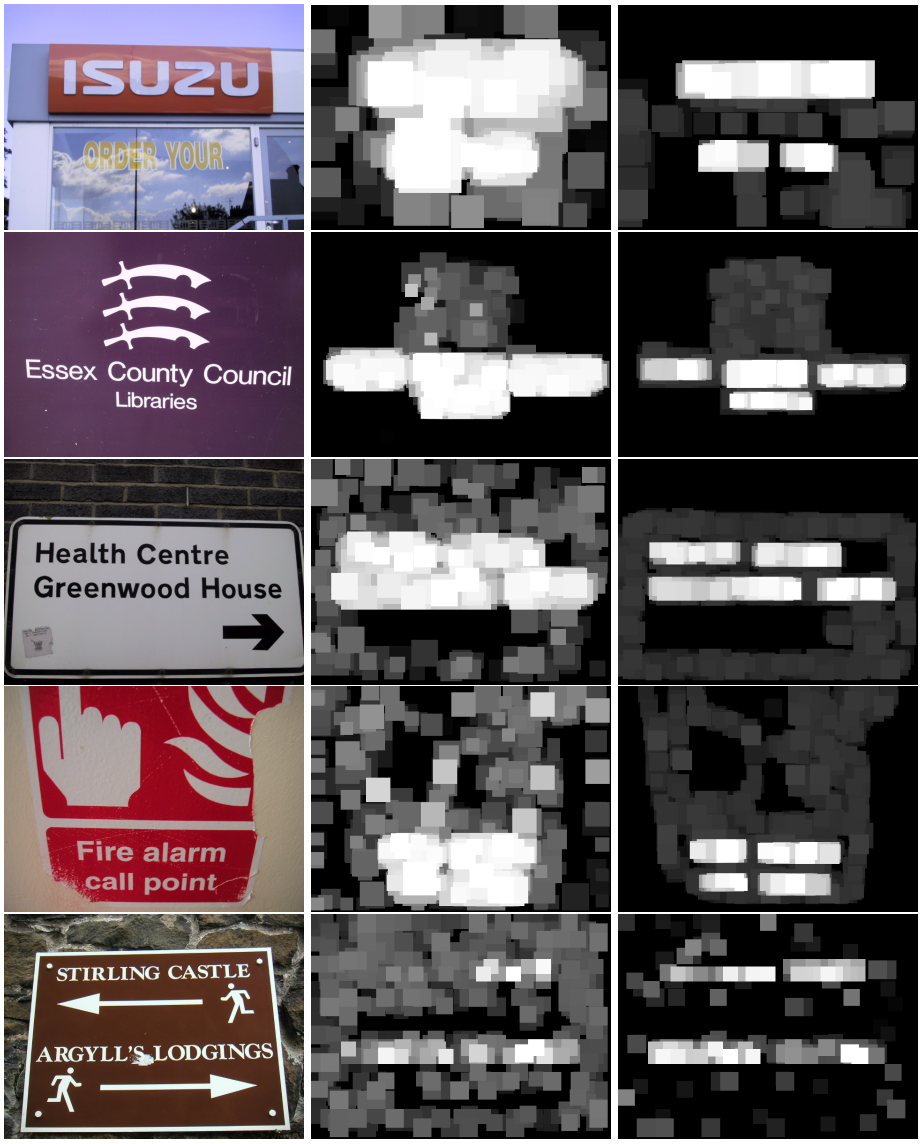


Figure 4.10: A illustration of text detection in different scales using contextual feature. The final detection hotmap is a maximal pooling result of all scales. The first subimage shows the original grayscale image for detection; the second subimage shows the final hotmap produced by coarse detector. The following subimages shows the hotmap of detections in six different scales, while the actual experiment is done in 30 different scales.

cascade, increasing precision, decreasing precision and zero gradient methods are tested for coarse detectors.

Figure 4.14 illustrate the number of samples remains at each cascaded stage. The samples are filtered at

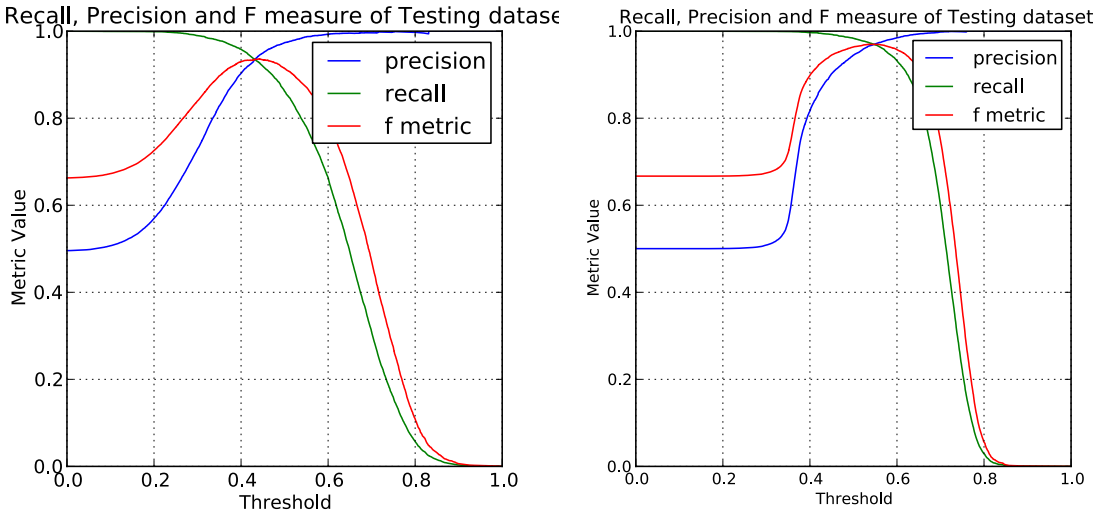


(a) (b) (c)

Figure 4.11: Comparison between local with contextual patch based detector. The hotmap indicates the classifiers response for a coarse stage classification. Brighter patch indicates higher response value from classifier, which means it is more likely to contain text. The hotmap is maximum pooled over 30 different scales. (a) The original image; (b) The hotmap generated from local feature based classifier; (c) The hotmap generated from contextual feature based classifier.

each stage. Therefore, the later stage classifiers work on a relatively smaller sample sizes. Different cases are illustrated in Figure 4.14.

For zero gradient method, the precision/recall curve, ROC curve and time consumption at each stage is illustrated in Figure 4.15. As stages go higher, the performance of the stage classifier also increases.



(a) (b)

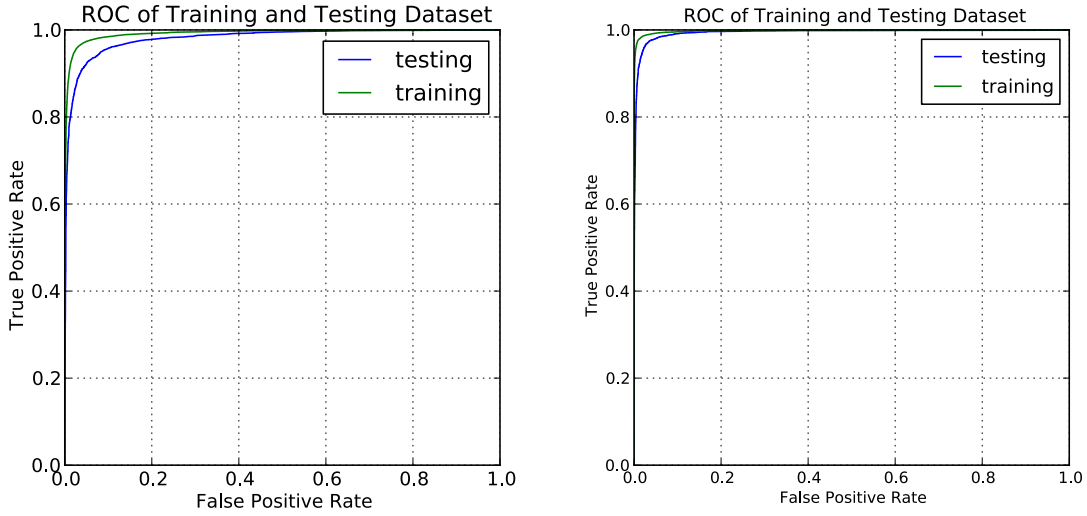
Figure 4.12: Performance of classifiers, including precision, recall and f-measure plot for different thresholds of a real-AdaBoost classifier for (a) coarse, (b) fine detector. According to the plot, the highest F-measure occurs near the point where precision equals recall value, however, this is not the case when observing more closely. The highest F-measure point is not necessarily the equal point, but a little drifted from that point towards left.

The increases in accuracy are reflected by curves that are closer to point [1, 1] as in Figure 4.15 (a) for a precision/recall curve, and curves closer to point [0, 1] as in Figure 4.15 (b). The time consumptions for all stages are shown in Figure 4.15 (c), the later stage is significantly faster, due to fewer data remained to classify, though the higher stage classifier is more sophisticated.

4.3 Result and Analysis

The contextual features are used to enhance the classification accuracy for coarse detection. The detectors performances are evaluated on ICDAR 2015 testing images in both pixel level and bounding box level by using local features only and combined contextual features.

The results are shown in Table 4.1. According to experiment results in Table 4.1 and Figure 4.16, the contextual feature successfully enhanced the performance on the testing dataset using pixel level and BB level ground truth. Th precision, recall and F-metric are increased comparing to classifiers using local features only.



(a) (b)

Figure 4.13: ROC for Detection
Receiver Operating Characteristic Curve for (a) coarse; (b) fine detector. A curve closer to the perfect classification point [0, 1] indicates a better classifier that separates the foreground and background. Our classifier ROC curves show very good performance for both coarse and fine detectors, which indicate the effectiveness of AdaBoost classifier and feature learning algorithm.

Table 4.1: Comparison of performance using local features and contextual features. The overall performances for the whole system are shown.

	Pixel Level(%)			BB Level(%)		
Methods	Recall	Precision	F-metric	Recall	Precision	F-metric
Local	85.57	81.73	83.61	76.44	82.04	79.14
Contextual	92.71	87.33	**89.94**	81.02	93.39	**86.77**

We compare the performance of different classification algorithms. The Support Vector Machine (SVM), Random Forest (RF), AdaBoost and cascaded classification (VJ) are tested and compared.

The SVM is trained with linear kernel and error term penalty parameter C equals to 10. The random forest is trained using 10 trees with non-specified maximal depth. Each node split considers $\sqrt{81000} = 285$ features. The performances are evaluated using pixel level and bounding box level ground truth on ICDAR 2015 testing dataset, and shown in Table 4.2 and Figure 4.17.

According to Table 4.2, AdaBoost algorithm provide the best performance on testing data. The precision, recall and F-measures are higher than other algorithms. The advantage of AdaBoost is its simplicity, as it is

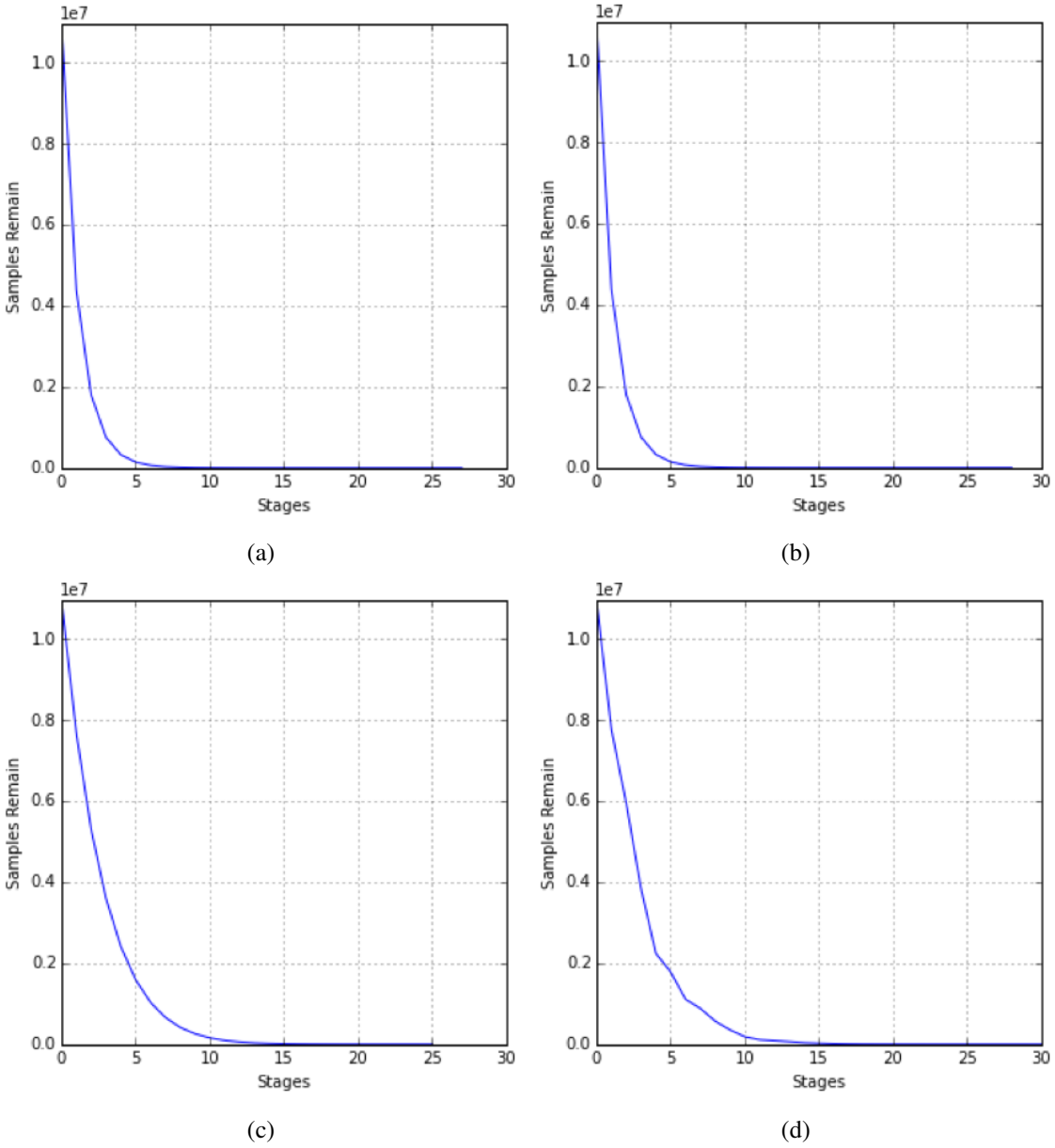
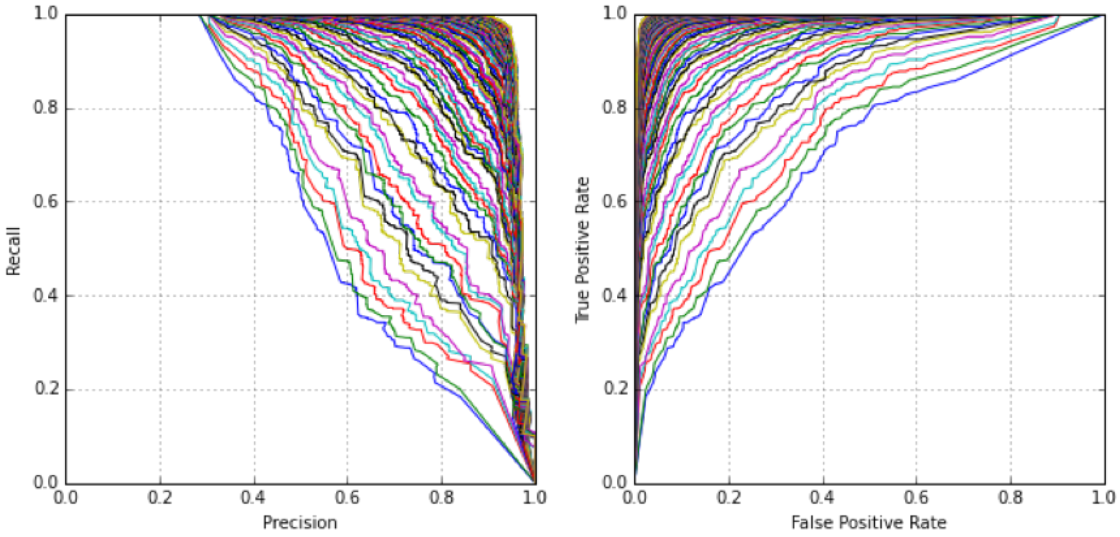


Figure 4.14: The number of samples left after each stage of classification cascade. (a)Standard cascade with fixed thresholds; (b)Increasing threshold; (c)Decreasing threshold; (d) Zero gradient threshold. All cases use 1000 decision stumps totally across the entire cascade classification.

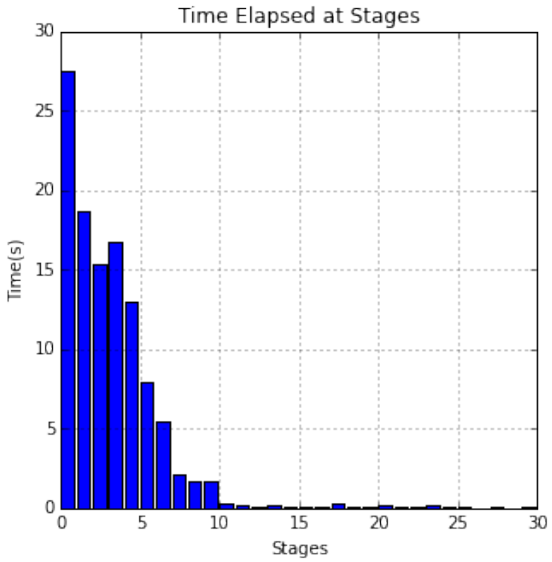
a linear combination of weak classifiers. While each weak classifier gets slightly better than random guess performance, the assembled strong classifier can be very efficient.

Another advantage of AdaBoost is it is less sensitive to the overfitting problem. For example, in random forest algorithm, if the tree depth is too large, the forest will degrade to decision tree classifier which does not



(a) Precision/Recall

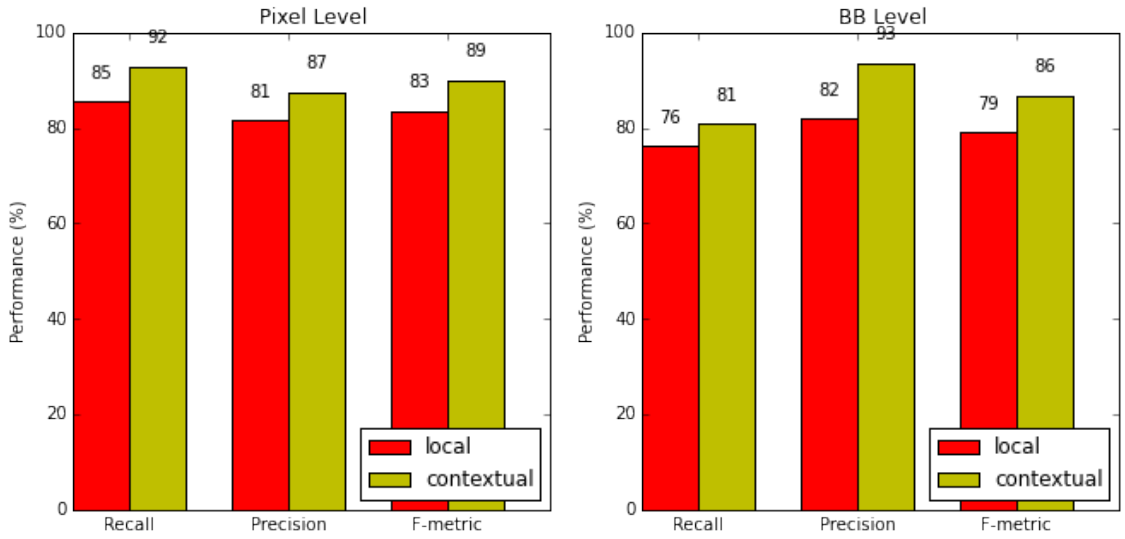
(b) ROC



(c) Elapse Time

Figure 4.15: The performance evaluation using zero gradient cascaded classifiers. The (a) precision/recall curve, (b) ROC curve and (c) Time taken for all iterations are plotted and shown. The precision/recall curve is computed for each stage classifier, and later stage classifiers have much better performance curves (closer point [1, 1]). The ROC is also plotted for stage classifiers, later stage classifiers have curves closer to perfect point [0, 1]. The time taken by each stage classifier indicates that later stage spent less time than early stages. Although later stage took more features to satisfy performance criterions, due to reduced number of samples, the later stages are faster.

generalize well on training data. For other classification algorithms, early termination is a way to manually reduce the overfitting effect and require careful engineering and human interventions.



(a) (b)

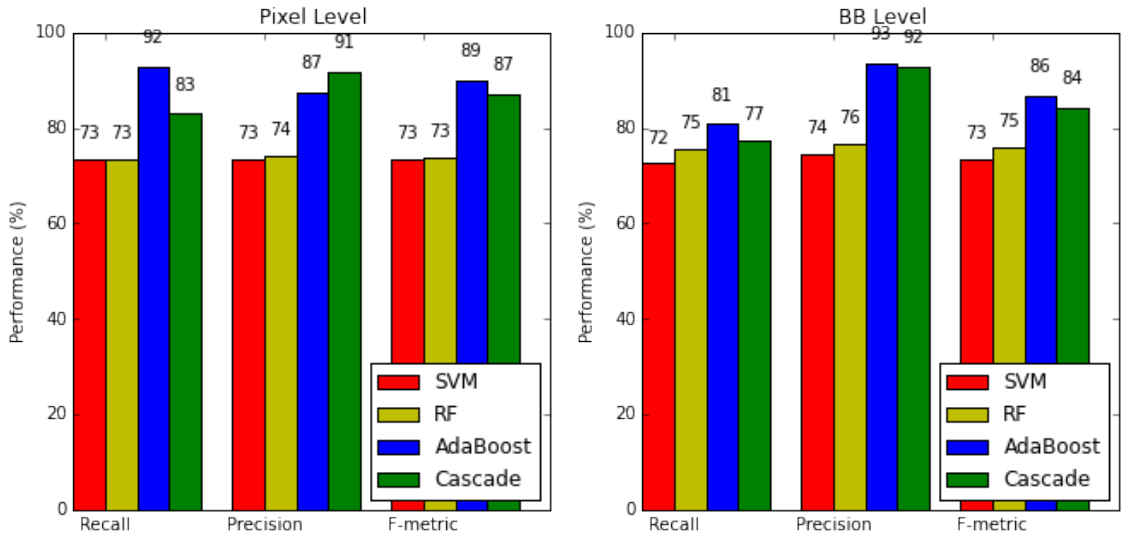
Figure 4.16: Comparison of performance using local features and contextual features, in both (a) pixel level and (b) BB level.

