CURVED TEXT DETECTION AND GROUND TRUTH GENERATION FOR NATURAL SCENE IMAGES

CH'NG CHEE KHENG

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

CURVED TEXT DETECTION AND GROUND TRUTH GENERATION FOR NATURAL SCENE IMAGES

CH'NG CHEE KHENG

**THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF COMPUTER SCIENCE**

**FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2018

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Ch'ng Chee Kheng

Registration/Matrix No.: WGA150035

Name of Degree: Master of Degree

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Curved Text Detection in Natural Images

Field of Study: Artificial Intelligence (Computer Science)

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

CURVED TEXT DETECTION

IN NATURAL IMAGES

ABSTRACT

At present, text orientation is not diverse enough in the existing scene text datasets. For instance, text with curve-orientation has close to zero existence in them and thus received minimal attention from the community. Motivated by this phenomenon, a new scene text dataset, *Total-Text*, which emphasized on text orientations diversity has been collected as the major contribution of this work. It is the first properly scaled scene dataset that features three different text orientations: horizontal, multi-oriented, and curve-oriented. In addition, several studies regarding other important elements such as the practicality and quality of groundtruth, evaluation protocol, insights of curved text, and the annotation process are presented in this work as well. These elements are found to be as important as the images and groundtruth to facilitate a new research direction. In addition, Polygon-Faster-RCNN, a text detection baseline, has been proposed as the second major contribution of this work. It has demonstrated its ability in detecting text in all kinds of orientations. Images of Total-Text and its annotation are available at https://github.com/cs-chan/Total-Text-Dataset.

Keywords: Curved text, Scene text detection

PENGESANAN TEKS MELENGKUNG

DALAM IMEJ SEMULAJADI

ABSTRAK

Kini, kepelbagaian orientasi teks dalam dataset 'scene text' yang sedia-ada adalah didapati kurang mantap, sebagai contohnya, ketiadaan teks orientasi lengkung dalam dataset tersebut. Akibatnya, kajian berkaitan dengan teks orientasi lengkung mendapat perhatian minimum daripada komuniti 'scene text'. Dimotivasikan oleh fenomena ini, dataset 'scene text' yang baru, *Total-Text*, telah dikumpulkan sebagai kontribusi pertama dalam tesis ini. Kumpulan Total-Text menekankan pada kepelbagaian orientasi teks. Ia merupakan dataset pertama yang mempunyai tiga orientasi teks yang berbeza: 'horizontal', 'multi-oriented', dan 'curve-oriented'. Di samping itu, beberapa elemen penting lain seperti: kepraktisan dan kualiti groundtruth, protokol penilaian, pemahaman teks melengkung, dan proses anotasi juga telah dikajiankan. Elemen-elemen tersebut berberat sama dengan dataset untuk memudahkan penyelidikan baru. Selain itu, satu teks pengesanan baseline, yang bernama Polygon-Faster-RCNN, telah dicadangkan sebagai penyumbang kedua untuk kajian ini. Polygon-Faster-RCNN telah mendemonstrasikan keupayaannya dalam mengesan teks dalam semua jenis orientasi. Imej Total-Text dan anotasinya tersedia di https://github.com/cs-chan/Total-Text-Dataset.

Kata kunci: Teks melengkung, pengesanan teks

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude towards Dr. Chan Chee Seng. It would have not been possible for me to complete my Master's Degree without your passionate supervision. I thank you for believing in me and took me under your wing for this journey. I also thank you for all the opportunities that you have introduced to me, I will always remember that.

Secondly, my family members, they are the most supportive family that I could ever ask for. I would like to thank my parents for their mental and financial support. It is a shame for me to request for such a huge support at the age of 27, but your unquestioning responses always made me feel indescribably warm. It is exactly this warming sensation, which allowed me to move forward under the hardest circumstances.

Next, I would like to name some of the very important friends who helped me throughout this journey. They are Abhishek Jhawar, Chang Yang Loong, Lee Sue Han, Loh Yuan Peng, Lim Jian Han, Ng Chun Chet, Nurul Japar, Tan Jia Huei, and Tan Yin Hua. I thank you all for first showing me the right way to kickstart my research, then later for contributing in my data collection and annotation, endless paper proofreadings and presentation feedbacks, inspiring research ideas, etc. I will always remember all the ups and downs we went through in the past two years.

Another very important person to be mentioned is Dr. Lai Weng Kin. He was the one who opened this very door for me. Thank you Dr. Lai, for making this journey a possibility for me.

Lastly, Ms. Ch'ng Shin Fang, she is probably the biggest fuel to this journey of mine. Thank you for believing in me when I decided to embark on this journey. Thank you for choosing to stay by my side when I chose to lengthen the distance of our relationship. Thank you for willing to walk through all the hurdles that I have brought to our relation-

ship. Thank you for all the encouraging words when I went through the toughest times. Thank you for all the sacrifies you made to make the Australia trip possible. You are the biggest reason why I am able to finish this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

T3	Total-Text-Tool.
ANN	Artificial Neural Network.
CNN	Convolutional Neural Network.
IoU	Intersection over Union.
NMS	Non-Maximal Suppression.
Polygon-FRCNN	Polygon-Faster-RCNN.
RNN	Recurrent Neural Network.

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

Human communicates, not only verbally but often visually through written text, stationary signs, fancy billboards, banners, posters, etc. The constant need of conveying information in the human society produces an abundant existence of text in every corner of the world. Human leverages the existence of text in natural scenes to drive, find out the cafe he or she wants to go, understand the context of the piece of art on the wall, be aware of the sales happening in certain stores on the street, the application list goes on and on. The abundant of such information in our everyday scenes calls for the incorporation of it in the design of intelligence systems. Figure 1.1 shows two natural images filled with scene text. By giving machine the ability to understand text in the wild, we could have a smarter navigation system, convenient shoot-and-translate mobile application, efficient image retrieval system, etc. All of these applications depend on the progress of the scene text understanding research.

Both scene text detection and scene text recognition are two active research topics under the umbrella of scene text understanding. Given a natural scene image, the goal of scene text detection is to determine the existence of text, and return the location if it is present. W. Huang et al. (2014); Y. Liu & Jin (2017); X.-C. Yin et al. (2015, 2014a,b); Zhang et al. (2016); Zhou et al. (2017) Meanwhile, the current attempts at scene text recognition problem assumes pre-cropped text image patches as input, aims to recognise the text in it (i.e. output in human language) (Jaderberg et al. (2014); B. Shi et al. (2016); T. Wang et al. (2012)). Figure 1.2 visualizes the general pipeline of both challenges, and this work aims to address the scene text detection problem.