Table 4.2: Comparison of different classification algorithms' performances, including SVM, Random Forest (RF), AdaBoost and Cascaded (VJ fashioned).

	Pixel Level(%)			BB Level(%)		
Methods	Recall	Precision	F-metric	Recall	Precision	F-metric
SVM	73.32	73.54	73.43	72.55	74.36	73.44
RF	73.47	74.11	73.79	75.43	76.49	75.96
AdaBoost	92.71	87.33	**89.94**	81.02	93.39	**86.77**
Cascade	83.01	91.67	87.13	77.27	92.63	84.26

The AdaBoost classifier performs well on text detection problem is also because of the data characteristics. As mentioned by Coates [109] and Bengio [84], the text features are simple comparing to other objects, like faces. This allows AdaBoost algorithm to achieve great performance.

The AdaBoost classifiers with different iterations are tested to find if 1000 iterations are necessary for training an acceptable classifier. The classifier is stopped at 400, 600, 800, and 1000 iterations, as in Table 4.3 and Figure 4.18. The performance in both pixel and bounding box level are shown. Apparently, the performance of the final detector monotonically increases with the number of iterations used in AdaBoost. However, the performance increase slows down after 600 iterations, with 81.10% F-measure for the overall system. This indicates a smaller number of iterations can be used to accelerate the system while sacrificing



(a) (b)

Figure 4.17: Comparison of different classification algorithms' performances, including SVM, Random Forest (RF), AdaBoost and Cascaded (VJ fashioned), in both (a) pixel level and (b) BB level.

Table 4.3: Performance with different numbers of iterations. The larger iteration number produce better performance of detection in both pixel and bounding box level.

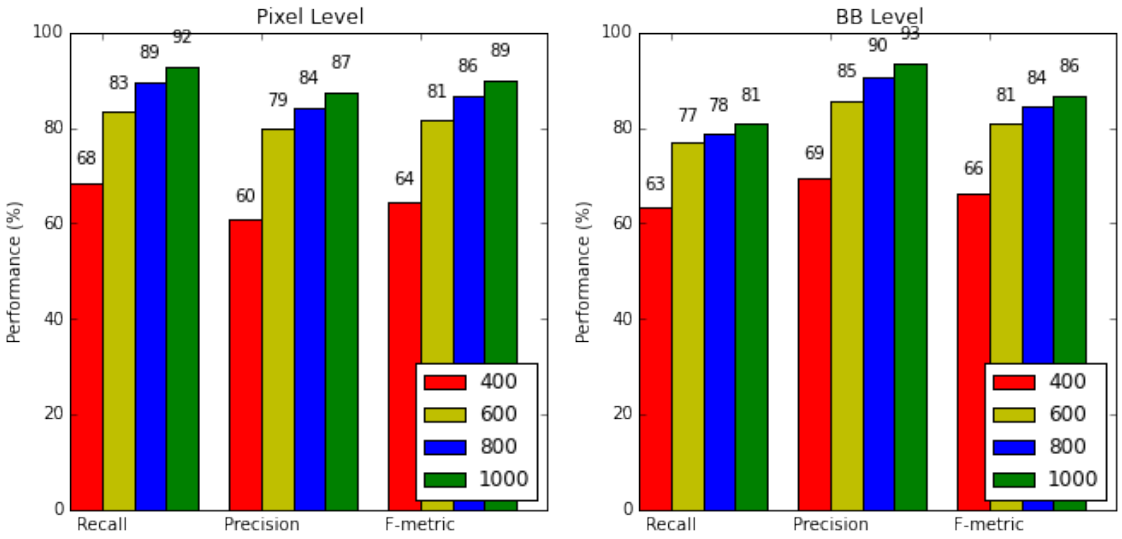
	Pixel Level(%)			BB Level(%)		
Methods	Recall	Precision	F-metric	Recall	Precision	F-metric
400	68.22	60.78	64.29	63.23	69.30	66.13
600	83.32	79.96	81.61	77.02	85.64	81.10
800	89.40	84.27	86.76	78.93	90.73	84.42
1000	92.71	87.33	**89.94**	81.02	93.39	**86.77**

the performance mildly.

We tested the overall performance with pixel level and bounding box level for cases where fine detector searched or not searched different aspect ratio and rotational angles, in order to evaluate the necessity of proposed grid searching. According to Table 4.4 and Figure 4.19, the F-measure of overall system reduces from 86.77% to 64.72% without either aspect ratio or rotational angle searching. The aspect ratio searching shows more significant effects towards final accuracy. While fixing aspect ratio will reduce F-measure to 68.50%, fixing rotational angles reduces F-measure to 74.79%.

This performance comparison shows that the testing images may contain larger aspect ratio variance than rotational angle variance. On the other hand, it may also indicate that our patch based classifier is more robust against different angles than different aspect ratios.

In classification, the acceleration method is to use cascaded classifier as described in Section 4.1.3.



(a) (b)

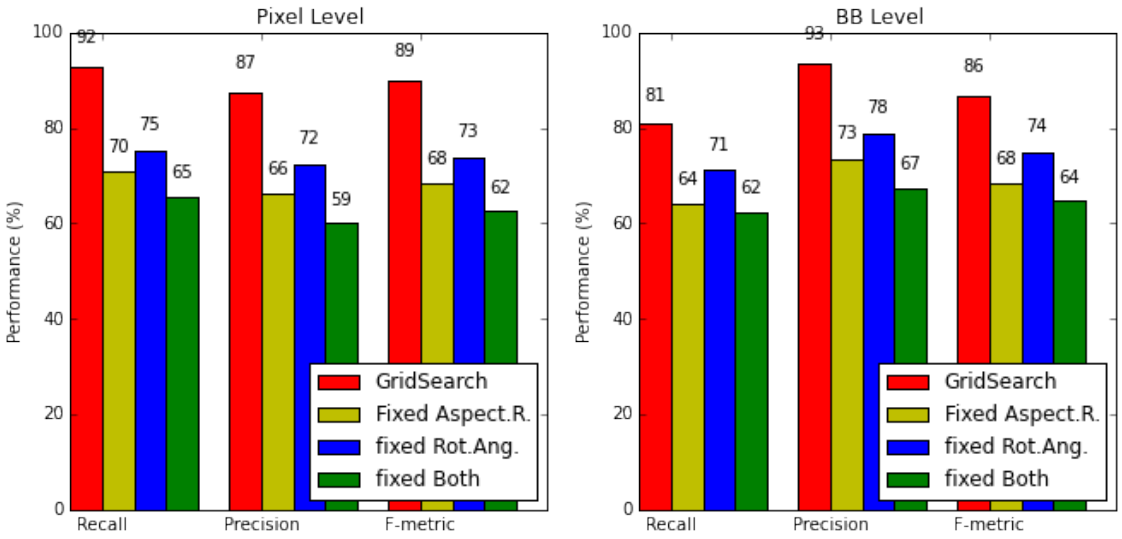
Figure 4.18: Performance with different numbers of iterations, including 400, 600, 800 and 1000 iterations in (a) pixel and (b) BB level

Table 4.4: Comparison of fine detection with and without different aspect ratios (Aspect.R.) and rotational angles (Rot.Ang.). Without varying aspect ratios and rotational angles, the detection performance degrades significantly.

	Pixel Level(%)			BB Level(%)		
Methods	Recall	Precision	F-metric	Recall	Precision	F-metric
Our system	92.71	87.33	**89.94**	81.02	93.39	**86.77**
fixed Aspect.R.	70.73	66.32	68.45	64.18	73.44	68.50
fixed Rot. Ang.	75.27	72.39	73.80	71.21	78.75	74.79
fixed	65.44	59.98	62.59	62.41	67.20	64.72

Cascaded classifier uses a small number of features to remove a lot of easy negatives in early stages, and pass the hard samples into later more sophisticated classifiers. The overall computational time can be reduced, especially for text detection case, where a majority amount of time is spent for convolving feature codebooks with images. If one can reduce the number of codebook convolutions, the computation time is reduced significantly. A performance comparison using AdaBoost and cascaded classifier is shown in Table 4.5 and Figure 4.20.

Here, confidence-rated Real AdaBoost is used as a benchmark for comparison. All cascaded classifier produces lower accuracy than single stage AdaBoost. Although precision and recall could be controlled in a V-J model, they are only precise for training data. For testing data, error rates are usually higher than training, and as miss-classification accumulate through 30 stages, the final performance decrease. To achieve



(a) (b)

Figure 4.19: Comparison of fine detection with and without different aspect ratios (Aspect. R.) and rotational angles (Rot. Ang.) in (a) pixel level and (b) BB level.

better F-measure, one could use different strategies to learn different classifier in each stage. One method is to increase the precision threshold from 50% to 79% through 30 stages, another is to decrease precision from 79% to 50%.

In increasing case, fewer features are used in early stage, therefore, the initial speed is faster. However, as precision value is low, more false samples are kept, therefore, more computation needs to be performed in later stages. In decreasing case, more features are used to achieve high precision, which means more convolution computations on initial samples. However, as more false samples are pruned, the later stage gets faster. Overall, the speed of different thresholding strategy is a trade-off between early and late accelerations.

As shown in Table 4.5, the standard cascade classifier uses fixed thresholds for precision and recall for all stages. The precision threshold is fixed at 50% and recall is fixed at 99%. The increasing-and-decreasing-precision case uses different thresholds for different stages. The increasing case precision is changed from 50% to 79% through 30 stages while decreasing case precision is changed from 79% to 50%. The zero gradient method consider the gradient of the F-measure of the overall performance and tries to find the extremal point which corresponds to largest overall F-measure. The approximation is used to achieve a closed form equation for the precision threshold. Decreasing threshold achieved better performance while more features are used in early stages. Therefore, the computation is slower. Increasing threshold used

Table 4.5: Comparison of cascaded classifier performances with different thresholding methods. The original AdaBoost classification performance without cascade is used as a benchmark, its accuracy is higher than all cascaded cases, but slower.

	Pixel Level(%)			BB Level(%)			Elapse Time (s)		
Methods	Recall	Precision	F-metric	Recall	Precision	F-metric	Conv.	Class.	Total
Normal	83.01	91.67	87.13	77.27	92.63	84.26	281	141	422
Inc. Pre.	75.76	90.64	82.53	70.25	92.84	79.98	222	127	**349**
Dec. Pre.	85.92	91.17	**88.47**	78.84	92.78	**85.24**	361	132	493
Zero grad.	85.59	91.35	88.38	78.34	92.61	84.88	289	138	427

fewer features in early stages, achieving fastest execution time. However, the zero gradient method gets more balance performance and speed. It achieves comparable computation time with standard cascade and comparable performance with decreasing precision threshold.

According to Table 4.5, the increasing precision case takes much less time than decreasing. However, decreasing case gets higher accuracy. It is because strict criterions are set and more discrimination power is concentrated to early stage classifiers, rather than spread across all stages. This setup is more like a single stage AdaBoost, rather than a cascaded classifier.

Zero-gradient method is introduced to tune precision thresholds for training and testing based on an expectation of overall F-measure maximization. The overall F-measure is a function of each stage classifier's metrics, and its maximum value occurs when its gradient equals zero. We simplify the relationship function between overall F-measure and precision, recall values in each stage and set the threshold to the point which makes F-measure gradient zero, as in Equation 4.17.

Overall F-measure of 86.77% was achieved using single stage AdaBoost. For cascaded case, decreasing threshold gets the best performance of 85.24% F-measure. The shortest computation time is achieved by increasing precision threshold case with 349 seconds per image average. Considering the classification needs to process samples generated from all positions across the image with different scales. Zero gradient case generates compatible performance and speed to top methods.

4.4 Summary

The patch scanning based approach for text detection is discussed in this chapter. The patch scanning classifier is built as a combination of a feature learning algorithm (Convolutional K-means), convolutional layer

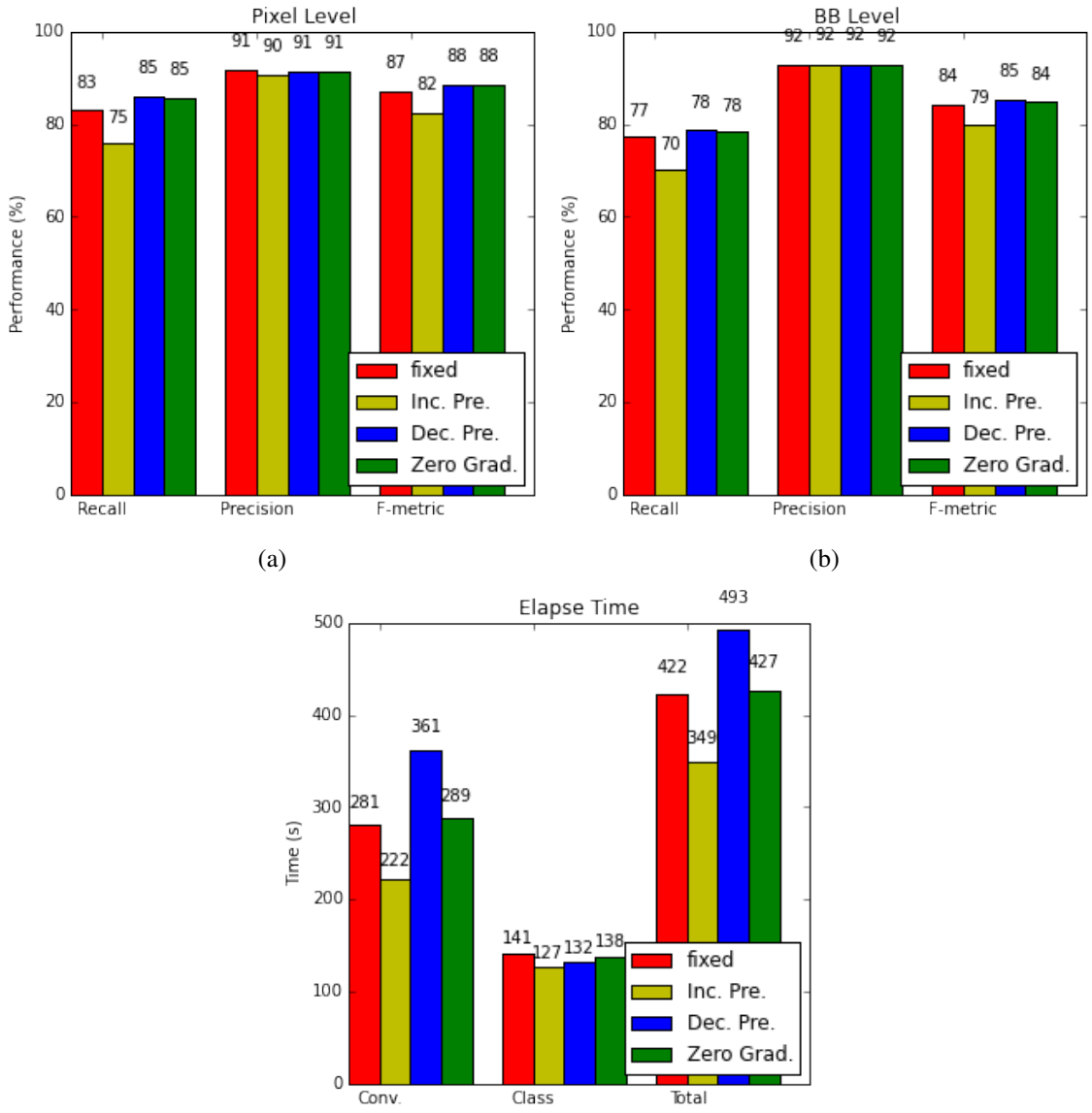


Figure 4.20: Comparison of cascaded classifier performance with different thresholding methods, including fixed threshold, increasing precision (Inc. Pre.), decreasing precision (Dec. Pre.) and zero gradient method (Zero grad.). The precision, recall and F-measure for (a) pixel and (b) BB level are evaluated. The (c) time elapsed for all cases are also compared.

and AdaBoost classifiers. The classification algorithm is built based on several reasons. First, the Convolutional K-means (CKmeans) feature learning algorithm is adequate to learn codebooks for text detection with simple computation. Second, a single layer convolution is sufficient to generate features using the CK-means learned codebooks. Third, AdaBoost algorithm is powerful enough to combine features generated by

convolutional layers and form a discriminative classifier from weak decision stumps.

The proposed system shares advantages that were provided by these basic algorithms. The features are learned from the data, preventing the use of strong prior knowledge and human intervention. The convolutional feature generation method allows weight sharing between different locations and reduces the number of parameters to be tuned during training. The AdaBoost combine different weak classifiers by re-weighting samples and weighing each decision stumps according to its performance and achieved the best performance when comparing to SVM and Random Forest.

The system is trained using ICDAR 2015 dataset. The image patches are firstly extracted from the image with 30 different scales, and foreground and background samples are equalized. 80% of samples are used as the training and 20% are used as the testing. The ICDAR 2015 testing images are then used for final evaluation. To train fine detector, extra samples that synthesized by Tao Wang are used to enhance fine classifier's performance.

We utilized a two-step structure to find text symbols with coarse and fine grids. The coarse detector uses contextual features to establish an initial estimation of text location predictions while the fine detector uses a grid search in different rotational angles and aspect ratios to find more accurate detections.

We tested our system in different situations. First, we compare the performance with and without contextual information, showing how contextual features help separate foreground and background samples more efficiently and with higher accuracy. Second, we compare the performance of different classification algorithms, showing that AdaBoost gets the best performance comparing to other popular methods. Third, we tested AdaBoost algorithm with different numbers of iterations. Finally, the fine detector is tested illustrating the necessity of implementing grid search with different rotational angles and aspect ratios.

To improve speed while preserving accuracy, cascaded architecture similar to VJ face detector is discussed. We tested different thresholding method for stage classifiers, including fixed threshold, increasing precision, decreasing precision and zero gradients. Although cascaded classifier decreases the overall accuracy, the zero-gradient based approach could balance speed and accuracy and provide an interesting alternative than standard Real-AdaBoost.

To answer our research question: *Can word detection accuracy be improved by combining convolutional k-means detection with subsequent processing?* We combined the feature learned from CKMeans with novel classification algorithm to address the problems in text detection and accommodate the relatively small size

of training data. The results are promising.

In the next stage, CKMeans based features are used along with CCs to further improve the overall performance. Spatial relationships are considered to classify edges between CCs as inner word and inter word relationships, and CCs are further pruned using CKmeans and spatial relationship based features.

Word bounding box level ground truth are used in training this classifier, where overlapping area is used as a criterion to label patches as foreground and background. However, in following CC leveled classification, pixel level ground truths are used along with BB level ground truth to improve the detection accuracy.

Chapter 5

Part Label Detector for Patent Drawings

Besides natural scene color images, other image based text detection could also be done by feature learning and convolutional patch-based methods. Document images can include but not limited to math equations, chemistry diagrams, musical notations and engineer drawings. They have unique characteristics and bring about very different challenges. It has a lot of practical interests for detecting text in patent drawing document images [110], which can help digitize and automate patent recording, sorting and retrieval. So one can for example display the text associated with part label overtop of a diagram images as it is being read online. The patent drawings are usually gray-scale images composed of lines, curves and strokes. In contrast, natural scenes are colored images composed by color pieces and blobs.

In this chapter, to answer the research question of our thesis: *Can we extend the convolutional k-means feature learning method for text detection in other types of images*, we discuss the text detection technique using feature learning and convolutional patch based method on patent drawing images, as an effort to extend the automatic text detection methods to different kinds of images. The proposed system achieves competitive performance, higher than the Top-1 player in USPTO competition participants. The harmonic mean of precision and recall (f-measure) is 82% for bounding box level detection and 73% for detection plus recognition.

5.1 USPTO Competition

United States Patent and Trademark Office (USPTO) stored an enormous amount of engineer drawing images describing the inventions graphically. USPTO needs to bring an archive of 8 million patents into digit documents and store in its information systems. Patent examiners, as well as patent lawyers and inventors throughout the world, rely on this archive to categorize, sort, locate and retrieve patents. This significant effort motivated research into automation of these process with both text [111] and images [112] information. By convention and by rule, these drawings contain only figures and numbered labels without text descriptions. The reader needs to scan the associated document to find the description of a given label.

The labels are composed of two categories: figure and part labels. The figure labels are usually under or on the right side of corresponding figures, while the part labels are usually embedded into the figures close to the corresponding parts, and often linked by a line or an arrow.

5.1.1 Data

A month-long online competition in which participants develop algorithms to detect both figures and part labels were hosted by USPTO on 'TopCoder.com' [1]. The challenge drew 232 teams, 70 of which (30%) submitted their solutions [113].

The patent drawings are provided in 300-dpi images scanned along with an HTML file containing corresponding text. This feature allows participants to utilize text information to enhance graphic recognition accuracy. There are totally 306 patent drawing images in competition 1, divided into 3 subsets, containing 178, 35, 93 images as training, provisional test, and system test sets, respectively. The training dataset was available for download to all participants, for designing and developing their own systems. The second dataset is used to evaluate their systems by making available privately to the participants only after a submission. The third evaluation dataset is used for the last submission of each team to produce the final comparison to prevent over-fitting.

Each patent drawing images contains one or more figures. Each figure has one title (figure label) and multiple part labels. In most cases, part labels affixed to their corresponding parts by lines and arrows. Part labels are composed of numerical and alphabet symbols. Besides figures, figure labels and part labels,

[1] http://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=15027&pm=11645

images may also contain tables, page titles, page numbers, patent numbers, inventor names, and dates.