Like many research fields, the scene text detection challenge has several common datasets for researchers to benchmark their algorithms. Over the years, tougher scene text datasets were launched to better represent text instance in real-word scenarios. For

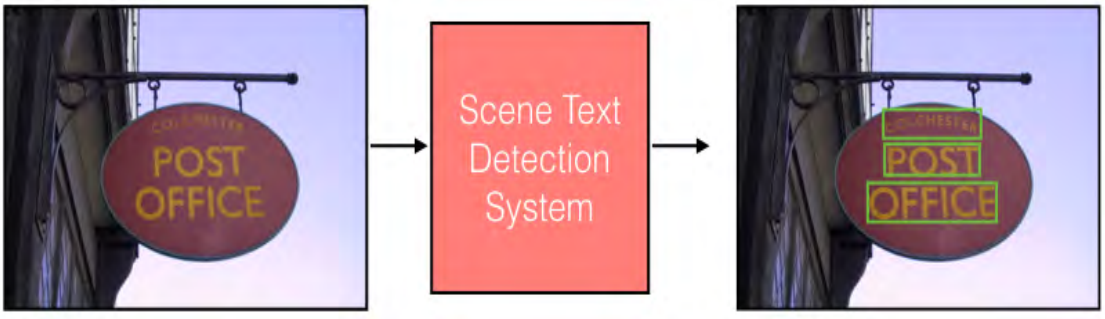
(a)

(b)

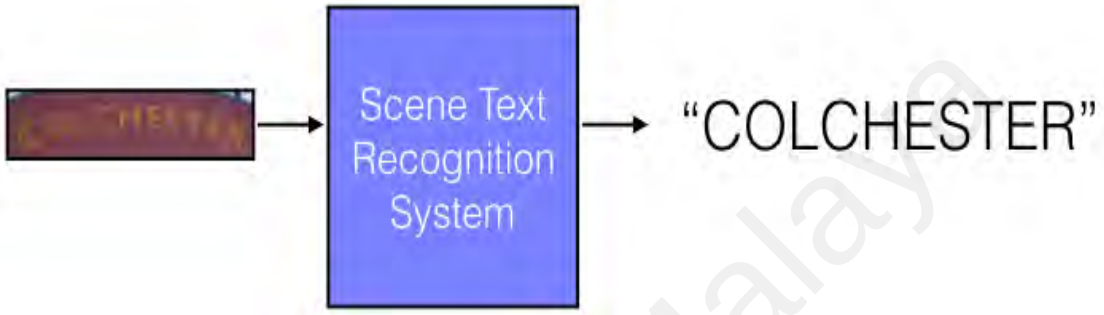
Figure 1.1: Image taken in (a): Time Square, New York, and (b): Jalan Tun Razak, Kuala Lumpur. Texts are an integral part of everyday scenes in the human society. It plays a big role in human communication.

instance, MSRA-TD500 (Yao et al. (2012)) were launched to introduce the problem of detecting text in arbritary orientation, the images of ICDAR2015 were captured without prior efforts in focusing, which is more likely the case in compared to well-captured photos in terms of application, the large scale COCO-Text (Andreas et al. (2016)) captured the wide scene diversity of the real-world, etc.

Despite the emergence of multiple datasets, text in curve orientation, in spite of its commonness in real-world scenes, went under the radar – it has close to zero existence in the existing scene text datasets. Without the motivation of a proper dataset, scene text detection algorithm with curved text in consideration is rarely seen. This would



(a)



(b)

Figure 1.2: Overview of (a): scene text detection, and (b): scene text recognition. At the point of this thesis writing, scene text detection is a prerequisite for the scene text recognition task.

prevent scene text detection system in achieving robust performance against text of all orientations in the real-world application. In the meanwhile, there is a concrete sign that the community is summoning the curved text data – multiple state-of-the-art performing multi-oriented scene text detectors have reported failure in detecting curved text (B. Shi et al. (2017); Xing et al. (2017); X.-C. Yin et al. (2015); Zhang et al. (2016)). Encouraged by this phenomenom, this work aims to explore the uncharted path of scene text detection – curved text detection, by facilitating this new research direction and providing a strong baseline solution.

1.1 Problem Statements

This work tries to address several problems revolving the core objective – curved text detection. The first and most important problem to solve is the lack of curved text data.

Figure 1.3 shows several famous signboards that feature curved text. As common as they are, curved text occupies less than 0.3% of the total text instances in the renowned scene text datasets like MSRA-TD500 and ICDAR2015. Being under-represented in the scene text datasets has several consequences. Firstly, it was overlooked by the scene text community, focus study on the curved text detection or recognition problem is rarely seen. To the best of author's knowledge, B. Shi et al. (2016) and Risnumawan et al. (2014)'s works were the only related studies before the introduction of the proposed dataset (Chng & Chan (2017)). Additionally, there are studies which shows bits and pieces of successful and failure detection result on the rare occurence of curved text in the existing datasets (B. Shi et al. (2017); Xing et al. (2017); X.-C. Yin et al. (2015); Zhang et al. (2016)); but the lack of a full scale dataset prevents them from evaluating their algorithm thoroughly on the ability to detect curved text. As mentioned earlier, studies or designs of detection algorithms based on under-represented data (i.e. missing of curved text) would suffer against the wild scenerios of real-life application.

Figure 1.3: Curved text is commonly seen in real world scenery.

The attempt to solve the first problem (i.e. collect a curved text dataset) is hindered by several roadblocks as well. Firstly, the conventional ground truth format such as the axis-aligned bounding box, rotated bounding box, quadrilateral does not fit the shape of curved text. Figure 3.9 illustrates the apparent downside of using the mentioned ground truth format in annotating curved text instances. Not only does such loosely fit ground truth affect the accuracy of the evaluation of benchmarking algorithms; the inclusion of large background area or overlapping of neighbour text instance makes it a bad training

data for learning based algorithms (W. He et al. (2017); Liao et al. (2017); W. Liu et al. (2016); Yao et al. (2016)). On a similar note, the design of Deteval evaluation protocol (Wolf & Jolion (2006)), was carried out based on the convention axis-aligned bounding box. The default setting of the protocol is found to be not optmised to cater the inclusion of curved text. It is a crucial problem to be addressed as the legitimacy of evaluation protocols plays an integral role in progressing the research field. Thirdly, curved text has more variants than text of other orientations (horizontal and multi-oriented), yet it remained unclear what kind of curved text sits in the wild. A better grasp of the curved text variety would be benefitial for future research problem. Lastly, the process of annotating is the largest bottleneck of collecting or scaling up a dataset. It is a slow and unrewarding process, yet the emergence of deep learning methods (Liao et al. (2017); Y. Liu & Jin (2017); Ma et al. (2017); B. Shi et al. (2017)) craves for more labeled data. All of these problems need to be addressed in the quest of facilitating a new research direction.

There is a large number of scene text detection works addressing scene text of horizontal (X. Chen & Yuille (2004); Epshtein et al. (2010); Kim et al. (2003); J.-J. Lee et al. (2011); Neumann & Matas (2013); Pan et al. (2011); Yi & Tian (2012, 2013); X. Yin et al. (2012); X.-C. Yin et al. (2013)) and arbritrary oriented (W. He et al. (2017); Y. Liu & Jin (2017); Tian et al. (2015); Yao et al. (2014); X.-C. Yin et al. (2015, 2014a,b); Zhang et al. (2016); Zhou et al. (2017)), but only little was contributed to the study of curved text detection (Risnumawan et al. (2014); Yuliang et al. (2017)). As such, this study aims to look into this glaring gap as the third and the final problem of this thesis.

1.2 Objectives

This work has three main objectives in response to the three main problems stated in Section 1.1. The first objective is to collect a new scene text dataset which prioritize the existence of curved text. The core idea of collecting this new dataset is to fill up the

gap in scene text datasets in terms of text orientations. The new dataset aims to spur an interest in the scene text community to look into the currently under-researched curved text detection problem.

On top of collecting images of curved text, the second main objective of this work is to look into all other supporting elements to pave the way for this new research direction. This task can be broken down into four sub-objectives: i) a new ground truth format is needed to properly bind text of all orientations, from horizontal to curved text, ii) the existing evaluation protocol needs to be fine-tuned to cater the introduction of curved text, iii) this work also aims to delve deeper into the variation of curved text, hoping to provides significant insights for future research journey, and iv) this work seeks for an effective method to increase the speed of annotation process, aspire to speed up future dataset collection effort.

The third main objective of this work is to propose a strong baseline to kick start the curved text detection research. The idea of the proposed method is to improve the robustness of text detection algorithm against text of all orientations, bridging its gap between the research and real-life application end.

1.3 Contributions

The first contribution of this work is the proposal of a new scene text dataset, namely *Total-Text*. To the best of author's knowledge, it is the first properly scaled scene text dataset that feature three major orientations: horizontal, multi-oriented, and curved. *Total-Text* consists of 1555 images in total, annotated with spatial location, transcription, and pixel mask for detection, recognition, and segmentation task. All of the images and ground truths were made publicly available for future research.

Alongside the dataset, the second contribution of this research is the studies of multiple supporting roles to faciliate this new research direction. As discussed, there are four

sub-objectives under this contribution. Firstly, a new ground truth format, regulated polygon was presented and used as the ground truth for *Total-Text*. The proposed regulated polygon is capable of binding text of all orientations tightly. Next, multiple experiments were conducted to prove that the existing evaluation protocol, Deteval, was not optimised to cater the introduction of *Total-Text*. Fine tuning was done to refined the mentioned evaluation scheme to improve its accuracy in terms of benchmarking. Together with the Pascal VOC evaluation method, both were made the official evaluation protocols of *Total-Text*. Then, this work presented an in-depth exploration on the variation of curved text, hoping to provide useful insights for future curved text related research. Last but not least, an assisted annotation framework was introduced, namely *T3*. Experiments have showns that *T3* is capable of reducing the polygon annotation time by 25%.

The last contribution of this thesis work is the proposal of a straightforward yet effective scene text detection model. The proposed method, namely Polygon-FRCNN, was built based on a deep learning framework, and has demonstrated its ability in detecting text of all orientations in the conducted experiments.

1.4 Outline

This chapter discusses the background, motivation, problems, objectives and the contributions of this thesis. The outline of the rest of the chapters are as follows:

Chapter 2 reviews closely related literature works to this research work. Specifically, the works of existing scene text datasets and detection algorithms are detailed in this section. In particular, the general motivation behind renowned scene text datasets and how they have impacted the research scenes are discussed in a detail manner. Additionally, all the statistics and features of these datasets are presented to give a general perspective on the research scene.

Subsequently, the review of the evolution and differences between modern scene text

detectors also take place in this chapter. Following that, the strengths and weaknesses of each methods are explored to justify the rationale behind the choice of extending the Faster-RCNN object detection framework in the design of the proposed method.

Chapter 3 details the proposed scene text dataset, *Total-Text*. The chapter starts by introducing the differences between text of different orientations (i.e. horizontal, multi-oriented, and curved), followed by all related statistics and attributes of *Total-Text*. Then, this chapter reveals a new ground truth format to cater text of all orientations, and all the annotation details and workflows come next. Then, this chapter is further with the in-depth study on the variation of curved text, the fine-tuning process of the Deteval evaluation protocol, finishing with the introduction of *T3*, the assisted annotation method.

Chapter 4 proposes a scene text detection model that is capable of detecting text of all orientations, namely Polygon-FRCNN. This chapter starts by detailing every part of the proposed model. Following that, the implementation details: training process, model's hyperparameters, variant of Polygon-FRCNN are elaborated. The proposed method is then benchmarked on various dataset to justify its claimed ability. Both quantitative and qualitative results are presented at the end of this chapter.

Chapter 5 draws the conclusion from all the findings and analysis of this thesis work. The current limitation of the proposed dataset and detection model are listed along with the potential of future development.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The availability of datasets pushes a research field forward. Not only it defines the scope of problem for researchers, it allows them to benchmark and solidify claims of their algorithms. Furthermore, the recent trend of incorporating deep learning models into scene text understanding algorithms (T. He et al. (2016a); W. He et al. (2017); Liao et al. (2017); W. Liu et al. (2016); Y. Liu & Jin (2017); Ma et al. (2017); B. Shi et al. (2017); T. Wang et al. (2012); Yao et al. (2016); Zhang et al. (2016)) demands a large amount of data and ground-truths. The first part of this chapter details i) the motivation behind every renowned datasets, ii) comparison of their statistics, iii) attributes possess by each of them, and iv) the impact they have made on the research scene.

Previous attempts at the scene text detection problem can be broadly separated into two categories: Before CNN and After CNN, both of which will be discussed thoroughly in this chapter. On top of that, this section further emphasizes on how previous works address multi-oriented text detection problem, which is in nature closer to this study. Following that, this chapter ends by discussing and relating two of the limited works on the curved text detection problem to the proposal of this research.