Each patent drawing is accompanied by text in HTML format, containing descriptions. As by convention, figure labels and part labels need to be explicitly referred in these text at least once, HTML files provide additional information to validate and modify character recognition results.

For input, participant systems need to receive a 300-dpi grayscale patent document image in JPEG format and an HTML file containing the text of the patent. For output, systems need to identify figure locations and labels, along with part label locations and corresponding text. Locations are represented by bounding box coordinates. Figure and part label regions and texts are represented separately using text files called *answer* files. *Answer* files begin with the number of detected regions (figures or part labels) on the first line. Each remaining line defines a region, by a polygon represented using a list of vertex coordinates followed by the associated text.

Participant systems need to implement two separate functions to produce these outputs: the first for figure data and the second for part label data. The competition imposes one-minute time limit per page and 1024 MB memory limit. There was no explicit code size limit, but the size smaller than 1 MB was advised. The programming languages supported are Java, C++, C#, or Visual Basic .Net.

The offline tester/visualizer is provided, allowing participants to visualize their results and check the precise implementation of the scoring calculation. The examples of visualization tool generated images are shown in Figure 5.1.

5.1.2 Participant Systems

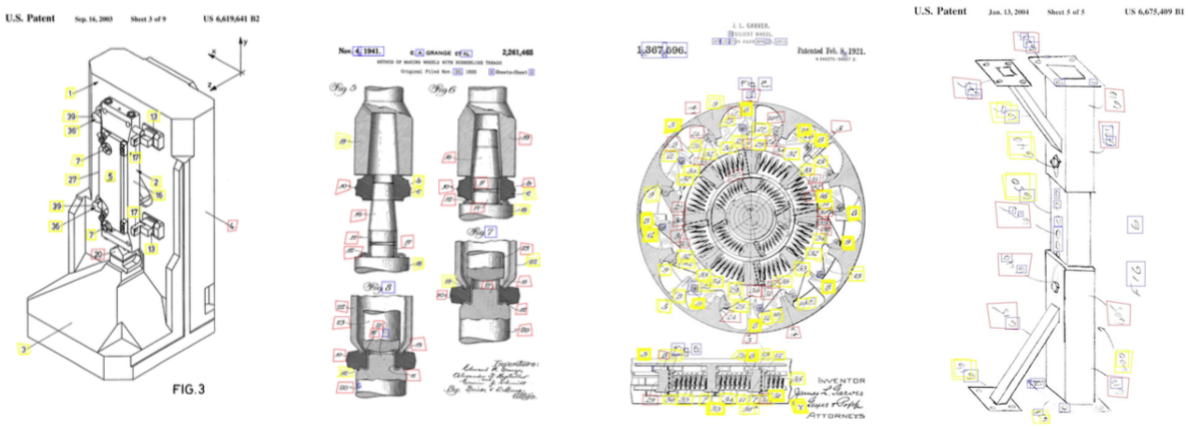
The top-5 systems adopted similar strategies but differed in approaches to text/graph separation, page orientation detection, region segmentation, OCR, validation, and use of HTML text.

All top-5 systems use a data-driven, bottom-up recognition architecture. Connected components are first extracted and then filtered using thresholds on sizes. Page headers and tables are then removed by filtering. Distances between connected components (CCs) are used to group candidates into clusters (labels). After this, pattern matching is applied to OCR results for words to locate figure labels and part labels. Most systems employ validation step to prune or correct recognition results.

Many parameters in these systems are set based on assumptions or ‘empirically’ by grid searching different values. Although, this causes sub-optimal performance, it is unavoidable considering the limited

competition time.

The top 1 player team in challenge 1 used a Connected Components (CC) based algorithm for text detection, measuring the shapes of CCs. When some metrics of a CC, like size, aspect ratio, boundary length, exceed prefixed thresholds, the CC is discarded. Template matching method is used to recognize detected characters. A language model is used to filter symbols. The length of a word is restricted to be equal to or less than 4 characters. Only character a, c, f, g and numerical symbols are legal in their lexicon model. Information from HTML files is used when recognition confidence is low.



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

Figure 5.1: Sample results from the first place system (leftmost two columns) and second place system (rightmost two columns) on the part label detection and text recognition task [113]. Part label regions are shown in yellow (hit), blue (false positive), and red (miss). The Top-1 system gets better results on image (a) than in image (b). The missing in (b) are caused by failure to recognize handwritten part labels; false alarms are caused by page titles. The Top-2 system gets better performance in (c) than in (d), as part labels are slanted handwritten characters which are more difficult to recognize and more prone to over-segmentation.

For part label detection, the first place system obtained a harmonic mean of recall and precision (f-measure) of 92% for bounding box detection and 81% for bounding box plus recognitions. For part labels, it achieved 81% for detection and 71% for detection plus recognition. The data and source code of top-5 systems are then made available by competition holder through UCI machine learning repository online under Apache License [2] . Examples of Top-1 and Top-2 participants part label detection are shown in Figure 5.1, where the hit bounding boxes are marked with yellow, false positives are marked with blue and missing are marked with red. The first two columns are from the Top-1 player and last two columns the are from the

[2] http://archive.ics.uci.edu/ml/datasets/USPTO+Algorithm+Challenge\%2C+run+by+NASA-Harvard+Tournament+Lab+and+TopCoder++++Problem\:+Pat

Top-2 player.

5.2 Methodology

Unlike competition participants, which usually used heuristic rule based methods to find location of the texts, we propose to use unsupervised learning method to generate adaptive and powerful features from the data [114]. The proposed system is composed of two major parts: Text detection and character recognition. Convolutional patch based method is used to classify patches as foreground and background for detection. The detection step is consisting of two parts: Unsupervised feature learning and AdaBoost classification, as in Fig. 5.2. For recognition, we apply Tesseract OCR onto detected region of interest (ROI). For detection system, it contains two components: Unsupervised feature learning and AdaBoost classification, as in Fig. 5.2

The proposed system focus only on part label detection and recognition, figure labels are not considered.

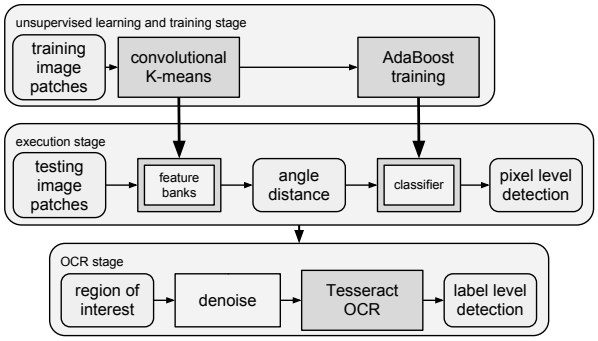


Figure 5.2: USPTO patent drawing part label detection and recognition system.

5.2.1 Detection

First of all, the training image and testing images are in different sizes. In order to match the image sizes, as well as the part label sizes, one need to reduce the testing images to $1/6$ of original.

The patches are extracted from the images in sliding window raster scan mode. Different from natural scene images, engineer drawing images contain major areas of plain white pixels. These regions contain no objects, so can be discarded to speed up the process. So the patches are used in training and testing only when they contain black pixels. The training patches are labeled based on the ground truth bounding box

coordinate information.

To address the characters with different scales, the image patches are produced with different sizes, with 5×5, 9×9, 15×15, and 19×19 covering small to large characters. Small image patch presents local structures while large patch presents relatively global structures in the image.

As for natural scenes, Convolutional K-means is used to learn features from the extract feature patches. The bank of quantized features we learned from the data is used to classify the samples into two different classes, foreground (part label) and background.

We use confidence-rated AdaBoost classifier. The weak hypothesis of our AdaBoost is based on a single threshold decision stump. The confidence of the classifier on each sample is the scalar distance from the current sample to the threshold [108]. Thus at each iteration of boosting, we find the single threshold which divides the training samples with the minimum weighted error. And we re-weight each sample based on the classification correctness and confidence rate. The miss-classified sample weight will increase in the next iteration. The final classifier is a weighted combination of classifiers in each iteration.

As seen from Chapter 3, while combining support vector seeded and randomly seeded convolutional k-means could help learn diverse/disjoint codebooks for enhanced classification, we may also implement this idea for document images and engineer drawings. Different from the previous experiment, where same datasets are used to learn different feature codebooks with different initialization methods, we implement the same learning process but to different datasets to learn differed codebooks.

There are three sets of feature banks unsupervised trained from different samples. One feature bank is trained using foreground samples only, another feature bank uses background samples, the third one uses both foreground and background. We can compare the performance of the classifier using different feature banks. Moreover, we could boost on features combined from all banks and achieve higher performance.

Fig. 5.3 shows two possible AdaBoost architecture alternatives using all three feature banks. In our experiment, we test both structures shown in Fig. 5.3, within each type of structure, we use image patch sizes from 5×5 to 19×19. Thus, from each case, we produced 4 different patch sizes by 128 number of clusters, totally 512 features.

In Fig. 5.3a, we concatenate all features from different feature banks and different patch sizes together to form a single feature matrix with each feature length equaling $512 \times 3 = 1536$. In Fig. 5.3b, we train three AdaBoost classifiers with different feature banks, then use a Meta-Boost to combine the results of the

three branches.

The Meta-Boost structure utilizes the classification results from each branch classifier. The branch classifier's output is actually the weighted sum of basic classifiers, at the final stage, the weighted sum is binarized to produce classification decision, positive or negative. However, if we omit binarization, AdaBoost can provide a real valued estimation of each sample, the sign is classification decision and magnitude is the fuzzy confidence rate of the classification. Utilizing this real number output allows designing another AdaBoost classifier using previous classification results as features and combining different classifiers' decisions to improve accuracy. Because AdaBoost classification results in branches are used as meta-data for the final classifier, so final classifier is called 'Meta-Boost'.

Meta-Boost will allow faster training in each branch classifier as they are shorter in feature length comparing to Type I ensemble and could be computed simultaneously. Instead of allowing classifier to freely choose best features in each iteration, Meta-Boost structure forced final classifier to include features from all three banks, including foreground, background and FG+BG bank. In our experiment, as foreground, background and FG+BG banks are describing different aspects of image patterns, classifiers could benefit from this forced inclusion. In AdaBoost, the confidence-rated weak hypothesis is used as in [108]. The AdaBoost runs to 256 iterations for each architecture in Fig. 5.3.

Patch based detection generate hotmaps which can be used to crop ROIs. To group the detection results into ROI, 20 by 20 neighborhood of each detected patch center (pixel) is considered. If 20 by 20 neighborhood of the patch centers intersect, then they are combined to form a larger region.

5.2.2 Denoising

Since the engineer drawing contains many objects that are visually similar to texts, it is hard to generate clean detection using convolutional feature based detector. Part labels are usually beside or embedded into figures and linked by arrows and curves, these neighboring objects are often included in the ROI and decrease the recognition accuracy. Furthermore, part labels often contain punctuations, like dots, brackets, and underscores, which need to be removed before recognition. De-noising becomes important for patent drawing images to be recognized with high accuracy. Several de-noising methods are used in our system to enhance detection results:

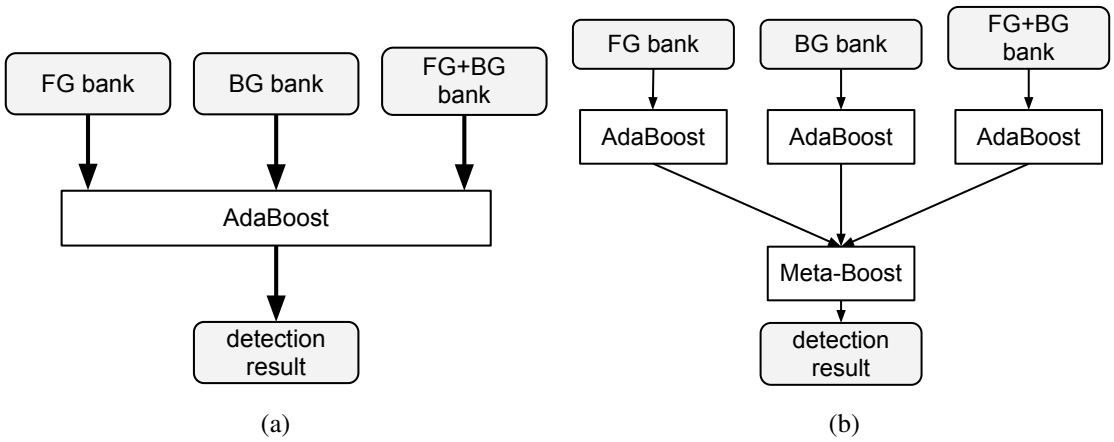


Figure 5.3: Two types of system architectures. (a) Type I uses all features in one single boost. (b) Type II boosts using different feature banks, and in a cascaded second stage, previous results with confidence ratings are used to generate a final decision. This method of using previous classification results as features are called Meta-Boost.

- A title remover is implemented to cut the page titles for each patent drawing image;
- The widths and heights of the detection ROIs are examined and used to filter large and small CCs;
- The total number of foreground pixels are counted for each ROI. Regions contain too few foreground pixels are discarded;
- Since true positives are usually at the center of ROIs, boundary objects usually contain neighboring figures and curves/arrows that link part labels to parts. A boundary object remover is implemented to remove these noise objects. Objects touching the boundaries of the ROIs are eliminated.
- To counter the case where one symbol is separated due to noise or detector's missing, we perform morphological close, which is a morphological dilation followed by erosion, capable of closing small gaps between CCs.
- An orientation correction component is added to convert all text lines into the horizontal direction.
- For ROIs contain multiple text lines, projection profile based method is used to cut the area into individual text lines;
- ROIs are slightly blurred with 3 by 3 uniform filter to remove deformation, holes, and noises.

As shown in Figure 5.1, the patent drawings contain page titles, which may cause false positive detections and reduce overall precision. A simple title removing is performed by cropping 300 pixels from the top of each page.

To filter the detected region based on their sizes, the widths and heights of the rectangle bounding boxes from detection are limited to be between 20 pixels and 200 pixels. Only bounding boxes within this range are kept while others are discarded as false positives. The total number of pixels within each CC are also restricted to be more than 40 pixels. CCs contain less than 40 foreground pixels are usually background objects, e.g. small dots, lines, and noises.

Because some patent drawings have part label written vertically with 90-degree counter-clockwise rotation, procedures are needed to rotate them back before recognition. The histograms of aspect ratios of detected ROIs are used for orientation correction. The aspect ratio of each object is defined as its height divided by its width. Horizontal characters usually have the aspect ratio greater than 1, while vertical characters usually have aspect ratios less than 1. The number of objects having an aspect ratio greater than 1 or less than 1 in each patent drawings is counted. If the voting suggests that ROIs are vertical, then all ROIs in this patent drawing are rotated by 90 degrees clockwise. Then the OCR is performed for each ROI.

The aspect ratio after rotation is then used to prune background symbols. Symbols like underscore, dash or other horizontal bars usually have a high aspect ratio. Therefore, CCs with aspect ratio larger than 2 are removed to reduce false detection caused by this kind of symbols.

As we select a Tesseract OCR mode that is only capable of parsing text characters in a single line, it performs poorly on multiple text line images. Therefore, we separate images beforehand using horizontal projection profile to divide a bounding box if multiple line text presents. The projection is computed along each horizontal line and a total number of foreground pixels are counted. The valley points of generated profile are then used to separate current bounding boxes into two or more bounding boxes. If there exists a peak point between the boundary and current valley point, and the difference between this peak and valley point is greater than 20 pixels, then the location of the valley point is marked as a cutting point. The coordinates of all valid cutting points are then used to divide the current ROI into several smaller regions.

5.2.3 Character Recognition

As patent labels are usually drawn using typeset characters, a typical OCR engine can recognize them with fairly high accuracy. Our system utilizes Tesseract [115] to recognize, meanwhile to correct the text location for higher accuracy utilizing the feedback of recognition confidence.

Tesseract is known as being capable of recognizing alphanumerical symbols with high accuracy on clearly printed text documents. In order to use Tesseract on patent drawings, we need to first localize the text position and crop the ROI to perform recognition.

We set page segment mode of Tesseract to treat each ROI as a single text line. This setting is important as other settings will produce much lower recognition results. A flag 8 is given for Tesseract here to indicate that candidate regions contains single line of text. The output of Tesseract consists of two parts, the recognized symbol class and the symbol location. Then symbol location from OCR output is used along with ROI position from detection to localize the part label more accurately.

The OCR output text labels are filtered. First, only English characters (a~zA~Z) and numerical symbols (0~9) are kept. As we are interested only in part labels, figure label candidates which contain 'Fig, FIG, Figure, FIGURE' are removed from candidates. Then alphabet symbol small 'o' and capital 'O' are replaced by a numerical symbol '0' to get better performance, since the later one is indistinguishable from the previous two but has a much higher frequency in documents. The number of alphanumerical symbols in recognition results are counted. Each part label should contain up to 4 symbols, of which there are up to 2 English characters and no less than 1 numerical symbols. Finally, if a part label contains only one symbol such as 'i', 'I', '1' or '0', then it is probably a noise object and removed from the final result.

To further increase the accuracy of the part label detection, one could utilize the associated HTML file to correct the recognition results when necessary. Since HTML files does not contain all part labels and not every number in those files are part labels, it is hard to use the HTML file directly to filter the detection and recognition results. Therefore, we proposed to utilize HTML file along with Tesseract OCR results on two different regions to produce enhanced information.

Tesseract, as powerful and accurate as it is, is dependent on the correctness of ROIs that it is applied to. Detection step may not crop the optimal region for recognition. Sometimes, the detected region is too small to contain whole characters, while in other cases, they are too large and contains background objects. In

order to enhance recognition accuracy, we apply the OCR on two kinds of candidate ROIs. The first one is the ROI contains candidate CCs dilated by 20 pixels. The second one is dilated by 40 pixels.

First, we compare the OCR results from the two kinds of regions as described above. All words from HTML are extracted and compared with label recognition results. If the OCR results are same, then no HTML information is used. If they are different, then Levenshtain distances are computed between the OCR results and HTML parsing results respectively. The word extracted from HTML file with minimum Levenshtain distance is then used as replacement for the OCR result.

5.2.4 Evaluation

The prediction area BC is a match to the ground truth area BG when the intersection between two areas is larger than a percentage α of the larger of the two areas:

$$\text{area}(BC \cap BG) \geq \alpha \max\left(\text{area}(BC), \text{area}(BG)\right) \tag{5.1}$$

where $\alpha = 0.8$ for figure matching, and 0.3 for part label matching. Different α values are used as part label sizes are much smaller than figures'.

To match recognition text labels, all letters are converted to lower cases. Only a-z, 0-9, (,), -, ', <, >, . (period), and / are kept for evaluation, other characters are trimmed. The trimmed prediction should be perfectly matched to the ground truth within the same bounding box in order to be considered as a hit.

However, there are some special cases, for example where two part labels may be indicated together, e.g., '102 (103)' or '102, 103' indicating parts 102 and 103. Periods/dots must be removed from the end of part labels. Subscripts are indicated using $<$ and $>$ (e.g., A $<$ 7 $>$ for A_7); superscripts are represented in-line (e.g., 123^b is represented by 123b).

The region area and label should be matched simultaneously for a part label to be considered correct matching. Partial credit is given for a correct detection with false recognition. The score for partial matching is 0.25 and full matching is 1. Recall, precision and f-measure are computed over all the bounding boxes and labels.

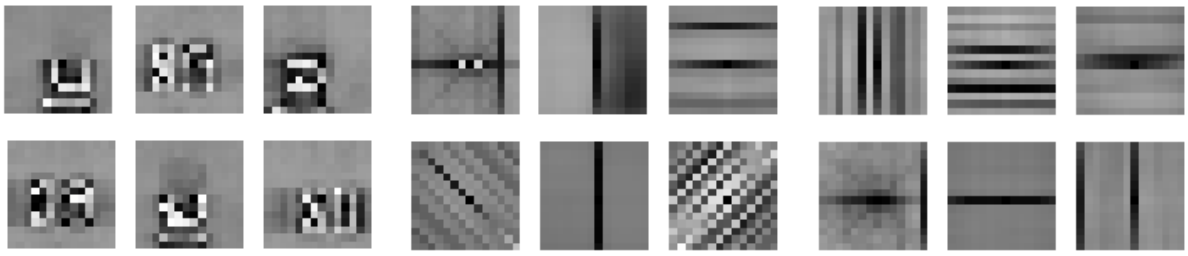
$$\text{Recall} = \frac{\sum \text{match score}}{|\text{Target Regions}|} \tag{5.2}$$

$$\text{Recall} = \frac{\sum \text{match score}}{|\text{Target Regions}|} \tag{5.3}$$

$$\text{F-measure} = \frac{2\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.4}$$

5.3 Results and Discussion

We learned the quantized feature banks from FG, BG and FG+BG images patches. From Fig. 5.4 we could see that feature banks using BG and FG+BG are similar, meanwhile, feature banks using only foreground differs more greatly from the other two. The background bank usually catches the horizontal, vertical and diagonal line patterns. The foreground bank catches the patterns where objects are laying in the center and homogeneous regions on the boundaries. This is easy to be imagined, as background commonly contains figures with lines and curves, and label texts are isolated symbols.



(a) (b) (c)

Figure 5.4: Quantized feature pattern bank using unsupervised learning algorithm from different sample pools. (a) Feature codebook examples learned from foreground samples; (b) Feature codebooks learned from background samples; (c) Feature codebooks learned from both sample sets. Using different features, one can address different aspect characteristics from training sample, and the learned features are less correlated.

For MetaBoost classification, one can check the weight distribution among three different branches to evaluate the significance of each feature learning scheme, as shown in Figure 5.5. The FG+BG branch gets the heaviest weight among three different feature learning methods, indicating that FG+BG feature based AdaBoost classifier contribute most to the final predictions. FG branch is weighted second and BG branch is weighted third. It is interesting that although BG branch has visually similar patches as in FG+BG, it is weighted much lighter. Because these two branches are similar but with different performance, during

training, MetaBoost classifier may more frequently choose from FG+BG since it has better accuracy. On the other hand, as FG branch learned more distinguished feature codebooks, it could compensate the FG+BG branch in more cases, therefore, weighted higher than BG branch.

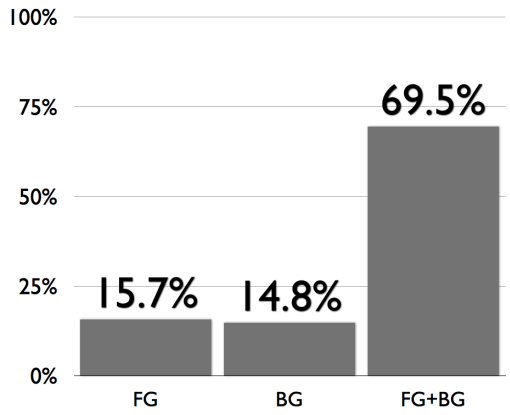
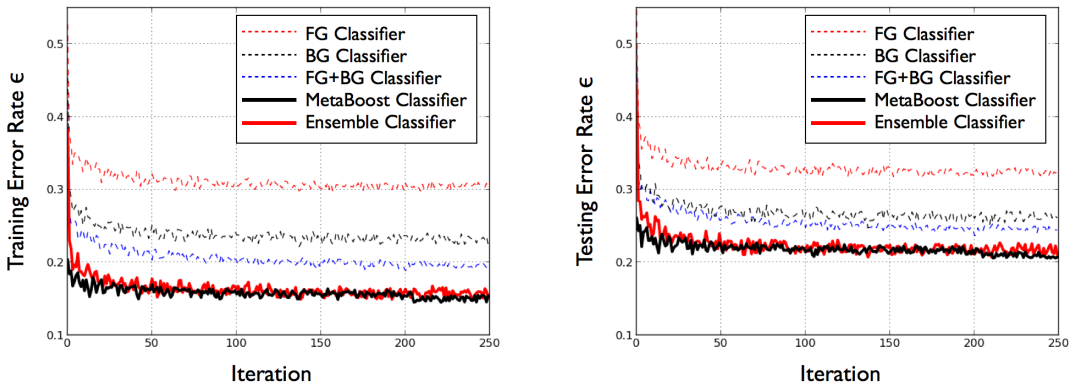


Figure 5.5: Meta-Boost weight distribution. The MetaBoost classifier used three branch classifiers' real valued predictions as input features, and the weight distribution of each branch features are bar plotted. The FG+BG branch is weighted maximum among the three with 69.5% after normalization. The FG branch is weighted second with 15.7% and BG branch is weighted third with 14.8%. Weight distribution indicates the preference of meta-classifier, which dependent mostly on FG+BG features.

The performance of ensemble method (Type I) and MetaBoost method (Type II) and each branch classifier is shown in Figure 5.6 for comparison. The error rate in each iteration, is plotted for both training and testing. Fig. 5.6 shows the unweighted error rate variation with iterations for training samples. The FG branch produced worst performance with final error rate 30.5% on training, while the FG+BG branch has best error rate 18.7% on training for among three branch classifiers. For testing data, error is converged to a higher value. However, for both training and testing, MetaBoost achieves better results than ensemble classifier. On pixel level, the error rate for ensemble classifier is 9.1% for training and 19.0% for testing; the error rate for MetaBoost classifier is 8.9% for training and 18.7% for testing.

The performance of text detectors are also evaluated using precision, recall, and F-metric, as listed in Table 5.1 and Figure 5.7. All precision, recall and F-metric is evaluated in pixel-level. We count the number of true positive pixels, a total number of detected pixels and a total number of foreground pixels to compute precisions and recalls. The AdaBoost classifier using only FG+BG data for training gets 76.04% f-measure, which is much higher than other branch classifiers using FG or BG. The Meta-Boost classifier achieves slightly better precision (65.9%) and F-measure (77.85%) comparing to Ensemble AdaBoost for testing



(a) (b)

Figure 5.6: Error rate through iterations for (a) training and (b) testing using different AdaBoost classifiers. Comparison has been made among classifiers using only foreground (FG) features, background (BG) features, FG+BG features. The classifiers using different architectures as shown in Figure 5.3 are also evaluated and plotted. The one uses Meta-Boost classification (Type II) methods achieved better accuracy after the same number of iterations of training.

samples.

Table 5.1: A comparison of our system using different training data and architectures, evaluated using pixel-level precision, recall and F-measure. Combining features from all cases increase the accuracy. The best recall is produced by ensemble method (Type I), where all features are concatenated and used in one shot; the best precision and F-measure is produced by Meta-Boost classifier, where each group of features are used to train individual classifiers and classification results are used as second stage features to train another classifier.

	Recall	Precision	F-measure
FG	85.3	35.4	50.03
BG	92.9	60.1	72.98
FG+BG	94.1	63.8	76.04
Ensemble	**95.2**	65.7	77.75
MetaBoost	95.1	**65.9**	**77.85**

The ROI detected by patch based detector is illustrated in Figure 5.8. As shown in the figure, there are still background objects need to be removed before OCR. These objects include lines or arrows connecting the part to part labels, dots, underscores, etc. Our denoising procedure used simple heuristic rules to eliminate those objects. The final output is the center objects which are usually part label characters as shown in Figure 5.9.

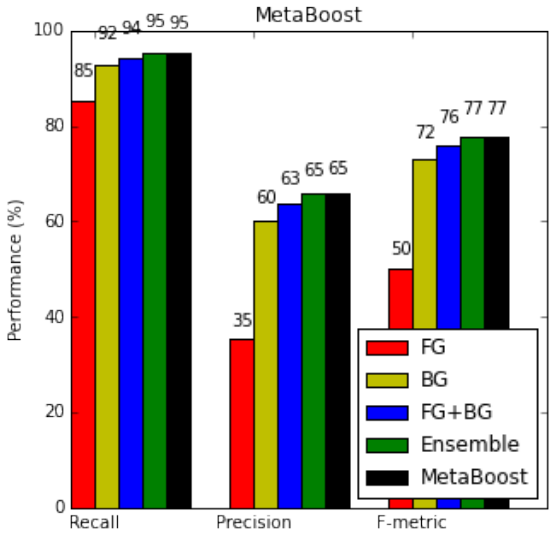


Figure 5.7: Recall, precision and f-measure of AdaBoost classifier using different training data and combination methods. The classifiers' performances using single dataset are shown, including FG, BG and FG+BG. FG+BG dataset achieves highest performance among the three. Accuracy could be further improved by combining different dataset, using ensemble method or Meta-Boost method. The later one achieves most accurate performance among all 5 cases.

For OCR, Tesseract performs poorly if using the CCs directly as inputs. We enlarge the detected CCs by 20 and 40 pixels respectively using zero padding (adding 20 or 40 pixels with zero values around the image patch), then use them as input for OCR engine to produce recognition labels.

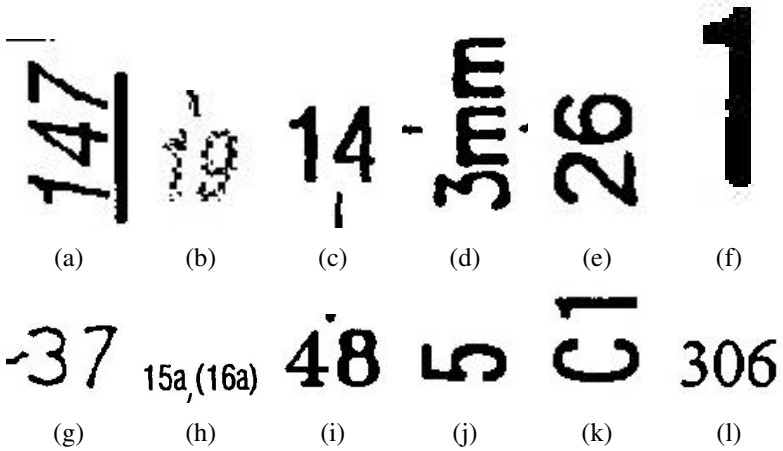


Figure 5.8: Example of patches extracted using MetaBoost classifier. The patches are usually larger than target characters and contain additional objects, like lines (a)(c)(d)(g)(h), arrows, underscores (a), and dots. These objects need to be removed before recognition using denoising techniques. The targets also contain holes and other noises (b) which can be suppressed by uniform blurring.

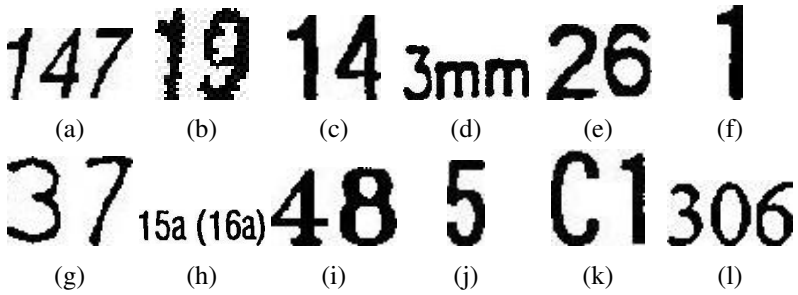
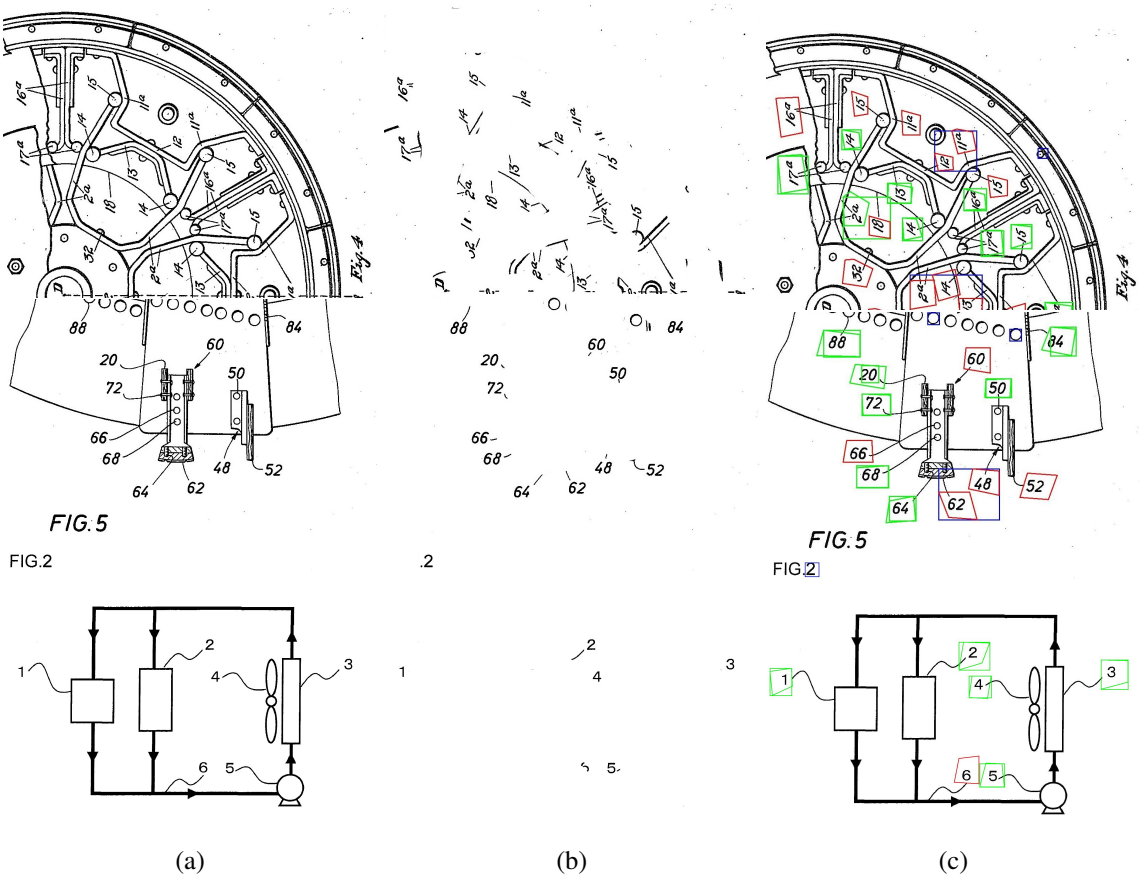


Figure 5.9: The cropped CCs after removing boundary objects and noises. Boundary objects (lines and curves) are removed (a)(b)(c)(d)(e)(g)(h)(i). Morphological closing is performed to remove holes (b). Orientations are corrected by checking aspect ratios of bounding boxes (a)(d)(e)(j)(k). CCs aspect ratio is checked to remove horizontal bars (a). The cropped CC will then be dilated with 20 and 40 pixels in each direction with zero padding for recognition.

The pixel level detection map examples are illustrated in Fig. 5.10b. The texts are extracted with very high recall value. Detection false positives are showing as noise in the image, most of them can be easily removed by OCR. The examples of OCR errors are illustrated in Fig. 5.10c. The bounding boxes detected are drawn overtop the original testing images. Correct detections are marked as green, while false positives and missing are marked as blue and red respectively. As in Figure 5.10 (c), part labels that are partially recognized and under-segmented are assumed as errors and marked as red. This evaluation method could be improved in the future.

More examples of final detection evaluated using visualizer provided USPTO competition are shown in Figure 5.11. In Figure 5.11(a), a false positive is marked because it's recognized as number 8; in Figure 5.11(e), the red bounding box indicates a part label miss-recognized as single number 3. According to examples, the part label recognition plays a vital role in selecting the correct candidates. Due to the limited information and simple lexicon model, the falsely recognized or missed part labels may reduce the overall performance of the detection.

Additional examples of final detection are shown in Figure 5.12. In Figure 5.12(a), the part labels are recognized incorrectly. For example, part label '22' are recognized as 'zz' and discarded; part label '198' and '196' are segmented into a single word and discarded due to exceeding length limit. False positives such as label 'Fig. 6', where 'F' is missing by the initial detection, is kept. The recognized label is 'ig. 6' and passed the lexicon model filtering which supposes to remove figure labels. In Figure 5.12(b), the part labels are all detected correctly. The label '12' is missing (red) because of the differences in bounding box sizes



(a) (b) (c)

Figure 5.10: Some example of detected region from USPTO engineer drawing images. (a) Original image; (b) text detection map and (c) Tesseract OCR output. The text detection stage removes background regions and allows OCR performed only on the ROI. The correct recognition and detection is marked by green rectangles, while incorrect are marked by red. There are two polygons around each part label, one of which represents ground truth and the other one represents prediction.

between prediction and ground truth. In Figure 5.12(d), the part labels are missed by the detector as they are embedded closely between lines. In Figure 5.12(d), many part-labels failed to be recognized due to the handwritten font. In Figure 5.12 (e), the part label is missed due to the lexicon model we utilized. The part label which contains more than 4 characters is discarded (‘220s1’) and which contains no numerical number (‘H’) is also removed by lexicon filtering.

The final performance evaluation is generated in label-level using tools provided by competition. Performance of detection and recognition are evaluated individually by computing precision and recall with perfect OCR or perfect detection, as shown in first two rows of Table 5.2. If detection results are perfect, our OCR engine produces 83.72% precision, 83.72% recall and 83.72% f-measure on the overall label level perfor-

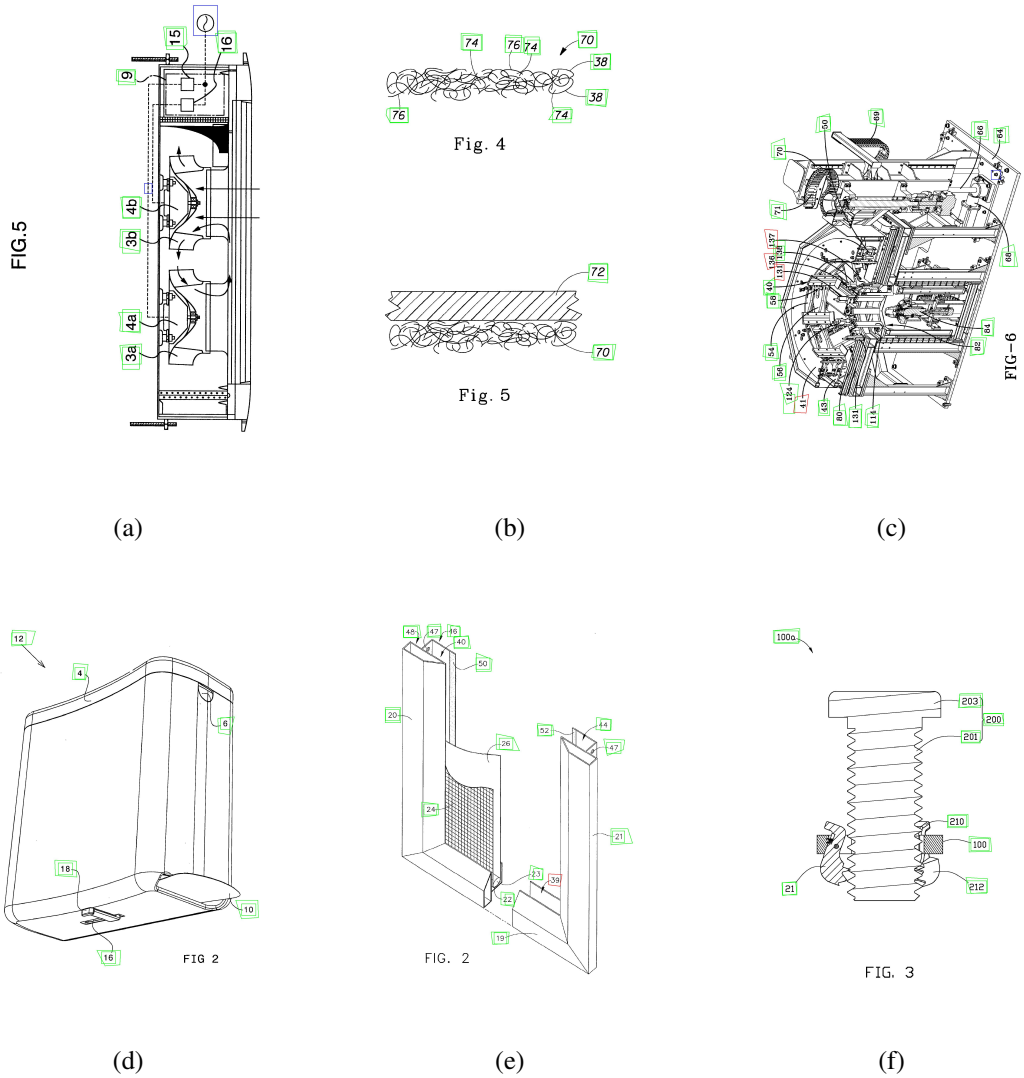


Figure 5.11: Example of bounding box detection and character label recognition results. The green bounding box indicates that the part label detector has located the position of part labels correctly while also recognize the symbols perfectly. Blue bounding boxes indicates false positives and red indicates missing. This result illustration is generated using tools provided by USPTO competition holder. Most of the detection are correct, except in (a) one false positive marked by blue bounding box; in (c)(e) recognition error.

mance. If OCR is set perfect, our detection system produces 78.26% precision, 96.12% recall and 86.28% f-measure. The proposed detector extracts most true positives regions with many false positives (high recall, low precision). The recognition step trims false positives using lexicon models to improve overall precision. Denoising processes are important. The proposed system achieves 70.10% precision, 69.33% recall initially [114] and improved to 71.91% precision and 73.55% recall after modifying the denoising process

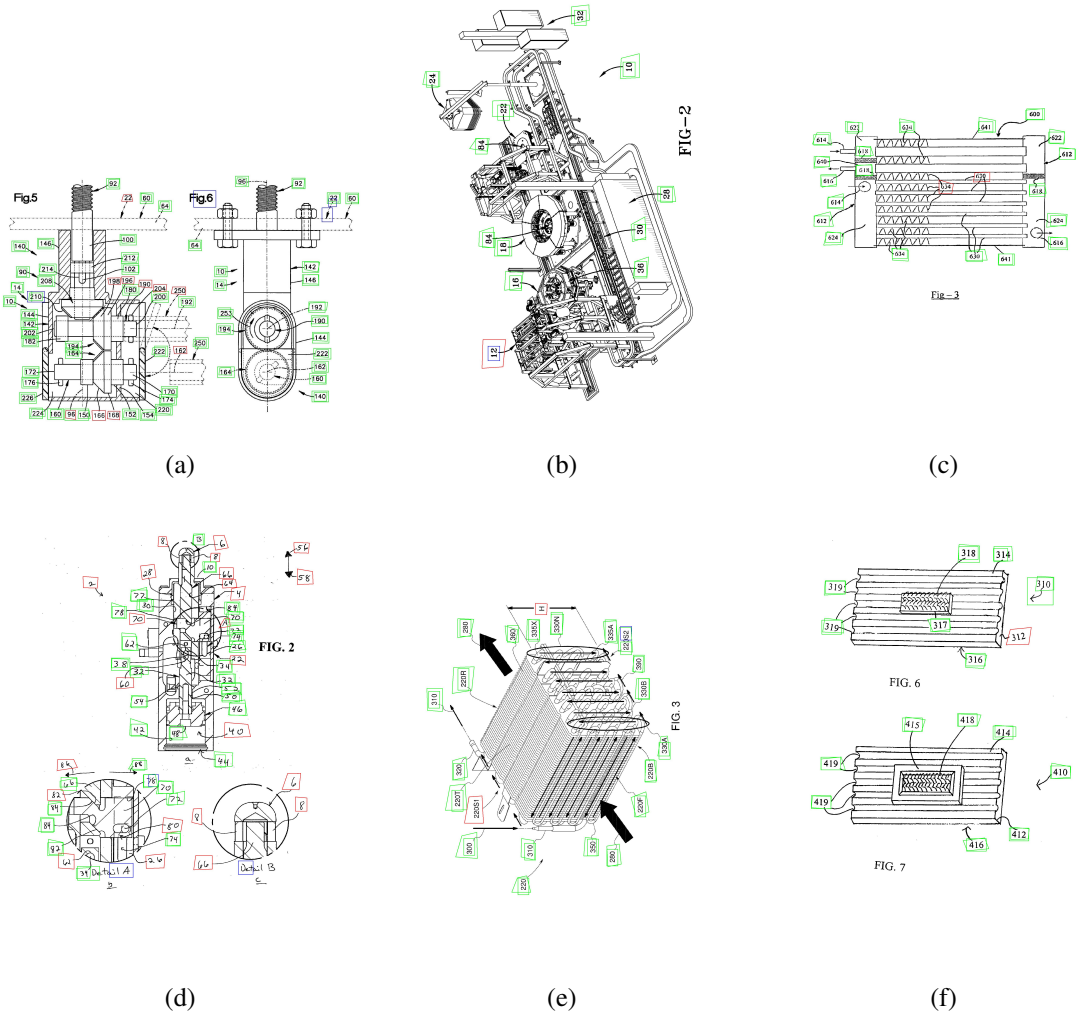


Figure 5.12: Example of bounding box detection and character label recognition results with errors, (a) correctly detection was incorrectly recognized; (b) correct recognition but bounding box is larger than ground truth; (c) false recognition due to occlusion; (d) false recognition due to hand-written characters; (e) false recognition due to strict lexicon model (part label is too long or contains too few numerical characters); (f) recognition error caused by missing parts.

and lexicon models, as listed in Table 5.2 and Figure 5.13.