2.2 Scene Text Detection Datasets

2.2.1 ICDAR2003-2013

The Internation Conference on Document Analysis and Recognition (ICDAR) is an international academic conference that advocates document analysis research. Apart from calling for the submission of conference papers, the conference committee also organizes the Robust Reading Competition. The competition includes challenges like text detection

in natural images, text recognition, text detection in video, etc. The ICDAR2003 dataset was collected for the first ever Robust Reading competition in 2003 (Lucas et al. (2003)). It has 509 camera taken scene text images, which is split into a training and testing set, with 258 and 251 images respectively. Several improvements were made to improve the quality of the dataset. For one, the ICDAR2011 dataset is the trimmed down version of ICDAR2003. Its total number of image was reduced to 484 to remove image duplication. Further improvements were made in the year 2013 (Karatzas et al. (2013)): the total number of image was again reduced, to 462, and pixel level ground-truth was added for the segmentation challenge. Ye & Doermann (2014)'s survey captured all of these changes in a comprehensive manner. The ICDAR2003, ICDAR2011, and ICDAR2013 datasets were the most popular datasets throughout the 2000s. They have a total of 1224 citations to-date (2018). Their popularity has indirectly influenced the design of many text detection systems. The fact that all the texts in these datasets are horizontally oriented has inspired researchers (X. Chen & Yuille (2004); Epshtein et al. (2010); Kim et al. (2003); J.-J. Lee et al. (2011); Neumann & Matas (2013); Pan et al. (2011); Yi & Tian (2012, 2013); X. Yin et al. (2012); X.-C. Yin et al. (2013)) to integrate such rule in their design. Several example images of ICDAR2013 can be seen in the first row of Figure 2.1.

2.2.2 MSRA-TD500

Several datasets (Nagy et al. (2011); Pan et al. (2010); Yi & Tian (2011); X.-C. Yin et al. (2015) were then introduced to address the lack of arbitrary-oriented text in the scene text datasets. MSRA-TD500 (Yao et al. (2012)) among all is the most popular dataset in this regard. It was used as a benchmark in many highly influenced multi-oriented[1] text detection works (Yao et al. (2014); X.-C. Yin et al. (2015, 2014b); Zhang

[1]There are two common notations for describing text in arbitrary orientation within the scene text understanding community: arbitrary-oriented (Jiang et al. (2017); Ma et al. (2017)) and multi-oriented (W. He et al. (2017); Y. Liu & Jin (2017); Tian et al. (2015); X.-C. Yin et al. (2014a) text. This thesis will

et al. (2016); Zhou et al. (2017)). MSRA-TD500 has 300 and 200 training and testing images respectively. A closer inspection on all the images of MSRA-TD500 revealed that most if not all the text instances are arranged in a slanted form as illustrated in the second row of Figure 2.1. As shown, the characters in these text instances are arranged in a straight line fashion, but with an offset to the horizontal line.

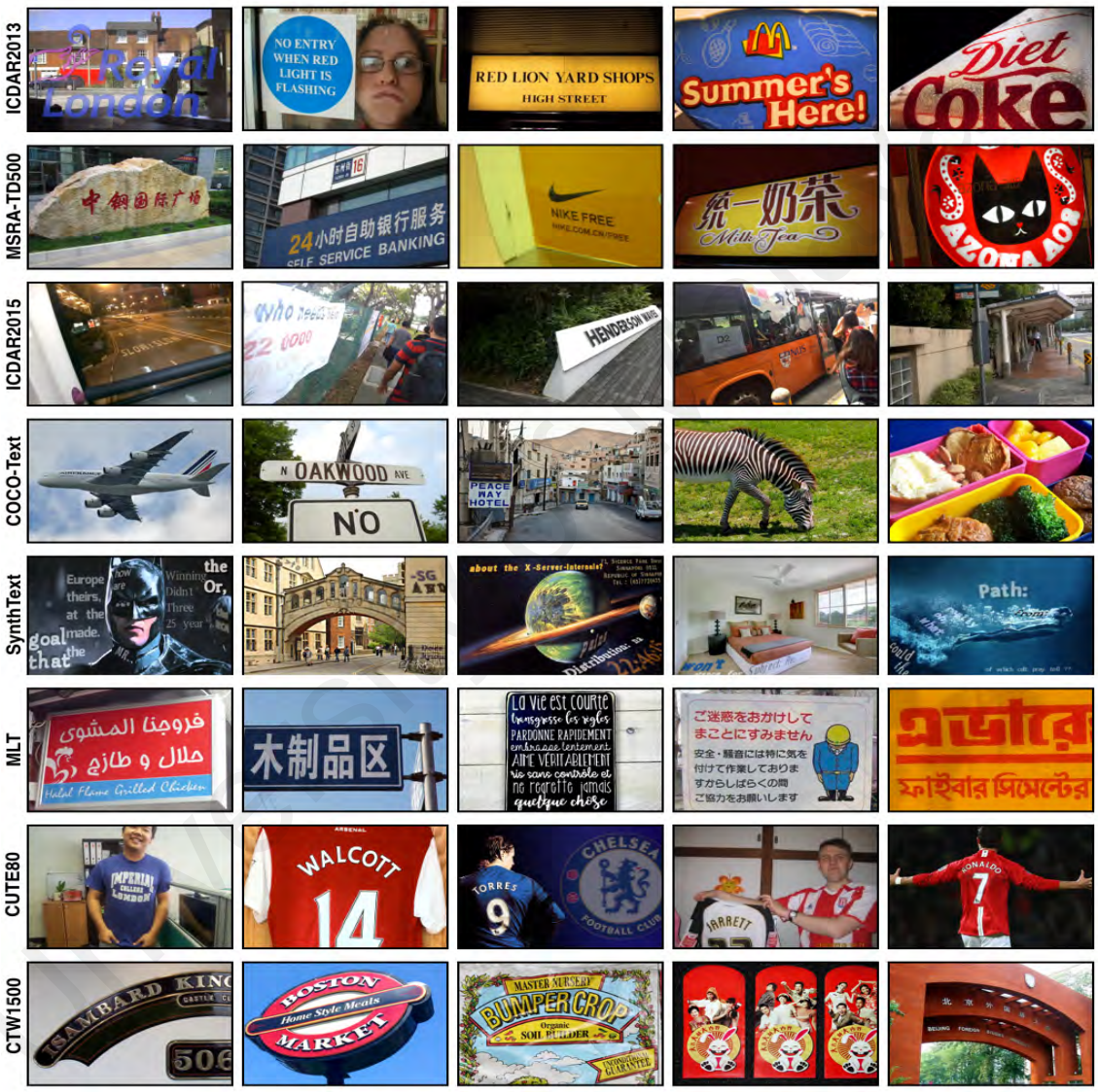


Figure 2.1: Images from various well known scene text datasets.

be proceeding with the notation of 'multi-oriented' in describing text in arbitrary orientation.

2.2.3 ICDAR2015

Over a decade of research efforts has turned the challenge of ICDAR2013 into a toy problem. In the ICDAR official website ICDAR (2018), the top-10 ranking algorithms reported the F-measure score of at least 0.9 (out of 1). Seeing the saturated performance reported on ICDAR2013, ICDAR launched a whole new dataset in 2015. The new challenge, namely 'Incidental Scene Text' (also commonly known as the ICDAR2015 dataset) was introduced with the intention of challenging the field with low quality scene text images. All of the 1670 images in the dataset were collected without prior effort in focusing on the text instances. As a result, most of the text instances are out-of-focused, blurry, and/or has major perspective distortion as illustrated in the third row of Figure 2.1. Such attributes are mainly motivated by the reason that a scene text detector can not always expect the text in natural image to be properly centered. It is a much needed new direction to lead scene text detection research to application ready level.

Both ICDAR2015 and MSRA-TD500 did not mention about the existence of curved text. Hence, an inspection was carried out on all the images of both datasets, which revealed that curved text only occupies less than 0.3 % of the total text instances in both datasets. Interestingly, one of the rare curved text instances in MSRA-TD500 (2nd row last column in Figure 2.1) is oftenly seen in multi-oriented text detection works (B. Shi et al. (2017); Xing et al. (2017); X.-C. Yin et al. (2015); Zhang et al. (2016)) as a failure example.

2.2.4 COCO-Text

COCO-Text (Andreas et al. (2016)) was released in the early 2016, which is the largest natural scene text dataset to-date with 63,686 images and 173,589 labeled text regions. COCO-Text is a subset of the larger MSCOCO dataset (Lin et al. (2014)). Since all the

images in COCO-Text were captured with no text in mind, almost half of the images have no text instances as depicted in the forth row of Figure 2.1. Apart from the common ground-truths such as spatial location and transcriptions, COCO-Text has even included: a) fine-grained classification between machine printed text and handwritten text, b) classification into legible and illegible text and c) script of the text. Workers from the Amazon Mechanical Turk platform[2] were hired to annotate such a large scale dataset.

2.2.5 SynthText

SynthText (Gupta et al. (2016)) is a synthetically generated scene text dataset with 800,000 scene text images. It was also intruduced in 2016, motivated by the demands of the deep learning methods (T. He et al. (2016a); W. He et al. (2017); Liao et al. (2017); W. Liu et al. (2016); Y. Liu & Jin (2017); Ma et al. (2017); B. Shi et al. (2017); T. Wang et al. (2012); Yao et al. (2016); Zhang et al. (2016)). Gupta et al. (2016) first obtained the text and image examples from the Newsgroup20 dataset (Karsch et al. (2011)) and Google Image Search respectively. The proposed generator pipeline starts by segmenting an image to different regions with color and texture cues. Text candidates are then placed into the proper image segment in order to retain image continueties (based on the observation that text in real images usually is contained in well defined region). Example images of SynthText can be found in the fifth row of Figure 2.1. The authors of this work also provided the said algorithm for researchers to generate their own data[3].

2.2.6 MLT

The MLT dataset (Nayef et al. (2017)) is the latest multi-script dataset that was collected for scene text detection, recognition, and script identification task. Both training and

[2] https://www.mturk.com/

[3] https://github.com/ankush-me/SynthText

testing set of it have 9000 images each, featuring nine languages (Chinese, Japanese, Korean, English, French, Arabic, Italian, German and Indian) with six different scripts. Sample images can be seen in the sixth row of Figure 2.1. The main motivation of this dataset is to challenge the community to design a scene text detector that is robust to different scripts that exist in today's world.

Given the scale of the COCO-Text, SynthText, and MLT datasets, the chances of them containing curved text is high. However, the factors that set *Total-Text* apart from these dataset are: a) curved text is a priority in *Total-Text*, occupying more than half the total text instances, rather than random occurence, and b) proper bounding region is employed as the detection ground-truth format. These factors make *Total-Text* a proper dataset to facilitate the currently under-researched curved text detection problem.

2.2.7 CUTE80

To the best of author's knowledge, CUTE80 (Risnumawan et al. (2014)) is the first scene text dataset that features curved text, which was introduced in 2014. In addition to the ground-truths for detection and recognition task, CUTE80 includes pixel-level ground-truth for segmentation task. The availability of it has inspired B. Shi et al. (2016) in solving text recognition problem with curved text images. Further, the enchanced version of polygon annotation format employed in *Total-Text*, which will be explained in Section 3.4.2 (d), is based on the one in CUTE80. Unfortunately, CUTE80 consists of only 80 images, which is too small for today's standard. Furthermore, its scenery diversity is low, most of them are football jersey, as illustrated in the seventh row of Figure 2.1.

2.2.8 CTW1500

CTW1500 (Yuliang et al. (2017)) is in principle the closest dataset to *Total-Text*. It was introduced to the community right after the publication of *Total-Text* (Chng & Chan (2017))

in late 2017. Both datasets are similar in terms of scale, the usage of polygon bounding region annotation, and was collected with at least one curved text per image. Despite having similar motivation, both datasets are different in a number of ways. First of all, *Total-Text* considers only latin scripts as text instances, while CTW1500 includes both the latin and chinese scripts in their annotation. Such difference leads to the second dissimilarity of both datasets, which is the granularity of annotation. *Total-Text*'s text instances were annotated at word-level, while CTW1500's were annotated at line-level. The third difference is the format of polygon bounding region between two. CTW1500's polygon is fixed to have fourteen vertices each while *Total-Text*'s is fixed to have ten vertices (as line-level text instances are usually bigger than word-level text instances). Lastly, CTW1500 is only annotated to facilitate the text detection task while *Total-Text*'s annotation consists of spatial location, transcription, orientation, and pixel-level mask for text detection, recognition, and segmentation tasks.

Table 2.1: Well known scene text detection datasets. Legends: D = Detection, S = Segmentation, R = Recognition, H = Horizontal, M = Multi-oriented, C = Curved.

Dataset	Image number (Train/Test)	Text instances	Language	Task	Granulality	Orientation	Year
ICDAR2013	462 (229 / 233)	1943	English	D, S, R	Word	H	2013
MSRA-TD500	500 (300 / 200)	1719	English, Chinese	D	Line	H, M [i]	2012
ICDAR2015	1670 (1000 / 670)	11886	English	D, R	Word	H, M [i]	2015
COCO-Text	63686 (43686 / 20000)	173589	English	D, R	Word	H, M [ii]	2016
SynthText	800000	8000000	English	D, R	Word	H, M [ii]	2016
MLT	18000 (9000 / 9000)	-	9 languages	D, I	Word	H, M [ii]	2017
CUTE80	80	-	English	D, S, R	Word	H, M, C	2014
CTW1500	1500 (1000/500)	10751	English, Chinese	D	Line	H, M, C	2017

[i] Curved text occupied less than 0.3 % of the total text instances.
[ii] The sheer number of these datasets make it difficult to obtain the exact number of curved text instances however, a quick look into a random subset of these datasets reveals that the occurence of curved text instance is very rare.