The error in USPTO label detection is caused by many reasons related to characteristic of engineer drawing images. Handwritten fonts in part labels are difficult to be recognized, so brings about missing regions. On the other hand, symbols and characters in figure labels, tables and other associated texts can introduce false positive regions. The lexicon model used in our system is simple. Although it could increase the overall performance by eliminating a large amount of false positive regions, it sometimes falsely remove

true positive regions as in examples shown in Figure 5.12(e).

Table 5.2: A comparison of different systems, evaluated using label-level precision, recall and f-measure. Our system achieved 71.91% precision and 73.55% recall on USPTO dataset which is higher than Top-1 system. The effects of each stage is also evaluated by making the other step performance perfect (replaced by ground truth). If OCR performance is perfect, one could achieve higher performance, which indicates that OCR has introduced more error than detection step.

	Recall (%)	Precision (%)	f-measure (%)
Perfect Detection + Real OCR	83.72	83.72	83.72
Real Detection + Perfect OCR	96.12	78.26	86.28
ICDAR	69.33	70.10	69.71
Proposed system	**73.55**	**71.91**	**72.72**

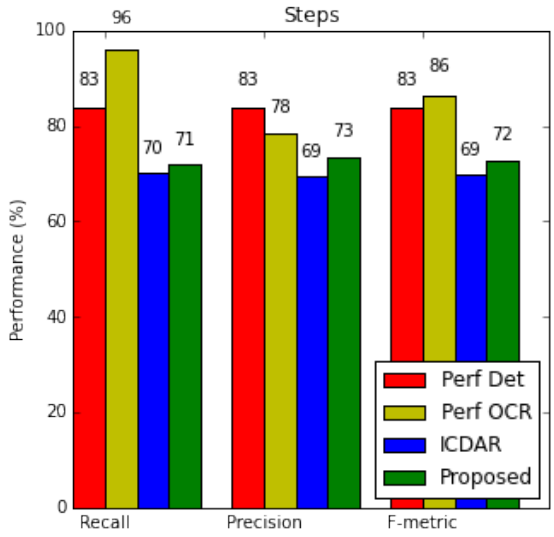


Figure 5.13: Comparison of effects to performance of each step. The system consist patch based detection, Tesseract OCR, and denoising systems using shape and lexicon information. By making detection perfect, the overall performance increases to 83.72% f-measure; by making OCR perfect, the overall performance increases to 86.28%. This indicates that there is more room for OCR to be improved rather than for detector. The performance of ICDAR system [114] and current system are compared. The performance is improved by utilizing denoising and lexicon models.

By combining the convolutional feature based detector, denoising unit, recognizer and lexicon model, we achieved performance higher than Top-1 participant of USPTO competition, as shown in Table 5.3 and Figure 5.14. The performances of Top-5 competition participants are compared with our system in both Bounding box level (BB) and label level (BB+label).

The label level evaluation is performed as described in Section 5.2.4. Both bounding box coordinates

and OCR labels are used to compare with ground truth information. For part label bounding boxes that matches and contains correct texts, they are counted as 100% matching; for part label bounding boxes that matches but contains incorrect texts, they are counted as 25% matching. The overall performance is then computed using this metric, along with total number of prediction regions and total number of ground truth regions. Precision, recall and f-measure values are computed and compared.

The BB level evaluation uses a simplified metric dependent only on positional matching but no recognition labels. It measures the accuracy of a part label detector in terms of finding the correct part label positions. To generate BB level performance, we modify the 'BB+label' level prediction so that all OCR labels are made correct for detected regions. Therefore, the 25% penalty is eliminated and only positional information can affect the performance.

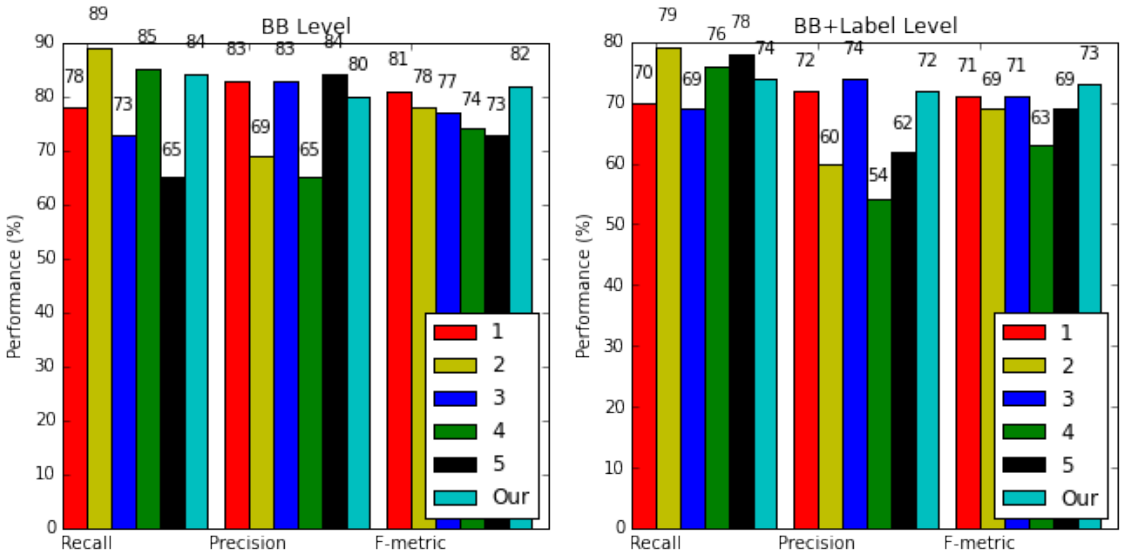
The 5th participant achieved best precision on BB level, and 2nd participant achieved best recall. Our system achieves best f-measure which is a harmonic mean of the precision and recall. For 'BB+label? level, 5th got the best precision and 2nd got the best recall. Our system gets the best f-measure again, which reflects that BB level performance and 'BB+label? level performance are highly related. .

Table 5.3: The part label location (BB) and recognition (BB + label) results evaluation. This evaluation is compared with Top-5 of Top-Coder participants. We have achieved higher performance than the highest performance in competition. System 1 to 5 indicates the participant systems with ranking from 1 to 5. The performance is not necessarily ranked from high to low, as it only contains part label detection results, no figure and figure label detection results are shown in the table.

	BB Level(%)			BB + label (%)		
System	Recall (%)	Precision (%)	f-measure (%)	Recall (%)	Precision (%)	f-measure (%)
1	78	83	81	70	72	71
2	**89**	69	78	**79**	60	69
3	73	83	77	69	74	71
4	85	65	74	76	54	63
5	65	**84**	73	62	**78**	69
Proposed System	84	80	**82**	72	74	**73**

The engineer drawing images are difficult for patch based convolutional feature detector. The engineer drawing images contains less visual cues than natural scene images and requires more denoising, preprocessing and post-processing to get an acceptable performance.

- The USPTO engineer drawing images contain figures that are drawn by lines and curves, which visu-



(a) (b)

Figure 5.14: Performance comparison between Top-5 USPTO competition systems and our system in both (a) bounding box level and (b) bounding box + label level.

ally are very similar to text and characters.

- The images contain a large amount of texts, including page titles, figure labels, associated description texts, texts in tables etc. To detect a particular type of text (part labels), lexicon model and heuristic rules are inevitable.

- Engineer drawing images have much fewer color features to be used than natural scene images.

Due to the reasons mentioned above, many denoising processes and heuristic rules on sizes of patches and lexicon models are used besides patch based detector. However, engineer drawings contain part labels with fewer scale variations than natural scene texts, which means less searching in different scales is needed for the detector. The engineer drawing contains large areas of white space, which require no searching. Therefore, engineer drawings can be scanned faster than natural scene images.

5.4 Summary

A part label text detection and recognition system for patent drawing image is proposed. Our system learns features from data directly, using Convolutional K-means. An AdaBoost classifier is trained to automatically

classify pixels as foreground or background. A Meta-Boost algorithm is proposed to shorten training while increasing accuracy. Denoising process, 'Tesseract' OCR and lexicon model are used to recognize symbols and refine symbol locations. Our system produces better performance than Top-1 competition participant for a difficult dataset.

Comparing to natural scenes, patent drawings have many different characteristics. They are grayscale or binary, containing lines and curves very similar to texts. The object of digitization of documents requires one to find specified types of texts (figure labels or part labels), therefore, one need to eliminate many other text regions, like page titles, tables etc. Therefore, utilizing visual features alone is insufficient for an accurate part label detector. Denoising process, OCR and lexicon model are necessary.

For learning and training patch based text detector, the process is very similar for natural scene and diagram images (patent drawings). Because patent drawing contains part labels with smaller scale variation, less scales can be used. Since large areas of patent drawings are white space, they can be ignored during patch scanning for both training and testing.

To enhance the performance of detection, different methods are used to increase the diversity of the learned feature banks. For natural scenes, we implement convolutional k-means with different seeding technique, first with randomly sampling and then support vectors. For engineer drawing, we use a different approach to increase the diversity of features, by learning features with different datasets. The features are learned with foreground, background, and foreground+background patches. Therefore, three groups of different features are learned to focus on different groups of training samples. To combine different features together, a novel method is proposed called Meta-Boost. Instead of concatenating all features from different groups together like we did for natural scene images, we train a detector for each group of features using AdaBoost. Then another classifier is trained using branch classifiers' prediction values as features. Meta-Boost architecture shows an improvement over simple concatenation for patent drawing images.

The experiment done for engineer drawings supports our hypothesis that patch based convolutional feature learning can be used for text detection in different types of images, including diagram images. Although our detector is powerful, OCR and denoising components are required to get high performance, because a specified type of text needs to be detected . It also supports that increasing in diversity of learned feature codebooks could help improve accuracy of detection.

Part of the research done for patent drawings was published on International Conference of Document

Analysis and Recognition, 2012 [114]. The USPTO competition was published on International Journal of Document Analysis and Recognition, 2016 [113], where our system was used as a state-of-art benchmark for competition of part label detection.

Chapter 6

Character and Word Detection

The patch based feature detector uses features learned by convolutional K-means to produce prediction for locations of text. The experiments have been done for both natural scene and patent drawings. For natural scenes, hotmaps are produced to indicate the likelihood of each patch being text. For patent drawings, the hotmap is used to crop the region of interest (ROI) from the image and Tesseract OCR is applied. The OCR helped eliminate false positive regions and produce recognition labels.

For the natural scenes, we would like to compare our results with ICDAR 2015 competition participants in task 2 (Text Localization), where bounding box coordinates for word locations are required but no recognition is necessary. In fact, as multiple fonts, handwritings, calligraphies and commercial logos are included in the ICDAR 2015 focused scene dataset, it is very hard to recognize the text with high accuracy. Therefore, the text recognition is another research question that beyond the scope of this dissertation.

On the other hand, to produce accurate bounding box coordinates from natural scene images is also challenging. The patch based detection hotmap could be used as a based for bounding box extractions, but the direct extraction accuracy is low due to under-segmentation of nearby words. Therefore, CC grouping and segmentation methods are required to estimate character locations as CCs with homogenous color, grown from the center of matched patch regions, and then clustering these CCs into estimated word locations.

This chapter tries to answer the research question: *Are visual features along sufficient for high-accuracy text detection in natural scenes?* Algorithms are designed to convert patch based detection results, which is generated using visual features alone, into word bounding boxes. The validation and segmentation classifiers are also solely based on visual features.

In this chapter, we introduce methods to estimate the locations of characters from our convolutional k-means based detection hotmaps, and then introduce the Word-Graph algorithm for defining locations for Word bounding boxes. We compare our results with state-of-art systems, and our recall (81.02%), precision (93.39%) and f-measure (86.77) all exceed the Top-1 competitor of ICDAR 2015.

6.1 Methodology

In this thesis, we applied region growing algorithm utilizing the fine detection results as seeds to produce pixel level accuracy from the patch level hotmaps. Second, the pixel level text detection results are grouped into connected components (CCs) and graph model based segmentation algorithm is applied.

The final results are bounding boxes of words. The system produces text files containing the x, y values of the upper left corners of the bounding boxes and widths (w) and heights(h) of the bounding boxes, in the format of x, y, w, h for each line corresponding to each bounding box.

6.1.1 Region Growing

The thresholded fine detection saliency map in Chapter 4 provides seeds for a flood-filling type of region growing to form CCs. For each position that was classified as text from Text-Conv, we start to grow regions iteratively until they reach an edge or a large shift in color.

Some small false CCs might appear, due to small homogeneous regions around character edges. We implement a surrounding suppression technique to remove these easy negative regions. When a text region is detected, its surrounding area is suppressed. The surrounding area is defined as 5 pixels in our experiments, a CC is discarded if it is fully covered by another CC with 5 pixel dilation.

Region growing will add new pixels into foreground regions iteratively by considering two criteria:

1. The edge intensity along the growing direction is small;

2. The color difference between a newly added pixel and a region seed pixel is small.

If $q = (i_q, j_q)$ is a pixel immediately adjacent to the contour of CC Q (while not in Q), and $p = (i_p, j_p)$ a pixel in Q, then the growing direction is defined as $\Theta = atan2(i_q - i_p, j_q - j_p)$. If multiple pixels

$p_1, p_2, ..., p_n \in Q$ in D are adjacent to q, the growing direction is defined as the average of all directions for $p_1, p_2, ..., p_n \in Q$.

Edge Image. Edge images are computed from the intensity gradient in each color channel, in the horizontal and vertical directions. For three color channels R, G, B, the color gradient map of the image can be computed by the Laplacian Matrix L:

$$L = D^T D, \text{where } D = \begin{bmatrix} \frac{\partial R}{\partial x} & \frac{\partial R}{\partial y} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \\ \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \tag{6.1}$$

The gradient amplitude can be computed using the largest eigenvalue of the Laplacian matrix λ, and gradient direction can be computed using the eigenvector corresponding to λ. Intensities of the gradient, $\frac{\partial R}{\partial x}$ etc. are computed discretely using Sobel kernels. The angle between the growing direction and gradient direction is computed by $\delta\Theta = \Theta_e - \Theta_g$, where Θ_e is the gradient (edge) direction and Θ_g is the growing direction. Notice that $\delta\Theta$ is regularized, therefore its range is between $(-\pi/2, \pi/2)$.

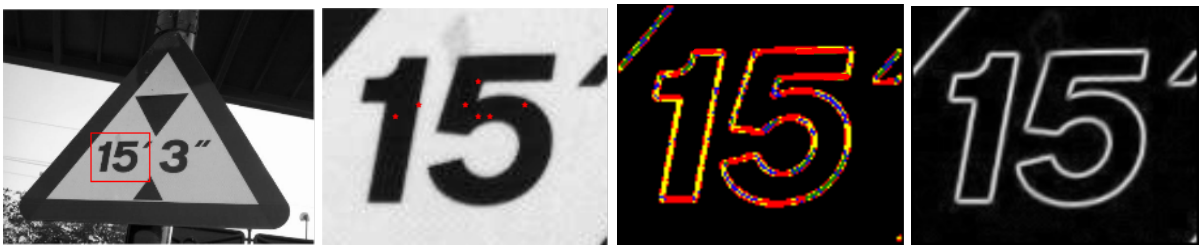
Cost Function. The region growing criterion C is defined as following equation:

$$C = cos(\delta\Theta)\ \lambda + \frac{\sum_{c \in R,G,B}(|I_{c,q} - I_{c,seed}|)}{Z} \tag{6.2}$$

where the first term represents the edge intensity along the growing direction, and the second term represents the color difference between boundary pixel q and the region's seed pixel. Z is a normalization factor, for datatype of 8 bit integers, the value of Z is 256. Regions grow from seeds iteratively, adding valid boundary pixels into the foreground region based on C.

Initially, seed pixels and region boundaries are labeled as foreground/background respectively. Unlabeled pixels with minimal cost are labeled iteratively. Region growing stops when no unlabeled pixels exist between foreground and background.

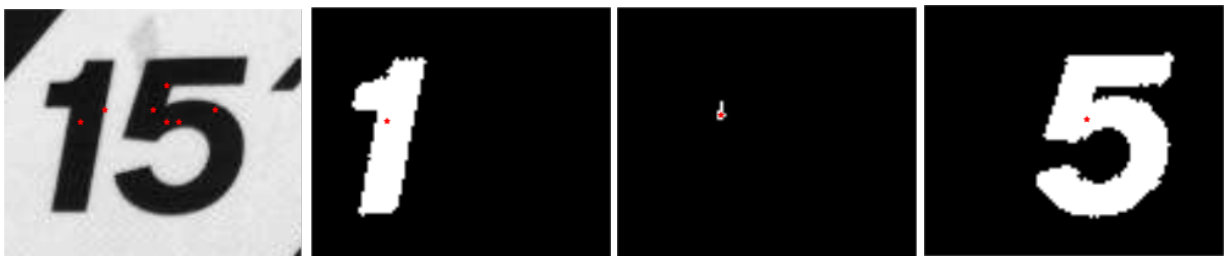
The cost function is a combination of edge values and color differences. An example of the region of interest produced by coarse detection and seeds produced by fine detection is shown in Figure 6.1. The original image is shown in Figure 6.1 (a) while one of its multiple regions of interest is marked by the red rectangle. The seeds produced by fine detection within the marked region of interest is shown in Figure



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

Figure 6.1: Example of detected region of interest from ICDAR image, fine detection produce seed pixels for the region growing algorithm. The intensity and orientation of edges are utilized to compute the cost function for each pixel and to grow foreground regions. (a) Original image with region of interest marked by red rectangle; (b) the seed points inside the region of interest; (c) the orientation of edges, horizontal edges are marked by red, vertical edges are marked by yellow, diagonal edges are marked by blue and green respectively; (d) The intensity of the edges.

6.1 (b). We also illustrate the orientation (c) and intensity (d) of edges of the current region of interest. To compute cost function, the first term in Equation 6.2 used both the intensity λ and orientation $\delta\theta$, while the second term used color differences.



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

Figure 6.2: Example of region growing to form CCs. (a) Region of interest with seeds marked by red stars; (b) region growing algorithm applied using the first seed; (c) region growing applied using the second seed; (d) region growing applied using the third seed. All other seeds are discarded, as they are included by the CC formed by the 3rd seed. The 1st and 3rd seed form regions containing characters, while the second seed forms regions containing false positive. The false positive can be discarded by validation process.

The region growing is applied to each seed, as shown in Figure 6.2. The seeds inside the characters could generate CCs of characters while seeds outside the character produce false positives. Those false positives could be discarded by validation classifier. The seeds which locate within previous seed grown regions are discarded to save time because they tend to form similar regions as previous ones.

Validation. It is hard to extract all character candidates using a single set of parameters and thresholds

because wild scene image text has colors and edges with large variations. In our system, different sets of parameters are selected and then CCs are filtered with a validation classifier.

After growing candidate CCs for characters, an AdaBoost classifier is trained to prune CCs that are invalid. To accommodate the high variation in colors and intensities in natural scenes, using the input image we generate multiple edge maps with multiple Gaussian smoothing kernels with sizes as 3, 5, 7, 9, 11 pixels, with variance equal to half the kernel size. Edge point thresholds are defined as 50%, 60%, 70%, 80% and 90% of the maximum gradient value. By combining different parameters, we over-generate samples and prune character hypotheses using the AdaBoost classifier as described in Section 4.1.3.

We train the verification classifier using pixel-level ground truth (which is provided in the ICDAR data set). We count overlapping pixels between generated CCs and true characters, and use CCs with overlapping area higher than 80% as foreground.

The confidence-weighted AdaBoost detector as used in Text-Conv is applied. Precision and recall values can be tuned by choosing different cut-off thresholds. As we need to keep as many true positives as possible for later processing, we select the threshold so that recall is higher than 95% (similar tuning is found in the Viola-Jones face detector [105]). In a cascaded system, candidates are eliminated stage by stage. To keep the overall recall value within a reasonable range, recall in each stage should be kept high. However, precision in each stage may be relatively lower. As more and more false positive samples are removed through cascaded stages, the final precision can be much higher.

The AdaBoost classifier used convolutional patch features similar as in window scanning stage. The codebooks are also borrowed from sliding windows stage. However, instead of generating image patches by scanning windows throughout images, convolution could be computed once for each CC in a fixed location. Therefore, the computation time is much reduced comparing to the previous stage. All candidate CCs are resized to 32 by 32 pixels while keeping the original aspect ratio of characters. Edge detection and spatial pooling are also performed. With 1000 codebooks and 3 by 3 spatial pooling, 9000 features are generated for each candidate CC. The features are feed into Confidence-Rated AdaBoost classifier.

The region growing method convert patch level detection map into pixel level. The foreground pixels are grouped into CCs. Validation classifier removes false positive CCs and increases precision. The formed CCs along with graph model based segmentation can be used to generate bounding box level predictions of words as discussed in the following section.