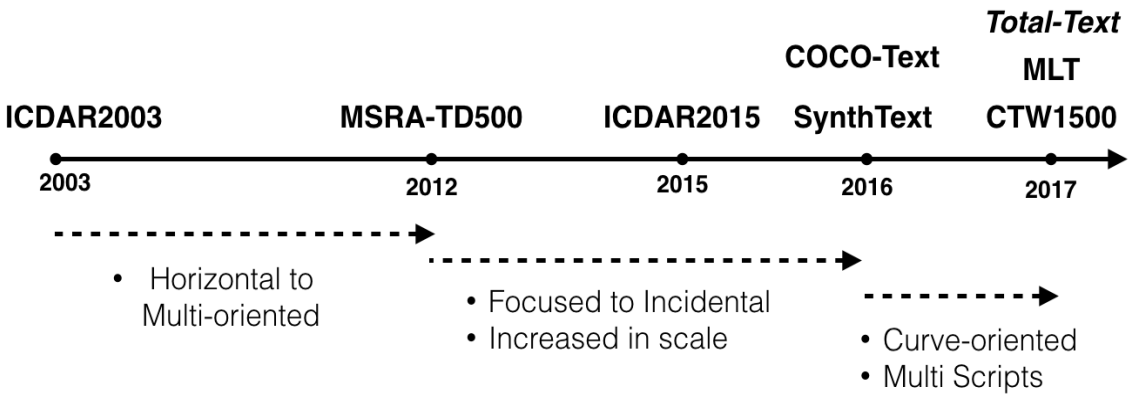


Figure 2.2: Datasets launching timeline.

2.3 Scene Text Detection Algorithms

2.3.1 Before CNN

Most of the earlier scene text detection works can be categorized into two main groups (Ye & Doermann (2014)): connected component analysis (CCA) and sliding window (SW) classification approach. CCA methods label subsets of connected components hueristically based on features such as color similarity and spatial location. Several works (Koo & Kim (2013); S. Lee et al. (2010); Pan et al. (2011)) employed statistical model to address the need of complex heuristic rules. Such approach enhances the adaptivity of CCA by a great magnitude. In contrast, SW methods classify every window in an image to either text or non-text patch in the feature space. Consequently, positive text patches are further grouped into text regions with CRF (K. Wang et al. (2011)), morphological operations (J.-J. Lee et al. (2011)), or graph methods (Coates et al. (2011); T. Wang et al. (2012)). While SW methods usually have simpler architecture, it is much more computationally expensive due to the need of classifying large numbers of windows.

Earlier feature extractors were mostly handcrafted. For instance, color (Garcia & Apostolidis (2000); S. Lee et al. (2010); X. Liu et al. (2008)), edge (R. Huang et al. (2013); Phan et al. (2012); Shivakumara et al. (2008)), texture (Li et al. (2000); Shiv-

akumara et al. (2010); Y. Zhong et al. (2000)), point (X. Zhao et al. (2011)), etc., were explored to represent text. All of these feature extractors were designed based on certain observation of text in natural scene images. For instance, Yi & Tian (2011) assumes that characters belong in the same text posses similar color, which is usually in contrast to the background color. On the other hand, Li et al. (2000) observed that text regions typically have different texture properties than the surrounding areas. However, no one assumption is broad enough to cover the complexity of real world sceneries. Case in point, multi-colored text or uneven lighting on the text region would have refuted the uniform color assumption; text with sparse characters on the other hand would not posses the texture pattern expected by Li et al. (2000).

Maximally Stable Extremal Regions (MSER) (Matas et al. (2004)) and Stroke Width Transform (SWT) (Epshtein et al. (2010)) are two sophisticated feature extractors based on pixel intensity and edge respectively. Both works have inspired many other algorithms (Bai et al. (2012); H. Chen et al. (2011); Dinh et al. (2007); Koo & Kim (2013); Mosleh et al. (2012); Neumann & Matas (2011); Nistér & Stewénius (2008); C. Shi et al. (2013); Y. Zhao et al. (2011)) in designing better scene text detector. The MSER algorithm is capable of detecting stable color region (also known as blob detection) in an adaptive manner. Regions with similar threshold resistance range will be grouped as one component. On the other hand, the SWT feature extractor was designed based on the observation that characters in a text have similar stroke width. The internal distance between the opposite edges of a character is defined as stroke width. Scene text detectors based on these two feature extractors had saturated the top part of the performance ranking table of datasets such as ICDAR2013 and MSRA-TD500, bringing the handcrafted feature extractor era to its peak moment. Figure 2.3 depicts the example of different feature spaces represented by different feature extraction techniques.

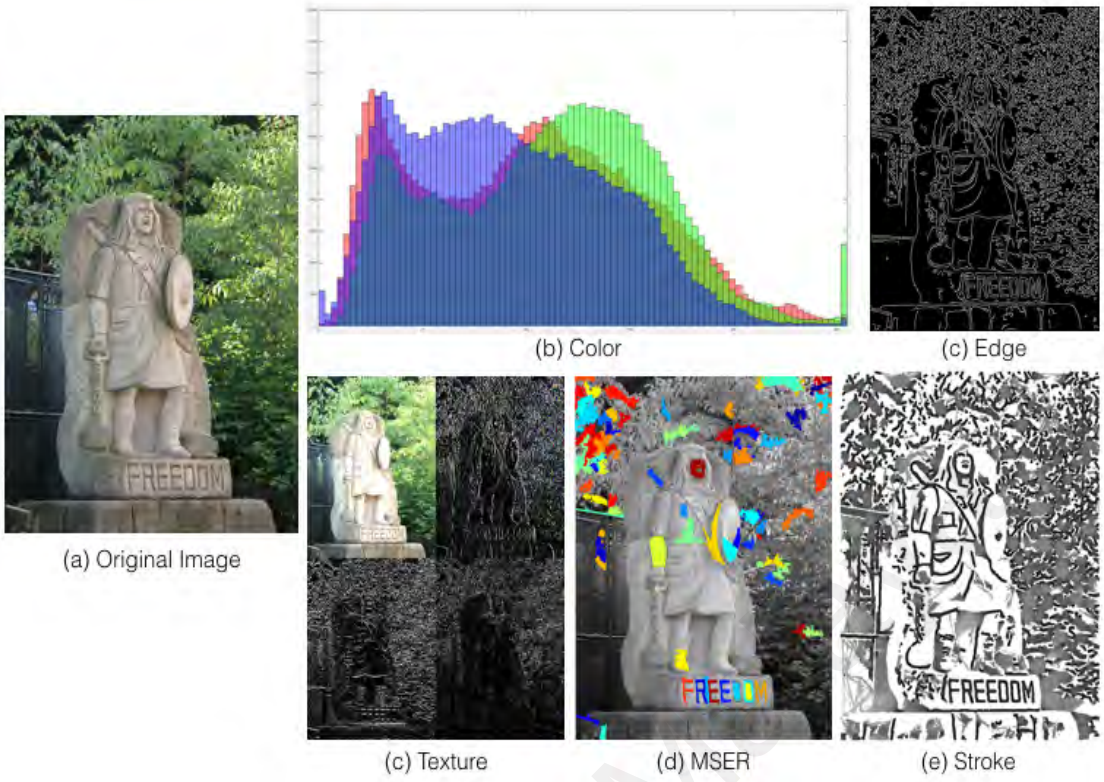


Figure 2.3: Example of different feature extractors used to represent text candidates.