6.1.2 Word Graph

Characters within a word usually have similar color and size, with relatively small and equally distributed distances. For English, natural scene text usually appears in horizontal text lines with a small rotation, with some exceptions. For example, isolated characters may form a word (e.g. 'a'), and text lines might be curved. Objects with regular textures like fences and bricks (as in Figure 2.4) share similar patterns with text, making them difficult to be removed using spatial relationship information alone. And so, to reliably merge characters into words, one needs to group CCs based on both their appearance and spatial relationships.

A Word-Graph word segmentation model is proposed, which is a combination of a graph model and random forest-based classifier for word forming. The Word-Graph algorithm groups detected character CCs and then uses context to prune false positive CCs. First, it uses a graph model $G(V, E)$, where characters are modeled as vertices V and their relations as edges E.

Two Random Forest classifiers are used in Word-Graph; the first for character merge/split classification, and the second for filtering invalid characters after forming words. Instead of designing features and predefined thresholds for linking characters as in [116], Word-Graph is data-driven, with little human intervention.

Features. Word-Graph utilizes shape and bounding box features as shown in Table 6.1, most of which are used widely for CC analysis, like width, height, distance, and orientation. The features are also normalized with different methods (minimum, maximum, average). Totally 29 features are generated. 7 features are normalized in four different ways, including minimum, maximum, average and no normalization. Therefore, totally $7 \times 4 + 1 = 29$ features are generated. For example, minimum normalization of center distance in x-direction is given as

$$\frac{x_{c,i} - x_{c,j}}{min(width_i, width_j)} \tag{6.3}$$

The total 29 features are listed in Table 6.1, where different distances and difference values are normalized with different methods. The combinations used by our feature generation method are marked by checkmarks, while others are marked by 'X'. Along with the 29 bounding box features, we also used 900 patch level features to enhance the segmentation performance. 100 features learned from convolutional K-means are convolved with difference image, spatial pooling is performed with 3×3 grids. Totally 929 features are used for segmentation.

Minimum Spanning Tree. One challenge for word segmentation is that two distant characters may

Table 6.1: Examples of character segmentation features. x_p and x_q are the center coordinates along x axis for bounding box p and q respectively; y_p and y_q are the center coordinates along y axis for bounding box p and q respectively. $D(i,j)$ is the Euclidean distance between pixel i and j. W_p and W_q are the widths of bounding box p and q respectively. H_p and H_q are the height of the bounding box p and q respectively. O_p and O_q are the angles of main axes of objects p and q respectively. $I_{c,p}$ and $I_{c,q}$ are average intensity value in color channel c for object p and q respectively. Different normalization methods are used. For example, minimum normalization of center distance in x direction is given as $\frac{|x_p - x_q|}{min(W_p, W_q)}$.

Feature	Definition	Normalization			
		Max.	Min.	Ave.	None
center distance x	$\|x_p - x_q\|$	✓	✓	✓	✓
center distance y	$\|y_p - y_q\|$	✓	✓	✓	✓
center distance Euclidean	$\sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$	✓	✓	✓	✓
closest boundary distance	$min(D(i,j) : \forall i \in p, \forall j \in q)$	✓	✓	✓	✓
width difference	$\|W_p - W_q\|$	✓	✓	✓	✓
height difference	$\|H_p - H_q\|$	✓	✓	✓	✓
orientation difference	$\|O_p - O_q\|$	✓	✓	✓	✓
color difference	$\sum_{c \in R,G,B}(\|I_{c,p} - I_{c,q}\|)$	x	x	x	✓

belong to the same word, or two close characters may belong to separate words. These cases increase the variations of the training and testing samples and reduce the efficiency of the classification significantly.

One approach to tackling this problem is to classify relationships of neighboring characters only. In the training data, a minimum spanning tree (MST) based on spatial distance is used to select training edges [117]. This ensures that only edges between neighboring characters are used to define positive (merge) and negative (split) examples, increasing the separability of the classes (vs. using all pairs of characters) and also the speed of training. During testing, distant characters' relationship could be derived through transitive linkage over 'merged' edges to form words.

MST trimming reduces the number of training samples, therefore, increase training speed. More important, it makes reduce the training and testing feature variation and make the classification more efficient as shown by the experiment in Table 6.2.

Random Forest Classifier. To classify edges between character as within word and between word, random forest classifier is used. Random forest is an ensemble learning method. It constructs multiple decision trees at training time and outputting the average prediction of individual trees. Random decision forest corrects for decision trees' habit of overfitting to their training set by averaging. Breiman and Cutler [118] developed the algorithm of random forest, which combining bagging and the random selection of

features.

The training of random forest is a general technique of bagging. For training set $X = x_1, x_2..., x_n$ and label $Y = y_1, y_2, ..., y_n$, one need to select a random subset with replacement from the training set repeatedly and train decision tree classifiers: f_b for each sample drawing X_b, Y_b. The final prediction for sample x_i is made by averaging or taking the majority of the prediction from all individual decision trees.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x') \tag{6.4}$$

This bootstrapping procedure leads to better performance as it decreases the variance of the model (by using averaging or taking the majority of multiple trees) without increasing bias (since all trees are trained with randomly selected samples). While a single tree is highly sensitive to noise and overfitted, the average of many trees is not. Training decision trees on the same dataset produce strongly corrected trees which defeat the purpose, bootstrapping sampling de-correlate the trees by showing the different training sets.

Besides sample bagging, random forest introduces another dimension of randomness to decrease variance which is called feature bagging. For each tree splitting node, a random subset of features is used and then maximum splitting metrics (e. g. Gini impurity, information gain) is computed. This random selection in feature space avoids correlation of decision trees if one feature is so strong that all trees select it.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split, therefore, features are randomly drawn from feature space at each node. For regression problems, the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default.

In the experiment, two parameters will affect the performance of the classifier. One is the number of features used for each node splitting, which is usually set to $\sqrt{p}$. Another is the maximum tree depth for each decision tree, deeper trees increase accuracy but decrease stability.

We choose to use the random forest for classification of Word-Graph model, mainly because it shows higher accuracy than AdaBoost or SVM. In the experiment, AdaBoost, SVM, and Random Forest are tested and compared. To balance the foreground and background samples, sub-sampling and over-sampling techniques (SMOTE) are tested.

Second Stage Character CC Validation. Besides segmentation, Word-Graph further improves detection precision with second stage validation. We combine features for individual character's information

along with spatial relationships between characters, providing additional cues to help remove isolated false positives. For this purpose, another Random Forest classifier is trained to remove hard negative samples at the final stage.

Like the first stage classifier, the patch-based features are used. Meanwhile, to enhance the performance, the edge classification results are used as features in second validation. A validation classifier will evaluate features addressing both the shape and spatial relationship of each character. For an isolated object, if its appearance is very close to a text character, it will be classified as foreground. For a not-so-like character object, if it is linked with other high-confidence characters, it could also be classified as foreground.

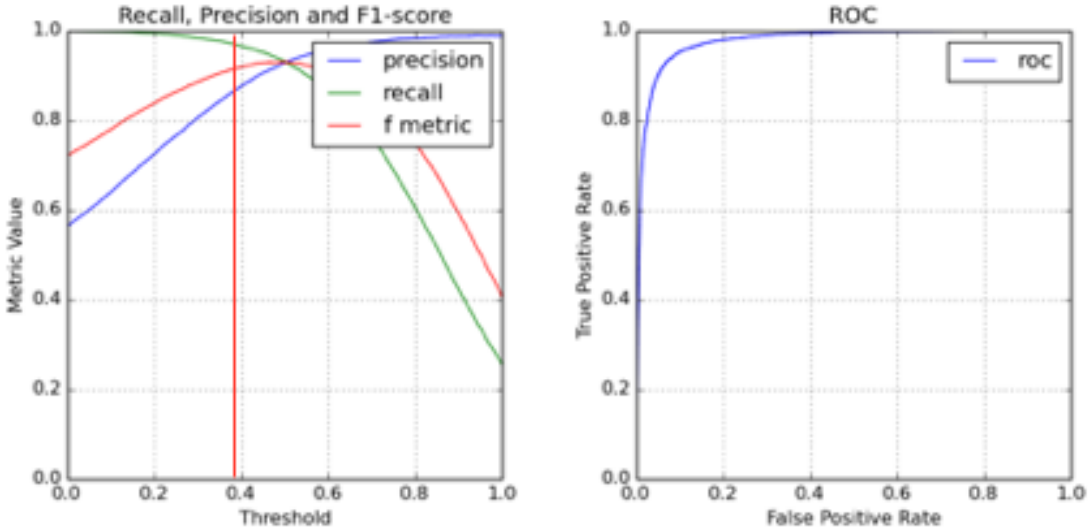
Unlike Meta-Boost used for document image text detection in Chapter 5, the classification results from the spatial relationship RF is not used directly. But the classification results help select the edge features used for the second stage of CC validation. As Random Forest, like Real AdaBoost could provide real numbers as the confidence of classification, the confidence rate is used as features for object pruning. The feature from the strongest link (highest confidence rated by Word-Graph), weakest link, the average of MST and average of all links. Same features as in Word-Graph segmentation are used for second validation. Along with convolutional features (900 features generated on 3×3 spatial pooling grid using 100 codebooks newly learned by convolutional K-means), it prunes false positives to further increase overall performance. Totally, $900 + 29 \times 4 = 1016$ features are utilized in second stage validation.

6.2 Results and Discussion

In our experiment, ICDAR 2015 data are used for training and testing as in Chapter 3 and Chapter 4. The fine detector is used to generate seeds for region growing algorithm. The region growing is first performed to form pixel level detection map and then group them into CCs. The bounding boxes of CCs are then extracted and feed to a graph model based Random Forest classifier, Word-Graph. The final output of the text detection system is the word bounding box coordinates constructed by CCs and their relationship classification results. We evaluate our system in the pixel, character bounding box, and word bounding box levels. The word bounding box level is compared with ICDAR competition participants and achieved better than the state-of-art performance.

6.2.1 Region Growing

Coarse detection produces predictions of text locations in different scales. Fine detection refines the prediction of position using smaller step size, the center locations of foreground patches are marked to be seeds. Those seeds are used to perform region growing process. The region growing starts from those seeds and include new pixels by checking cost function iteratively.



(a) (b)

Figure 6.3: The 1st stage verification classifier performance using AdaBoost. (a) Precision, recall and f-measure are plotted along with threshold (vertical red line); (b) ROC curve of the verification. The threshold is selected, so the recall value is high. This selection of threshold produces no optimal f-measure for current stage, but benefit the overall performance. As more false positive samples can be eliminated by second stage classifier.

To train the validation classifier, 4721 foreground and 7835 background samples are used. To tune the final threshold for the confidence-rated classifier, a grid search is performed to maximize the overall detection rate and then the threshold is set to the point of 96% recall and 88% precision (see Figure 6.3). This configuration produces the highest f-measure for the overall system for word detection on the training dataset although not the highest f-measure for this stage of classification.

Some examples of region growing results before and after verification are shown in Figure 6.4. The original images are shown in Figure 6.4(a), while the candidate CCs formed by region growing algorithms are shown in Figure 6.4(b). The foreground text, holes inside text and small regions between text are

(a) (b) (c)

Figure 6.4: Examples of region growing and validation to form CCs. (a) The original image; (b) The detected CCs after patch based coarse and fine detection, and region growing with multiple parameters are used to generate candidate CCs; (c) The validation AdaBoost classifier is used to trim background CCs from the image and the final pixel level detection map is formed.

common formed regions. Different regions formed from different seeds are marked by different colors. Validation classifier is then used to trim these candidates and form a cleaner pixel map of text, as shown in Figure 6.4(c).

For ICDAR dataset, pixel level detection results could not be compared directly with participant systems. The CCs need to be grouped into words and bounding box coordinates need to be extracted to get the

bounding box evaluation. Therefore, Word-Graph model is proposed to automatically group CCs into words utilizing random forest classification on edges between CCs. Furthermore, the character classification results could be used to further improve the detection accuracy.

6.2.2 Word Graph

After CCs are formed, the bounding box of each CC is extracted. The edges between each CC are used to classify the relationship between each pair of CCs as the inner-word or inter-word.

First, the MST is performed to trim the edges used for training. The Euclidean center distances are used to find the MST edges, restricting the training edges to those between neighboring CCs. This process reduces the training sample numbers and also decrease and variance of sample feature values. Totally 41179 edge samples are generated from ICDAR 2015 training set, where 5697 inter-word and 35482 inner word edges are generated. The edges are divided into 80% of training and 20% of testing by random selection. 929 features are extracted for each edge to train segmentation classifiers.

The edges used in Word-Graph are illustrated in Figure 6.5, where bounding boxes of characters and words are plotted along with edges between character pairs. The yellow lines represent edges between non-neighboring characters that are trimmed by MST during training. Red lines represent edges between characters belonging to different words. Blue lines represent edges between characters within the same word. In training, only the edges represented by red lines and blue lines are used. In testing, all edges are classified, including those edges marked by yellow lines. Distant characters relationship could be derived through transitive linkage over 'inner-word' edges to form words, even their direct relationship is classified as 'inter-word'.

The effects of using MST for training sample reduction is shown in Table 6.2 and Figure 6.8. The MST trimming is very important, as the overall system performance degrades significantly without using it.

Second, samples for inner word and inter-word edges are balanced. To train the Random Forest classifier for edges, a challenge is that the foreground and background training samples are highly unbalanced. As non-neighboring edges are removed from the training set using MST, within-word edge edges are far fewer than between-word edges.

To deal with the imbalanced samples, we compare different balancing methods, including using the original distribution without balancing, over-sampling to increase between word samples (using SMOTE

Figure 6.5: Segmentation using Word-Graph. The bounding box of characters are marked with green rectangles; the bounding box of words are marked with blue rectangles. Each character is a node in the graph model. Edges after MST trimming are used as training samples. The edges trimmed by MST are marked as yellow lines. Inter words edges are marked as red lines. The inner word edges are marked with blue lines.

Table 6.2: Comparison of different cases for Word-Graph model using word bounding box level evaluation. The effect of utilizing MST trimming the training edges are shown by evaluate the overall system in word bounding box level. Without MST trimming, the classifier performance degrades significantly on testing data. Effectiveness of second stage validation is also shown. The overall system achieves the best performance by including MST trimming and both stages of validations.

Method	recall(%)	precision(%)	f-measure(%)
w/o MST	65.45	73.21	69.11
w/o 2nd Validation	76.22	92.13	83.42
Complete	81.02	93.39	86.77

[119]) and random subsampling to decrease within word samples. While the first method uses unbalanced

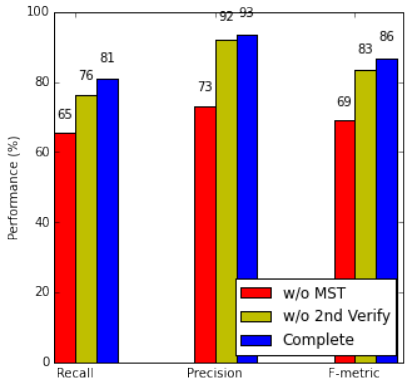


Figure 6.6: Effects of MST based sample trimming and the second stage validations using word bounding box level evaluation. If the training samples are not reduced by MST, the overall performance is degraded to 69.11% f-measure. If the second stage classifier is not implemented, the overall f-measure is reduced to 83.42%. The complete system achieves overall f-measure of 86.77%.

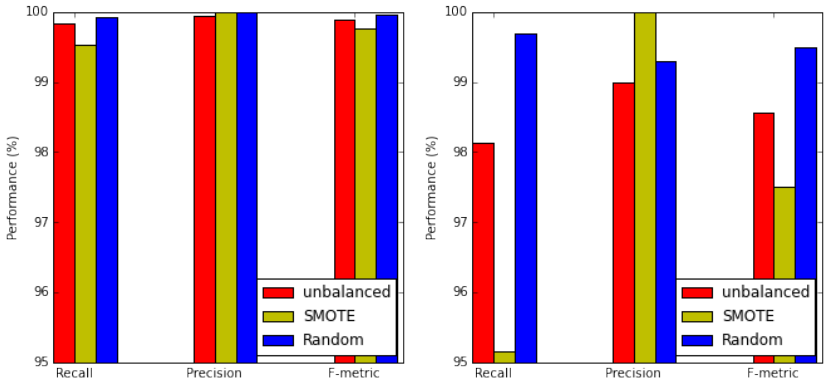
Table 6.3: Precision, recall and f-measure of random forest classifier for word segmentation using different sample balancing methods. Training and testing sample performance are listed with highest performance written in bold font.

	Training(%)			Testing(%)		
Method	recall	precision	f-measure	recall	precision	f-measure
Unbalanced	99.84	99.94	99.89	98.14	99.00	98.57
SMOTE	99.53	**100.00**	99.76	95.15	**100.00**	97.51
Subsampling	**99.92**	**100.00**	**99.96**	**99.70**	99.29	**99.49**

samples, the latter two methods both equalize the samples so that inner word and inter-word edges have the same number. The performance of using different sample balancing methods are shown in Table 6.3.

Experiments show that balancing samples are very important for reliable segmentation and character verification. We found that subsampling could produce better accuracy for testing samples than over-sampling. We were able to train an accurate segmenter using just 4557 inner-word and 4557 inter-word samples (80% of total 5697 inter-word edges).

We tested different classification methods to compare the performances. AdaBoost classifier with 1000 iterations, SVM classifier with linear kernel and C equals 1 and Random forest classifier is compared. Random forest classifier contains 100 trees and a maximum depth of 10. Each node split considers $\sqrt{929} \approx 30$ features. The performance comparison is shown in Table 6.4. The random forest classifier gets the best performance among all three classification methods, so it is selected as our segmentation method.

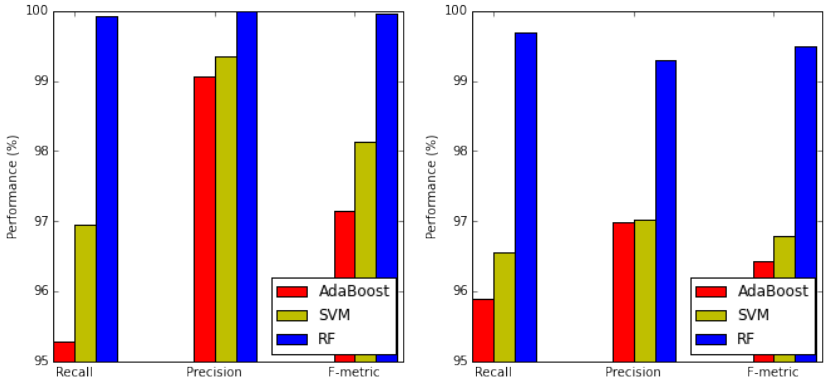


(a) (b)

Figure 6.7: The performance on (a) training and (b) testing edge samples when equalize training samples with different methods, including no equalization, SMOTE oversampling and random selection subsampling. Random selection shows best performance among all three cases. Y axis shows from 95% to 100%.

Table 6.4: Precision, recall and f-measure of different classification methods, including AdaBoost, SVM and Random Forest (RF). The random forest classification method gets best performance for both training and testing samples.

	Training(%)			Testing(%)		
Method	recall	precision	f-measure	recall	precision	f-measure
AdaBoost	95.29	99.07	97.15	95.89	96.98	96.43
SVM	96.95	99.35	98.13	96.56	97.03	96.79
RF	**99.92**	**100.00**	**99.96**	**99.70**	**99.29**	**99.49**



(a) (b)

Figure 6.8: The performance on (a) training and (b) testing edge samples using different segmentation classification algorithms, including AdaBoost, SVM and Random Forest. Y axis shows from 95% to 100%.

The second validation classifier is used to further increase the performance of text detection accuracy.

Different from the first validation classifier using convolutional patch features, the second stage validation classifier adds spatial relationships to enhance performance. The segmentation classification result contains the information of relationships between current character and characters around it. The strongest edge (with highest within word prediction), the weakest edge (with the highest inter-word prediction), the average values of all edges and average of MST edges are used in second validation classifier.

Characters from ground truth are used to define complete and MST graphs over characters and these ideal graphs are then used in training the random forest. The random forest classifier hyperparameters are set similar to the segmentation classifier, with 100 trees and a maximum depth of 10. Each node split considers $\sqrt{1016} \approx 32$ features. The performance comparison for different cases with or without second validation is shown in Table 6.2 and Figure 6.8. The second stage classifier could help the overall system f-measure increase from 83.42% to 86.77%, exceed the Top-1 participant of ICDAR 2015 competition on text detection task.

6.2.3 Overall Performance

In this chapter, starting with the seeds produced by fine detection, the region growing method converts the initial patch-level character detection map to a pixel-level character detection map. Two stage validations remove false positive CCs and increase overall precision. The word graph model produces character linkage predictions and group character into words. Character bounding boxes along with segmentation predictions are used to form word bounding box coordinates and saved into text files.

The final performance could be evaluated at different levels, including the pixel, character bounding box, and word bounding box levels.

Pixel Level. Some examples of pixel level predictions are shown in Figure 6.9. The pixel level performance is evaluated using precision, recall and f-measure considering only the overall number of intersection pixels between prediction and ground truth, overall number of prediction pixels and overall ground truth pixels, as shown in equations below:

$$\text{Precision} = \frac{P \cap G}{P} \tag{6.5}$$

$$\text{Recall} = \frac{P \cap G}{G} \tag{6.6}$$

$$\text{f-measure} = \frac{2\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.7}$$

where P indicates prediction pixels produced by our text detection system, and G indicates ground truth pixels. The performance is evaluated using all 233 test images of focused scene text in ICDAR 2015 competition.

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

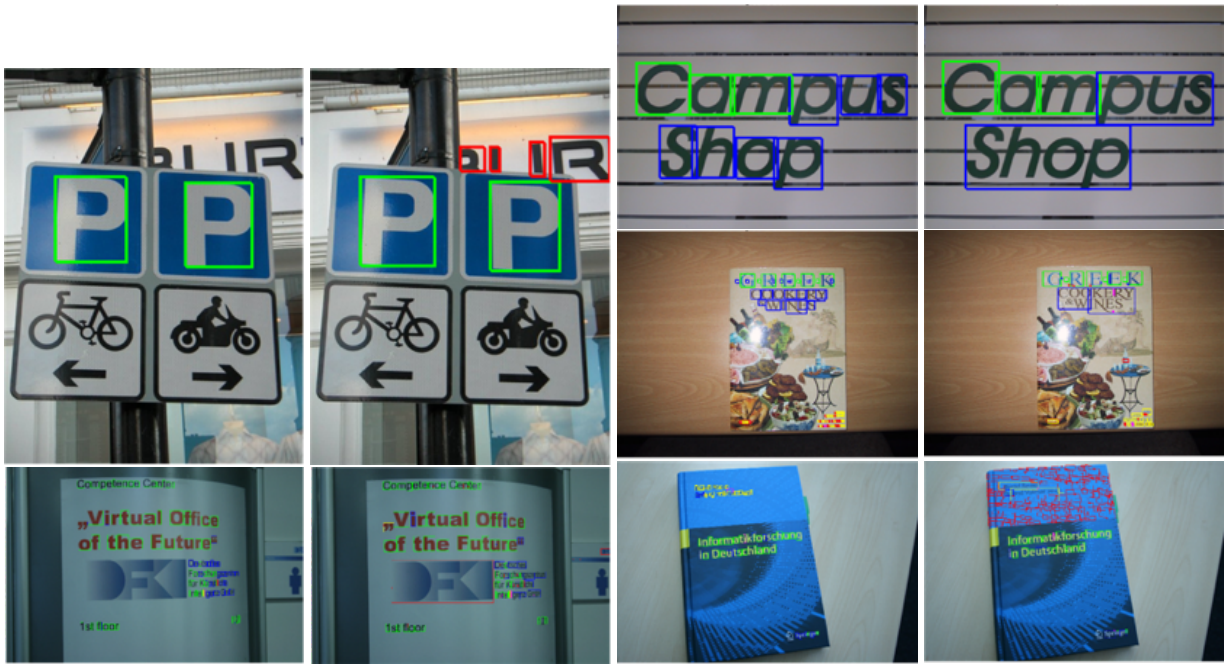
Figure 6.9: Examples of pixel level detection of the ICDAR 2015 dataset. The pixel level detection map is generated from flood based region growing process and cleaned by verification classifier. The pixel level map is binary, where each pixel has a corresponding indicator represent it as foreground as background. The pixel level detection is evaluated using pixel level ground truth provided by ICDAR and precision, recall and f-measures are computed. (a)(c) The original images; (b)(d) Pixel level detection map.

Character BB Level. Individual CCs are character candidates, extracting their bounding boxes and comparing with ground truth character bounding box coordinates can produce character level performance measurement.

We used the same criterion as in ICDAR 2015 for word bounding boxes to evaluation the performance in character level. The bounding box prediction should have at least 80% recall and 50% precision to be recognized as a hit. One-to-multi and multi-to-one matching are also allowed, but with a 0.8 deduction towards overall counting.

In some cases, multiple characters are connected due to poor segmentation, uneven lighting or noise. This causes under-segmentations. On the other hand, over-segmentation also happens especially for characters that were composed by multiple CCs. We didn't design a segmentation algorithm especially for characters, as we are more interested in word bounding box accuracy and its segmentation problem is address by Word-Graph. If a dedicated character segmentation method is implemented, higher character bounding box

level accuracy is expected. Some examples of character bounding box level detection results are shown in Figure 6.10.



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

Figure 6.10: Example of character bounding box evaluation results. The character bounding boxes are evaluated with ground truth provided by ICDAR. The left column indicates the ground truth bounding boxes while the right indicates the predicted bounding boxes. The evaluation follows the method that was provided by [1]. Green boxes indicate perfect matches. The blue boxes indicate one-to-many under-segmentation of prediction, where characters are connected into larger bounding boxes. Magenta bounding boxes indicate the predication has low recall, but acceptable precision. Where yellow indicates low precision but good recall. The red bounding boxes indicate a wrong detection.

Word BB level. The characters are connected using edge classification results from Word-Graph to form word bounding boxes. The word level accuracy is computed and compared with current state-of-art systems. In this step, all edges between pairs of characters within each image are classified as within or between words. Only within word edges are used to form linkage and to form word bounding boxes.

The word bounding box is saved in text files in ICDAR 2015 format. The text files are uploaded and evaluated using ICDAR 2015 web-based tools to compare with competition participants. The web-based tool produces the final evaluation metrics in terms of precision, recall and F-measure. Hit, miss, and partial detection are marked and shown in web pages. Some examples are shown in Figure 6.11 and Figure 6.12.

Figure 6.11: Example results of text detection in natural scene, evaluated by ICDAR 2015 robust reading competition web tool. The green bounding box indicates the correct detection.

Figure 6.11 shows examples where word bounding boxes are successfully detected, which are indicated by green bounding boxes. Meanwhile, we still have space to improve. Figure 6.12 shows cases where our text detection system fails or partially fails. Figure 6.12a shows a case where two words are overlapping to each other are are falsely segmented into one word. Figure 6.12b and 6.12c shows an image of a book cover, where two lines of text are falsely segmented into one word. Figure 6.12d shows an image where text are over-segmented. The words are composed of characters that are far away from each other, therefore, Word-Graph miss-classify each character as a single word. Notice a word 'Engineering' is falsely classified as two words in this case. The image is suffered from very strong highlight and the information is lost and cannot be recovered. Therefore, heuristic rules, human knowledge or lexicon model must be used to combine characters into a single word. In Figure 6.12e, multiple false positive rectangles are shown as red. These are digital numbers lay in the background of the books. These should not be count as false positives, as the definition of text is blurred in this case. In the last sample, Figure 6.12f, the word is also over-segmented due to the long distance between characters.

More examples are shown in Figure 6.13 6.14 6.15 6.16, where red bounding boxes represent missing and false positives, green bounding boxes represent correct detections, yellow bounding boxes represent under-segmentation, and blue bounding boxes represent over-segmentations. Both yellow and blue bounding boxes indicate one-to-many and many-to-one cases, therefore, the performance is penalized by a rate of 80% for those cases.

(a) (b) (c)

(d) (e) (f)

Figure 6.12: Example partially incorrect results of text detection in natural scene, The bounding boxes marked the text detector's prediction. Red box indicates wrong detection, yellow indicates under-segmentation where multiple words are connected into a single large bounding boxes, and blue indicates over-segmentation where words are broken or characters are separated.

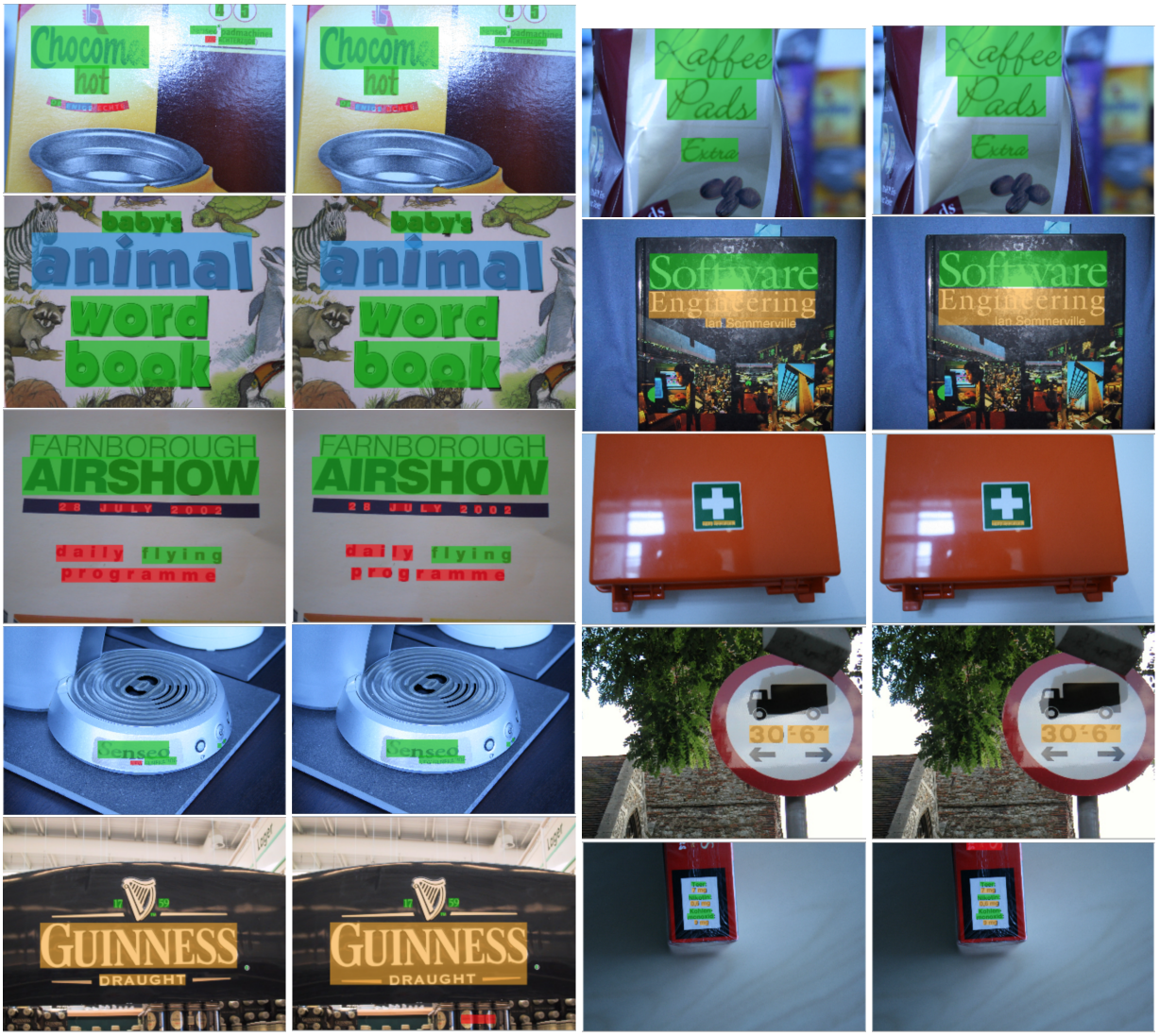
From the results, we notice that many over-segmented texts are marked as red, although characters are correctly detected. This is caused by ICDAR evaluation method of computing overlapping area. Since many words are written with very large distances between characters in natural scene images. Since the spaces between characters are discarded by prediction, the over-segmented bounding boxes have much smaller total area comparing to ground truth which includes those spaces. Therefore, some bounding boxes are marked as red and counted as missing or false positives. They are penalized heavier (treated as misses) than it supposes to be since over-segmentation should only be deducted by 20%.

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

Figure 6.13: Examples of character bounding box evaluation results with comparison of ground truth and detections. (a)(c) are ground truth bounding boxes; (b)(d) are prediction bounding boxes. Red bounding boxes represent missing and false positives, green bounding boxes represent correct detections, yellow bounding boxes represent under-segmentation, and blue bounding boxes represent over-segmentations.

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

Figure 6.14: Examples of character bounding box evaluation results with comparison of ground truth and detections. (a)(c) are ground truth bounding boxes; (b)(d) are prediction bounding boxes. Red bounding boxes represent missing and false positives, green bounding boxes represent correct detections, yellow bounding boxes represent under-segmentation, and blue bounding boxes represent over-segmentations.



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

Figure 6.15: Examples of character bounding box evaluation results with comparison of ground truth and detections. (a)(c) are ground truth bounding boxes; (b)(d) are prediction bounding boxes. Red bounding boxes represent missing and false positives, green bounding boxes represent correct detections, yellow bounding boxes represent under-segmentation, and blue bounding boxes represent over-segmentations.

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

Figure 6.16: Examples of character bounding box evaluation results with comparison of ground truth and detections. (a)(c) are ground truth bounding boxes; (b)(d) are prediction bounding boxes. Red bounding boxes represent missing and false positives, green bounding boxes represent correct detections, yellow bounding boxes represent under-segmentation, and blue bounding boxes represent over-segmentations.

The summary of machine learning algorithms used in each stage and component of text detection systems are listed in Table 6.5. Since different machine learning algorithms are tested for different components, the algorithms with the highest accuracy is kept for the final systems for the natural scenes and document images. The algorithms utilized in the final system is marked by bold font.

Different feature learning methods are proposed using random seeding or support vector seeding. Classification methods are proposed to test detection accuracy, including AdaBoost, cascaded AdaBoost, SVM, and random forest. AdaBoost is selected for natural scenes. MetaBoost, which is a combination of three

branch AdaBoost, is used for engineer drawing. Validation classifiers use AdaBoost for the first stage and random forest for the second. And segmentation classifier utilizes random forest.

Table 6.5: A summary of machine learning methods tested for our proposed text detection is listed. Both feature learning, detection, validation and segmentation methods are shown. The methods utilized in final system are marked with bold.

	Feature Learning	Detection	Validation	Segmentation
Natural Scene	**Convolutional K-means (random seeding)**	**AdaBoost**	**AdaBoost**	AdaBoost
	Convolutional K-means (SV seeding)	RF	**RF**	**RF**
		SVM	SVM	
		Cascaded		
Engineer Drawing	**Convolutional K-means (FG, BG, Both)**	**MetaBoost**	-	-
	Convolutional K-means (FG or BG or Both)	AdaBoost	-	-

The pixel level, character bounding box level and word bounding box level evaluation are shown in Table 6.6 and Figure 6.17. In pixel level, the proposed system achieved 93.14% f-measure, which means a very high proportion of pixels are detected correctly by our system. The pixel level evaluation only considers the overall number of pixels, therefore, a large character has heavier weights in detection performance than a small character. Even the characters are detected correctly if the stroke width is different from the ground truth, the performance is not perfect. Therefore, our system achieved very high accuracy in terms of total pixels detected without considering segmentation results.

The character bounding boxes performance is lower than pixel level (90.26%). False character segmentations could reduce the performance: Some touching characters are grouped into the same bounding box; characters containing different CCs ('i', 'j', etc.) or breaking strokes are divided into separate bounding boxes. Therefore, character bounding box performance is lower than pixel level.

The word bounding box is evaluated using ICDAR 2015 tools and gets the lowest performance among three different evaluation methods. The segmentation error decreases the performance since some words within the same text line are grouped into single bounding box or words with large character distances are separated into different bounding boxes, as shown in example figures.

The word bounding box performance is then compared with ICDAR 2013 and 2015 participant systems and achieved better than state-of-art performance, as shown in Table 6.7 and Figure 6.18. The precision, recall and f-measure are all higher than the Top-1 Competition team, StradVision. The final recall is 81.02%, precision is 93.39% and f-measure is 86.77% for word bounding box level.

Table 6.6: ICDAR 2015 Focused Scene Text Results in different evaluation levels, including pixel, character bounding box (Char BB) level and word bounding box (Word BB) levels.

Level	Recall (%)	Precision (%)	F-measure (%)
Pixels	92.75	93.53	93.14
Char BB	87.51	93.20	90.26
Word BB	81.02	93.39	86.77

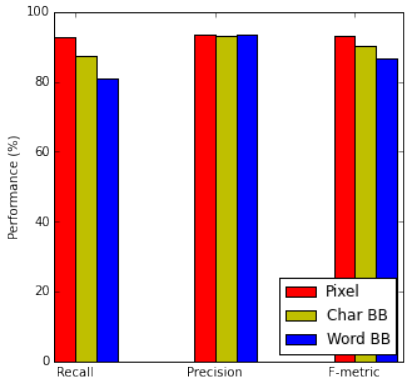


Figure 6.17: Pixel, character bounding box and word bounding box level detection results, in terms of precision, recall and f-measure.

Table 6.7: Word Detection Metrics (ICDAR Robust Reading Task). We compare our system's performance with state-of-the-art system from 2013 and 2015 competitions. The top 3 systems for each competition are listed. We achieved higher recall, precision and F-metric values comparing to the best systems. No lexicon model or OCR is used to achieve this accuracy.

Method Name	Recall(%)	Precision (%)	F-score (%)
Our System	**81.02**	**93.39**	**86.77**
2015 Competition			
StradVision	80.15	90.93	85.20
Text-CNN [79]	76.95	92.78	84.13
VGGMaxNet [92]	77.32	92.18	84.10
2013 Competition			
USTB_TexStar [93, 94, 95]	66.45	88.47	75.89
Text Spotter [77, 96, 97, 78]	64.84	87.51	74.49
CASIA_NLPR [98, 99]	68.24	78.89	73.18

6.3 Summary

The region growing to construct characters and word segmentation method (Word-Graph) are discussed in this chapter. The previous patch detection methods could produce patch level hotmaps for text detection,

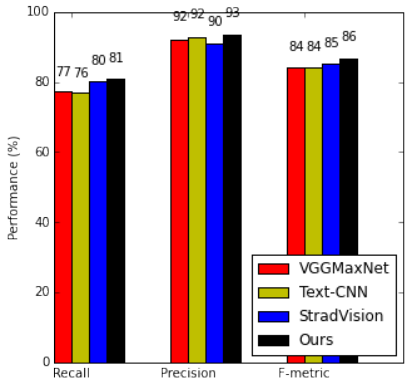


Figure 6.18: Performance comparison with state-of-art systems in ICDAR 2015 competitions, including StradVision, Text-CNN and VGGMaxNet. The proposed system achieved best performance among all metrics.

but it is not convenient to be used directly to compare with other state-of-art systems. In order to compare the performance, algorithms are developed to convert patch level detection into pixel level, grouped into CCs and then words. In our system, the patch scanning based detection algorithm (Text-Conv) is combined with CC based detection method (region growing, validation and segmentation) to produce a state-of-art performance on word bounding box level.

The region growing method uses fine detection locations as seeds to form CCs with edge and color information. The CCs are then pruned by validation classifier to reduce false positives. The segmentation group characters into words, meanwhile, using segmentation results, one can further improve performance by adding a second stage validation classification.

For segmentation, MST is used to prune training samples. Inner word and inter-word edge samples are balanced using random selection subsampling method. Then random forest algorithm is implemented to classify edges. The CCs and segmentation predictions are then used together to form words.

The region growing method over-generate samples using different parameters. In the future, the region growing method could be modified, therefore, the optimal parameter could be found from training, avoiding over-generations. More features could be used for segmentation and validation classifiers, e.g. stroke width transform, HoG and SIFT, to further increase the accuracy of detection.

For the future experiment, one can use techniques to form CCs directly from images, e.g. MSER, and perform segmentation, grouping and validation directly on those candidates. The new method can be compared with our current system and might achieve faster execution since no patch scanning is required.

On the other hand, an approach that tries to detect words directly and holistically rather than built bottom-up from individual characters might also be studied.

We are also interested in how well would the system work with other languages, like Cantonese, Mandarin, and Japanese. Since only visual features are used for text detection in natural scenes, our system should obtain high accuracy for different languages if trained properly.

Chapter 7

Conclusion

Our thesis statement **A simple algorithm for text detection in natural scenes can be produced to get higher accuracy than current systems without the use of image meta-data, OCR, or textual language models.** was supported by many experiments introduced in this dissertation. Visual features are learned, used along with subsequent processing methods to detection text in different images, including natural scenes. Similar feature learning and detection algorithm are extended to other images, e. g. patent drawings, and achieved very competitive performance.

We explored different feature learning approaches, classification methods, and different ground truth. We compared the proposed system with current state-of-art systems on different types of images, including natural scenes and engineer drawings.

The proposed system accurately detects text at the pixel, character, and word levels. We discussed the feasibility and effectiveness of using different methods for feature learning, including convolutional k-means with random selection and support vector seeding and combining features from different datasets using MetaBoost.

The cascaded classification method is proposed, including patch scanning based and CC based methods. For patch based step, contextual features are used in coarse stage, and different rotation and aspect ratios are used for fine stage.

Convolutional, shape, color, edge and spatial relationship features that represent different levels of intrinsic text characteristics are used in a cascaded detection model. More visual information is used in CC based stages to obtain a higher accuracy, this was made possible by utilizing different levels of ground truth,

including first bounding box level and then a combination of pixel and bounding box level.

7.1 Answers to Research Questions

Through our experiment, we tried to answer several interesting research questions:

1. Is visual feature alone sufficient for high-accuracy text detection in natural scenes?
2. How can features obtained via convolutional k-means be made more discriminating?
3. How can detection accuracy be improved by combining convolutional k-means detection with subsequent processing?
4. How can we use different levels of ground truth information to improve detection accuracy?
5. How can convolutional k-means be used effectively to detect text in images other than natural scenes?
6. How does detection in grayscale document images using convolutional k-means differ from detection in color images from natural scenes?

The first question is answered by comparing our system performance with state-of-art systems in ICDAR 2015 text detection competitions. Since we achieved higher accuracy than the top-1 player without using other information like OCR or lexicon model, it supports the hypothesis that visual feature alone is sufficient for accurate detection, as shown in Chapter 3, 4 and 6.

The second question discusses the possibilities of different feature learning models could enhance features' discrimination powers. We start by learning features with convolutional k-means with different initialization methods, first with randomly selected points and then with support vectors. The experiments support that combining different feature learning initialization methods could help achieve better separability. We also tested a method called MetaBoost, which combines the features learned from different datasets, including foreground, background, and total samples. The MetaBoost algorithm combines the feature banks and gets improved accuracy. Both experiments show that improving the diversity of features leads to a more efficient detection. The discussions are included in Chapter 3.

The third question discusses the subsequent processing which utilizes the learned features and forms classifiers to select foreground regions. Chapter 4 covers this topic.

For natural scene, since the variation of data is large and background is complicated, the searching for text location is divided into several stages. The patch scanning based method and CC based method are combined to generate bounding box coordinates of words. The patch-based method consists of coarse and fine detections, representing glancing and focusing. The detected patches are converted to pixels and CCs, then grouped and segmented into words.

In coarse stage of patch based detection, contextual detection windows that include features from neighboring characters and/or background improves accuracy. This is consistent with earlier work where incorporating neighboring character detections improve detection accuracy [89, 90]. In fine stage, using differing aspect ratios and rotations improves detection accuracy dramatically, without the need to retrain the patch classifier.