2.3.2 After CNN

CNN is one of the ANN models that has shown promising potential in solving the computer vision problems. Its main difference with ANN is that instead of connecting every neuron between previous layer and current layer, it only connects a subset of it (usually adjacently connected neurons, visual illustration can be seen in Figure 2.4). Such model was first proposed in (Fukushima (1988)), which was inspired by the observation that neurons in the visual cortex only respond to local features like edges and lines; while cells in the area higher than visual cortex are found to repond to more complicated patterns like circles, rectangular, and even a human face. Figure 2.5 depicts one of the earliest models of modern CNN, LeNet-5 by LeCun et al. (1998), which is made up of several fundamental operations: i) convolution, ii) subsampling[4], and iii) fully connected layers. The front part of the CNN is usually layers of convolution and pooling, which function

[4]Better known as pooling today

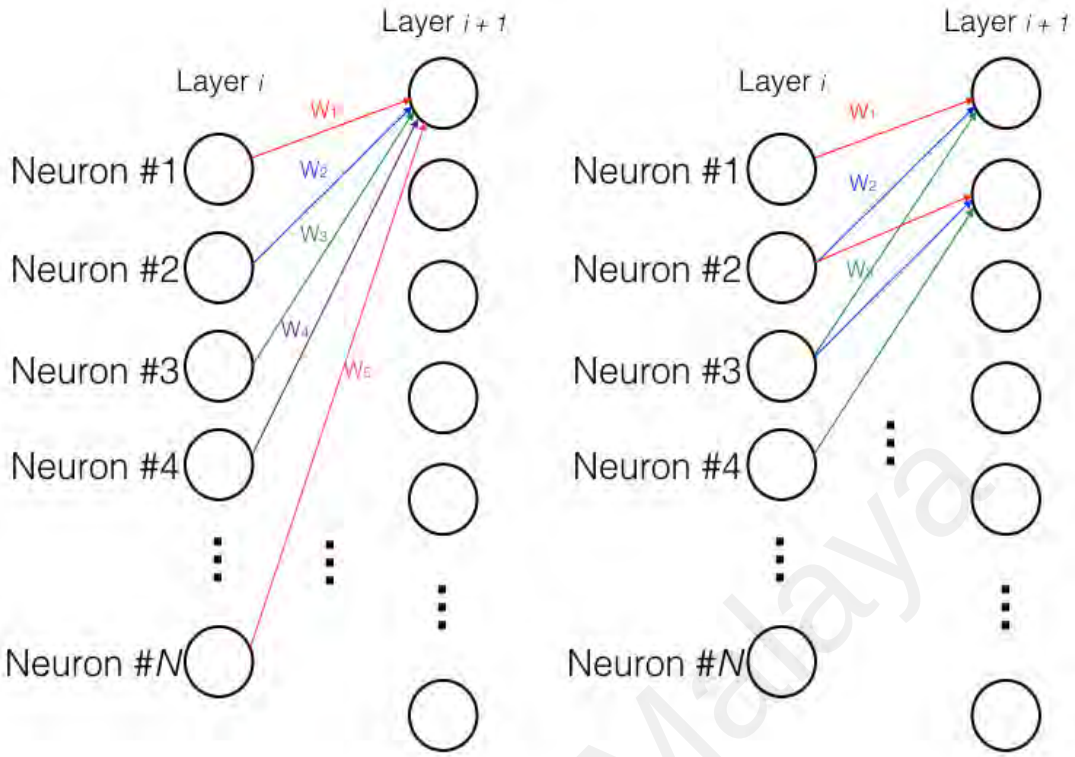


Figure 2.4: Difference between ANN(left) and CNN(right) in terms of neurons connection.

as a *feature extractor*. There are differences between the feature maps in different layers of the network. The earlier[5] feature maps constitute of local features such as edges, color blob, texture since each neuron in these layers is only exposed to a small region of the input image. As the layer goes deeper (or higher), the neurons in the feature maps are exposed to a larger area of the input image, hence result in higher level features like shapes, patterns, and sometimes abstract that is incomprehensible by human (Zeiler & Fergus (2014)). Upon transforming the input image to a high dimension feature space (multiple feature maps), the fully connected layers then *classify* which classes does it belongs to. In the case of LeNet-5, it was designed to recognize digits, which has 10 classes (0-9), the output is a 10-channel vector of scores ranging from 0-1. The class with the highest score determines the digit identity, for instance, the output vector of *[0.0, 0.1, 0.8, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]* gives the digit output of '2'. The variables (filter kernels) of

[5]Or lower, shallower, these are all the accepted convention in the community.

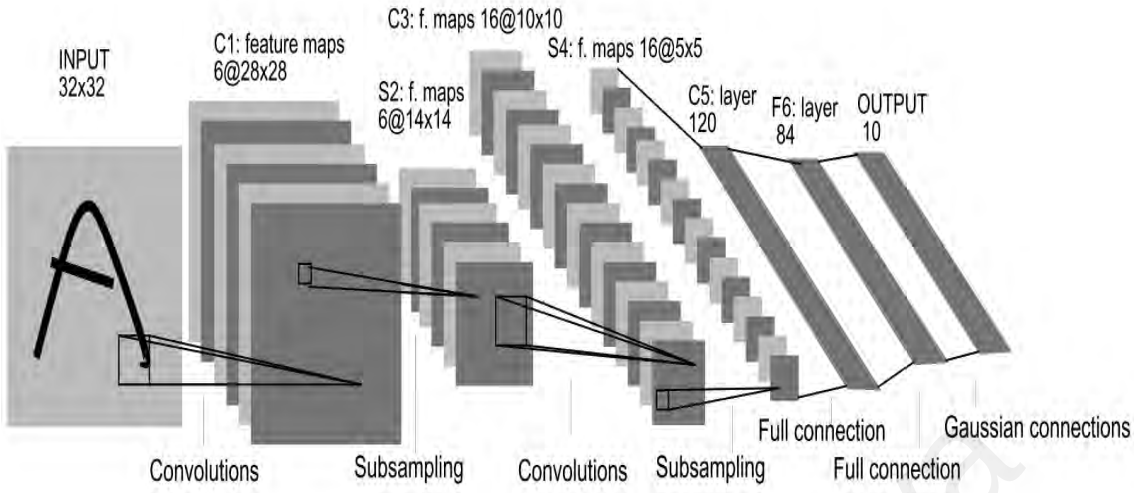


Figure 2.5: One of the earliest models that defines the general structure of CNN today – LeNet-5.

the network is randomly initialized at first, hence the prediction started out with random guess, and get better as the training proceeds.