The fourth question discusses how utilization of different levels of ground truth could help increase accuracy. For natural scenes, different levels of ground truth are provided and utilized in training and evaluations.

As in Chapter 4, the bounding box ground truth is firstly used to label patches as foreground and background with different criterions for coarse and fine detection. For coarse detection, loose criterions are used; for fine detection, rigid criterions are used to label foreground patches. In Chapter 6, Pixel level ground truth is used to label CCs for CC based detection. We used both levels of ground truth to make possible of building a system combining of both patch-based and CC-based detection.

In Chapter 6, for evaluation, pixel, character bounding box and word bounding box level precision, recall and f-measures are computed. Each level reflects a different aspect of performance of the detector. Pixel level evaluation measures the overall detected pixels and compares with the ground truth pixels; character level evaluation measures the number of correctly detected characters and penalizes incorrect character segmentation; word level measures the number of correctly detected words and penalizes incorrect word segmentation. The word level measurement combines the accuracy of detection and segmentation, suppresses the effects of different stroke widths between prediction and ground truth, averages the weights between large and small characters. Therefore, it is a more meaningful evaluation metric for text detection in natural scene.

The fifth question discusses the effects when applying similar feature learning algorithm to different kinds of images, e.g. engineer drawing diagrams. The features learning method similar to natural scene

is utilized in Chapter 5, but techniques specifically designed for diagram images are used to remove noise objects. OCR is performed and lexicon model is applied.

For document images, the system used relative simpler patch based detection stage, but much more post-processing and denoise methods are proposed to remove false positives. OCR are implemented to further reduce false positives and produce text labels.

This is necessary for USPTO part label detection since many texts (figure labels, page titles, tables) within those images are not part labels and can only be removed by checking their lexicons.

The sixth question discusses the difference between a colored image and gray-scale image and was discussed in Chapter 5. First, the natural scene images are formed by regions, patterns and blobs with different colors, while diagram images often consist of background objects like lines and curves which are very similar to texts. Therefore, one needs to apply OCR and lexicon model to remove objects that are indistinguishable from text using visual features alone. Second, the grayscale engineer drawing contains large areas of blank space. And one could avoid searching in those areas which contain only plain white pixels.

7.2 Contribution

Many new methodologies and improvements upon previous algorithms are proposed in our thesis.

Feature learning method convolutional k-means is modified and improved. By combining convolutional K-means with random and support vector seeding, one could learn both discriminative and representative features from the data automatically. Previously, feature learning is known as unsupervised process. We corporate supervised feature learning method to utilize label information and enhance performance on testing data. This hypothesis is supported by support vector seeded feature learning performance on ICDAR dataset.

Features learned from different dataset are used in a MetaBoost manner. Therefore, the learned feature from different dataset could generate classifiers that are less correlated to each other. Using the classification results from these classifiers as features, one could produce stronger performance. This hypothesis is supported by our USPTO part label recognition system. Both modifications of feature learning method support that by increasing the diversity of learned features, the classifier accuracy could be improved.

Subsequent processing and detection method that utilizes the learned features are refined and improved. Coarse-to-fine detection is proposed which is similar to human visual attention. Contextual features are used in coarse scanning and differing rotation angles and aspect ratios are used in fine scanning.

Innovative region growing and graph model based segmentation methods are introduced. Patch level prediction map is converted to CCs. And precision is improved by using two stage of validation with spatial relationship information.

Different levels of ground truth are combined to train and evaluation detection system. By using ground truth on two levels, including pixel and bounding box level. We managed to train validation classifiers and improve the overall performance.

Detection systems with high accuracy are proposed. The ICDAR natural scene images are used to test our detection system and an accuracy better than the state-of-art is obtained without using a language model. The best results for USPTO part label detection is also obtained, but with the inclusion of a lexicon model.

New analyzes are included for text detection system. Different approaches for feature learning are discussed, including different seeding methods and their effects on the final performance. Feature learning with different datasets are also discussed, and novel algorithms MetaBoost for combining features from different branches are proposed.

Classification methods are studied. Cascaded classifications are tested and compared with single stage AdaBoost. We compared methods to improve accuracy of the system by using different thresholds of precisions (increasing, decreasing and zero-gradient for F-measure).

The effects of using different sample equalization and classification methods are tested for word segmentation system, Word-Graph. And the effects of different denoise methods are tested for document images.

7.3 Future Work

For the convolutional K-means algorithm, including the randomized seeding using support vectors, statistical hypothesis tests can be performed to characterize the improvement. Different initialization techniques, using regular random seeds, randomly sampled support vectors and the combination of the two can be run multiple times for each case. Then the means and variances of accuracies in each case can be computed.

Those values can then be used to evaluate the significance of the improvement of the proposed method.

The system could be accelerated using pre-segmentation techniques, e.g. MSER. The CCs could be extracted directly from images using these methods, and patch could be extracted based on CC sizes and locations. Therefore, fewer scales and locations are needed to be scanned. This way, one could potentially increase the speed of the detection by a significant amount.

It is interesting to note that we achieved high accuracy for natural scenes using only visual features, without the use of a language model or dictionary. However, character recognition could be integrated into our system. Using character labels and a lexicon, one can probably increase the detection rate further for a specified language.

Besides texts, the feature learning, classification and subsequent processing methods could be adapted to other types of objects, e.g. cars, pedestrians and faces. Different from texts, which are composed of strokes and lines, these objects are more complicated. They are usually combined with different components. For example, faces are composed by eyes, noses, lips etc. Therefore, feature learning methods might need to be modified to adapt their properties. Deeper learning algorithms with multiple layers might be necessary for detection of these objects. It is very interesting to develop algorithms for different kinds of objects using deeper or wider networks and compare their performance in the future.

Bibliography

[1] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, *et al.*, "Icdar 2015 competition on robust reading," *Int'l Conf. Document Analysis and Recognition (ICDAR)* , 2015.

[2] L. O'Gorman and R. Kasturi, *Document image analysis*, vol. 39, IEEE Computer Society Press Los Alamitos, CA, 1995.

[3] B. Su, S. Lu, and C. L. Tan, "Binarization of historical document images using the local maximum and minimum," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pp. 159–166, ACM, 2010.

[4] M. Paul, H. Okuma, H. Yamamoto, E. Sumita, S. Matsuda, T. Shimizu, and S. Nakamura, "Multilingual mobile-phone translation services for world travelers," in *22nd International Conference on on Computational Linguistics: Demonstration Papers*, pp. 165–168, Association for Computational Linguistics, 2008.

[5] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *Intelligent Transportation Systems, IEEE Transactions on* **5**(1), pp. 42–53, 2004.

[6] C. C. Aggarwal and C. Zhai, *Mining text data*, Springer Science & Business Media, 2012.

[7] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *Image Processing, IEEE Transactions on* **23**(11), pp. 4737–4749, 2014.

[8] J. Weinman, E. Learned-Miller, and A. Hanson, "Scene text recognition using similarity and a lexicon with sparse belief propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **31**, pp. 1733–1746, Oct 2009.

[9] G. Nagy, "Twenty years of document image analysis in pami," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**, pp. 38–62, Jan 2000.

[10] V. Wu, R. Manmatha, and E. Riseman, "Textfinder: an automatic system to detect and recognize text in images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **21**, pp. 1224–1229, Nov 1999.

[11] A. Jain and B. Yu, "Document representation and its application to page decomposition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **20**, pp. 294–308, Mar 1998.

[12] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)* **7**(2), pp. 84–104.

[13] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition* **37**(5), pp. 977 – 997, 2004.

[14] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **37**, pp. 1480–1500, July 2015.

[15] J. Smith, "Image retrieval evaluation," in *Content-Based Access of Image and Video Libraries, 1998. Proceedings. IEEE Workshop on*, pp. 112–113, Jun 1998.

[16] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**, pp. 1349–1380, Dec 2000.

[17] M. Moll, H. Baird, and C. An, "Truthing for pixel-accurate segmentation," in *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pp. 379–385, 2008.

[18] E. H. B. Smith, "An analysis of binarization ground truthing," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, *DAS '10*, pp. 27–34, ACM, (New York, NY, USA), 2010.

[19] E. Smith and C. An, "Effect of "ground truth" on image binarization," in *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pp. 250–254, 2012.

[20] A. Torralba, B. Russell, and J. Yuen, "Labelme: Online image annotation and applications," *Proceedings of the IEEE* **98**, pp. 1467–1484, Aug 2010.

[21] C. Nieuwenhuis and D. Cremers, "Spatially varying color distributions for interactive multilabel segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**, pp. 1234–1247, May 2013.

[22] N. Zhou, W. Cheung, X. Xue, and G. Qiu, "Collaborative and content-based image labeling," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, Dec 2008.

[23] S. Chen and H. S. Baird, "Pixel accurate document image content extraction," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, *SAC '11*, pp. 245–251, ACM, (New York, NY, USA), 2011.

[24] A. Yuille, S.-C. Zhu, D. Cremers, Y. Wang, B. Yao, and X. Yang, "Lecture notes in computer science," **4679**, pp. 169–183, 2007.

[25] Y. Qu, C. Chen, D. Wu, and Y. Xie, "Image labeling via incremental model learning," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1573–1576, Sept 2010.

[26] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1457–1464, Nov 2011.

[27] E. Saund, J. Lin, and P. Sarkar, "Pixlabeler: User interface for pixel-level labeling of elements in document images," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 646–650, July 2009.

[28] H. Bunke, A. L. Spitz, A. Antonacopoulos, D. Karatzas, and D. Bridson, "Ground truth for layout analysis performance evaluation," **3872**, pp. 302–311, 2006.

[29] J. D. Hobby, "Matching document images with ground truth," **1**(1), pp. 52–61, 1998.

[30] H. Baird, "Document image defect models and their uses," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pp. 62–67, Oct 1993.

[31] P. Heroux, E. Barbu, S. Adam, and E. Trupin, "Automatic ground-truth generation for document image analysis and understanding," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, **1**, pp. 476–480, 2007.

[32] G. Zi and D. Doermann, "Document image ground truth generation from electronic text," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, **2**, pp. 663–666 Vol.2, 2004.

[33] T. Strecker, J. van Beusekom, S. Albayrak, and T. Breuel, "Automated ground truth data generation for newspaper document images," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 1275–1279, July 2009.

[34] H. Bunke, A. L. Spitz, L. Yang, W. Huang, and C. Tan, "Lecture notes in computer science," **3872**, pp. 324–335, 2006.

[35] M. A. Ramírez-Ortegón, L. L. Ramírez-Ramírez, V. Märgner, I. Ben Messaoud, E. Cuevas, and R. Rojas, "An analysis of the transition proportion for binarization in handwritten historical documents," *Pattern Recognition* **47**, pp. 2635–2651, 8 2014.

[36] G. E. Kopec, "Multilevel character templates for document image decoding," **3027**, pp. 168–177, 1997.

[37] D. Snyder, "Text detection in natural scenes through weighted majority voting of dct high pass filters, line removal, and color consistency filtering," Master's thesis, Rochester Institute of Technology, 2011.

[38] S. Lucas, "Icdar 2005 text locating competition results," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pp. 80–84 Vol. 1, Aug 2005.

[39] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision* **77**(1-3), pp. 157–173, 2008.

[40] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, **2**, pp. 1605–1614, 2006.

[41] I. Irfanullah, N. Aslam, J. Loo, M. Loomes, and R. Roohullah, "A framework for high level semantic annotation using trusted object annotated dataset," in *Signal Processing and Information Technology (ISSPIT), 2010 IEEE International Symposium on*, pp. 491–495, Dec 2010.

[42] D. Iakovidis and C. Smailis, "Efficient semantically-aware annotation of images," in *Imaging Systems and Techniques (IST), 2011 IEEE International Conference on*, pp. 146–149, May 2011.

[43] I. Kadar and O. Ben-Shahar, "Small sample scene categorization from perceptual relations," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2711–2718, June 2012.

[44] S. Mao and T. Kanungo, "Empirical performance evaluation methodology and its application to page segmentation algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **23**, pp. 242–256, Mar 2001.

[45] P. Stathis, E. Kavallieratou, and N. Papamarkos, "An evaluation technique for binarization algorithms.," *J. UCS* **14**(18), pp. 3011–3030, 2008.

[46] P. Sarkar, E. Saund, and J. Lin, "Classifying foreground pixels in document images," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 641–645, 2009.

[47] F. Shafait, D. Keysers, and T. Breuel, "Pixel-accurate representation and evaluation of page segmentation in document images," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, **1**, pp. 872–875, 2006.

[48] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging* **13**(1), pp. 146–168, 2004.

[49] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "Icdar 2003 robust reading competitions," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pp. 682 – 687, aug. 2003.

[50] S. Lucas, "Icdar 2005 text locating competition results," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pp. 80 – 84 Vol. 1, aug.-1 sept. 2005.

[51] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos, "Icdar 2009 page segmentation competition," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 1370 –1374, july 2009.

[52] D. Karatzas, S. Mestre, J. Mas, F. Nourbakhsh, and P. Roy, "Icdar 2011 robust reading competition - challenge 1: Reading text in born-digital images (web and email)," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 1485 –1490, sept. 2011.

[53] A. Shahab, F. Shafait, and A. Dengel, "Icdar 2011 robust reading competition challenge 2: Reading text in scene images," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 1491–1496, 2011.

[54] S. Zhu and R. Zanibbi, "Label detection and recognition for uspto images using convolutional k-means feature quantization and ada-boost," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pp. 633–637, IEEE, 2013.

[55] B. Gatos, K. Ntirogiannis, and I. Pratikakis, "Dibco 2009: document image binarization contest," **14**(1), pp. 35–44, 2011.

[56] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082* , 2013.

[57] K. Ntirogiannis, B. Gatos, and I. Pratikakis, "An objective evaluation methodology for document image binarization techniques," in *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pp. 217–224, Sept 2008.

[58] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2963 –2970, june 2010.

[59] K. I. Kim, K. Jung, and J.-H. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **25**(12), pp. 1631–1639, 2003.

[60] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng, "Text detection and character recognition in scene images with unsupervised feature learning," in *Int'l Conf on Document Analysis and Recognition (ICDAR)*, pp. 440–445, Sept 2011.

[61] T. Wang, D. Wu, A. Coates, and A. Ng, "End-to-end text recognition with convolutional neural networks," in *Int' l Conf. on Pattern Recognition (ICPR)*, pp. 3304–3308, Nov 2012.

[62] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3538–3545, 2012.

[63] H. Chen and etc., "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," in *ICIP*, 2011.

[64] J. Gao and L. Yang, "An adaptive algorithm for text detection from natural scenes," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, **2**, pp. II–84–II–89 vol.2, 2001.

[65] P. Clark and M. Mirmehdi, "Recognising text in real scenes," *International Journal on Document Analysis and Recognition* **4**, pp. 243–257, 2002. 10.1007/s10032-001-0072-2.

[66] D. Chen, J.-M. Odobez, and H. Bourlard, "Text detection and recognition in images and video frames," *Pattern Recognition* **37**(3), pp. 595 – 608, 2004.

[67] N. Sharma, P. Shivakumara, U. Pal, M. Blumenstein, and C. Tan, "A new method for arbitrarily-oriented text detection in video," in *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pp. 74–78, 2012.

[68] C. Yi and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," *Image Processing, IEEE Transactions on* **20**, pp. 2594 –2605, sept. 2011.

[69] M. Ben Halima, H. Karray, and A. M. Alimi, "Arabic Text Recognition in Video Sequences," *ArXiv e-prints* , Aug. 2013.

[70] C. Mancas-Thillou and B. Gosselin, "Color text extraction with selective metric-based clustering," *Computer Vision and Image Understanding* **107**(1–2), pp. 97 – 107, 2007. ¡ce:title¿Special issue on color image processing¡/ce:title¿.

[71] M. Leon, V. Vilaplana, A. Gasull, and F. Marques, "Region-based caption text extraction," in *Analysis, Retrieval and Delivery of Multimedia Content*, N. Adami, A. Cavallaro, R. Leonardi, and P. Migliorati, eds., *Lecture Notes in Electrical Engineering* **158**, pp. 21–36, Springer New York, 2013.

[72] X. Wang, L. Huang, and C. Liu, "A video text location method based on background classification," *International Journal on Document Analysis and Recognition* **13**, pp. 173–186, 2010. 10.1007/s10032-009-0104-x.

[73] M. Petter, V. Fragoso, M. Turk, and C. Baur, "Automatic text detection for mobile augmented reality translation," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 48 –55, nov. 2011.

[74] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, *CVPR'04*, pp. 366–373, IEEE Computer Society, (Washington, DC, USA), 2004.

[75] K. Zhu and etc., "Using adaboost to detect and segment characters from natural scenes," in *Proceedings of the International Workshop on Camera-based Document Analysis and Recognition*, 2005.

[76] S. Hanif, L. Prevost, and P. Negri, "A cascade detector for text detection in natural scene images," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1 –4, dec. 2008.

[77] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *Computer Vision (ACCV)*, pp. 770–783, Springer, 2011.

[78] L. Neumann and J. Matas, "Text localization in real-world images using efficiently pruned exhaustive search," in *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 687–691, Sept 2011.

[79] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-Attentional Convolutional Neural Networks for Scene Text Detection," *arXiv:1510.03283* , Oct. 2015.

[80] H. Shen and J. Coughlan, "Finding text in natural scenes by figure-ground segmentation," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, **4**, pp. 113 –118, 10 2006.

[81] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1457–1464, 2011.

[82] Y.-F. Pan, X. Hou, and C.-L. Liu, "A hybrid approach to detect and localize texts in natural scene images," *Image Processing, IEEE Transactions on* **20**, pp. 800 –813, march 2011.

[83] A. Chowdhury, U. Bhattacharya, and S. Parui, "Scene text detection using sparse stroke information and mlp," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 294–297, 2012.

[84] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**(8), pp. 1798–1828, 2013.

[85] M. Dikmen, D. Hoiem, and T. Huang, "A data driven method for feature transformation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3314–3321, 2012.

[86] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature* **521**, pp. 436–444, 05 2015.

[87] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* **86**, pp. 2278–2324, Nov 1998.

[88] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Analysis and Machine Intelligence* **32**, pp. 1627–1645, Sept 2010.

[89] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," in *CVPR*, **1**, pp. I–261–I–268 Vol.1, June 2004.

[90] Y.-F. Pan, X. Hou, and C.-L. Liu, "Text localization in natural scene images based on conditional random field," in *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 6–10, July 2009.

[91] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *ICML*, **96**, pp. 148–156, 1996.

[92] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition," *arXiv:1406.2227* , June 2014.

[93] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao, "Robust text detection in natural scene images," *IEEE Trans. Pattern Analysis and Machine Intelligence* **36**, pp. 970–983, May 2014.

[94] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao, "Accurate and robust text detection: A step-in for text retrieval in natural scene images," in *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, *SIGIR '13*, pp. 1091–1092, ACM, (New York, NY, USA), 2013.

[95] X. Yin, X.-C. Yin, H.-W. Hao, and K. Iqbal, "Effective text localization in natural scene images with mser, geometry-based grouping and adaboost," in *Int'l Conf. on Pattern Recognition (ICPR)*, pp. 725–728, Nov 2012.

[96] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *CVPR*, pp. 3538–3545, June 2012.

[97] L. Neumann and J. Matas, "On combining multiple segmentations in scene text recognition," in *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 523–527, Aug 2013.

[98] Y.-M. Zhang, K. Huang, and C.-L. Liu, "Fast and robust graph-based transductive learning via minimum tree cut," in *Int'l Conf. on Data Mining (ICDM)*, pp. 952–961, Dec 2011.

[99] B. Bai, F. Yin, and C. L. Liu, "Scene text localization using gradient local correlation," in *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 1380–1384, Aug 2013.

[100] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *CoRR* **abs/1406.2227**, 2014.

[101] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

[102] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.* **11**, pp. 3371–3408, Dec. 2010.

[103] G. Huang, H. Lee, and E. Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2518–2525, 2012.

[104] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[105] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR 2001*, **1**, pp. I–511 – I–518 vol.1, 2001.

[106] J. Sun, J. Rehg, and A. Bobick, "Automatic cascade training with perturbation bias," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, **2**, pp. II–276–II–283 Vol.2, 2004.

[107] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics* **28**(2), pp. 337–407, 2000.

[108] Y. F. Robert E. Schapire, *Boosting: Foundations and Algorithms*, ch. 9.2. The MIT Press, 2012.

[109] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*, pp. 561–580, Springer, 2012.

[110] T. Kanungo, R. M. Haralick, and D. Dori, "Understanding engineering drawings: A survey," in *In Proceedings of First IARP Workshop on Graphics Recognition*, pp. 217–228, 1995.

[111] M. Lupu and A. Hanbury, "Patent retrieval.," *Foundations and Trends in Information Retrieval* **7**(1), pp. 1–97, 2013.

[112] N. Bhatti and A. Hanbury, "Image search in patents: a review," *International Journal on Document Analysis and Recognition (IJDAR)* **16**(4), pp. 309–329, 2012.

[113] C. Riedl, R. Zanibbi, M. A. Hearst, S. Zhu, M. Menietti, J. Crusan, I. Metelsky, and K. R. Lakhani, "Detecting figures and part labels in patents: competition-based development of graphics recognition algorithms," *International Journal on Document Analysis and Recognition (IJDAR)* , pp. 1–18, 2016.

[114] S. Zhu and R. Zanibbi, "Label detection and recognition for uspto images using convolutional k-means feature quantization and ada-boost," in *Int'l Conf. on Document Analysis and Recognition (ICDAR)*, pp. 633–637, Aug 2013.

[115] R. Smith, "An overview of the tesseract ocr engine," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, **2**, pp. 629 –633, sept. 2007.

[116] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *CVPR*, pp. 1083–1090, June 2012.

[117] A. Canter, "Assessing threat posted to video captcha by ocr-based attacks," master's project, Rochester Institute of Technology, NY, USA, June 2013.

[118] L. Breiman, "Random forests," *Machine Learning* **45**(1), pp. 5–32.

[119] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research* , pp. 321–357, 2002.