The training of a CNN is an optimization problem. Stochastic gradient descent is one of the proven methods that works well in training a CNN (Krizhevsky et al. (2012); LeCun et al. (1998); Szegedy et al. (2015)). Generally, the training process can be described as follows: i) the network receives an input image and makes its prediction, ii) the prediction is being compared with the associated ground-truth, iii) the *loss* is calculated and being *backpropogated* through the network. After each iteration, the weights of the network are supposed to move closer to the global optima (Kiefer & Wolfowitz (1952)), and hence results in better decision making. Such end-to-end (feature extraction to classifier) training scheme allows CNN to learn the best set of filters and classifier from the real data in an unbiased manner, which is a really tough challenge for human feature extractor engineers.

Deep CNN models[6] caught computer vision field's attention when AlexNet (Krizhevsky et al. (2012)) outperformed all other non deep learning based models by a large margin

[6] 'Deep' as in many layers, a convention in the deep learning community.

in the ImageNet-2012 classification competition (ILSVRC-2012). Since then, deep CNN models' footprint can be seen in many problems like video classification (Karpathy et al. (2014)) and detection (Hou et al. (2017)), image captioning (Xu et al. (2015)), instance segmentation (Long et al. (2015)), kangaroo detection (Saleh et al. (2016)), etc. T. Wang et al. (2012) and Jaderberg et al. (2014) are two of the earliest works that incorporated deep CNN model in the scene text understanding problem. T. Wang et al. (2012) trained a 2-layer[7] CNN to classify every single patch of an input image into either text or non-text class. Subsequently, image region with 'high' response (classified as text) will be processed and fed into the recognition module (which has similar network arcthitecture as the detection module) to further classify into one of the 62-class (26 for low-capital letters, 26 for high-capital letters, and 10 for digits). Roughly 60k image patches (32×32 in size) from ICDAR2003 and Char74k were used as the training data in the training process. On top of the text/non-text and 62-class classification tasks, Jaderberg et al. (2014) added two more training tasks, which are a) 37-class of case-insensitve characters, and b) 604-class of bigrams (pairing of characters). The authors argued that these added supervisions will increase the performance of both detection and recognition ability. Larger amount of training data was used in the training process: a total of 186K of image patches (24×24 in size) cropped from all the ICDAR variants, KAIST, Chars74k, StanfordSynth, and FlickrType constituted the training dataset. Both T. Wang et al. (2012) and Jaderberg et al. (2014)'s model achieved state-of-the-arts result, with 83.9% and 86.8% respectively in the character recognition task on the ICDAR2003 dataset, outperforming previous work (Coates et al. (2011)) by at least 2.2%. Inspired by these works, T. He et al. (2016b); W. Huang et al. (2014); Zhang et al. (2015) also leveraged on the discriminative power of CNN as a classifier to differentiate text and non-text components.

Over the years, researchers have found out several methods to improve the perfor-

[7]Only convolutional layers are counted in the convention of CNN

mance of CNN. For instance, increasing the number of layers (Simonyan & Zisserman (2014)), inventing newer architecture like ResNet (K. He et al. (2016)) and GoogLeNet (Szegedy et al. (2015)), etc. Several groundbreaking architectures like Fully Convolutional Network (FCN) (Long et al. (2015)) and region-based CNN (R-CNN (Girshick et al. (2014)), Fast-RCNN (Girshick (2015)), and Faster-RCNN (Ren et al. (2015))) had caused an unprecedented improvement in the instance segmentation and object detection challenge respectively. Since then, many state-of-the-art works in other domains such as scene text detection (Liao et al. (2017)), biomedical image segmentation (Ronneberger et al. (2015)), vehicle detection (Fan et al. (2016)), etc., have integrated such architectures in their solutions. Consequently, most of the modern scene text detectors can be categorized into three main groups: segmentation-based, proposal-based, and single-network-based solutions, which will be discussed as follows.

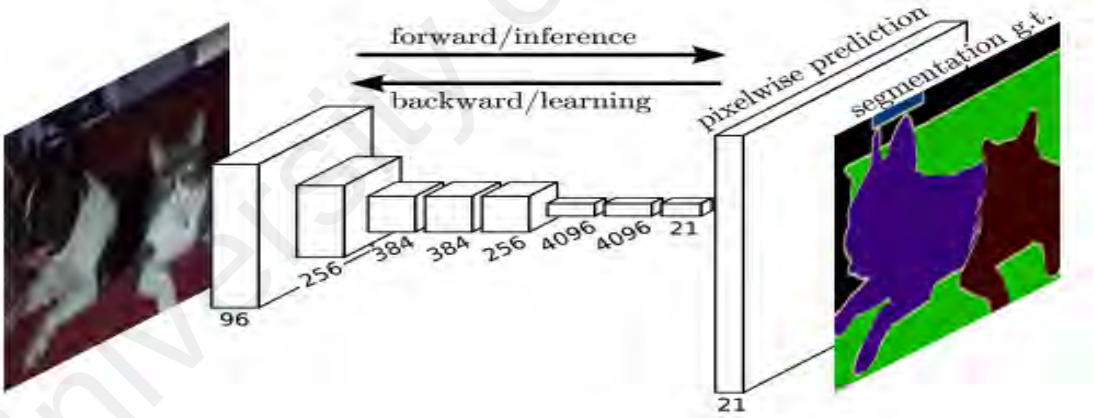


Figure 2.6: The FCN architecture. It is trained to label every pixel of the input image with their corresponding class.

2.3.2 (a) Segmentation-Based Solutions

Segmentation-based approaches are largely encouraged by the efficiency of FCN in making dense level pixel prediction. Figure 2.6 is an illustration of the well-known FCN artchitecture. Long et al. (2015) replaced the conventional fully connected layer with

convolutional predictor, in turn allows the network to process arbitrary-sized input image in one go. Additionally, it has demonstrated superior performance on the instance segmentation problem. The fine detail of the output saliency map (as illustrated in Figure 2.6) has apparent benefit for the detection task. Back to the scene text community, Zhang et al. (2016) argued that by utilizing CNN as a character detector, Jaderberg et al. (2014); T. Wang et al. (2012)'s models have restricted CNN's potential due to the local nature of characters. Hence, Zhang et al. (2016) proposed to use FCN to detect text region as a whole instead. In comparison to previous works (Jaderberg et al. (2014); T. Wang et al. (2012)) which treated every patch in an image individualy, Zhang et al. (2016)'s FCN was trained to utilise both *global and local cues* of a text region in the inference process. Specifically, skip layers (Long et al. (2015)) were used to combine both high and low level feature maps, which corresponds to global and local feature. Figure 2.7 depicts the difference between the saliency map generated by T. Wang et al. (2012) and Zhang et al. (2016)'s approach, which has demonstrated the effectiveness of the latter. Several other works (T. He et al. (2016a); Polzounov et al. (2017); Yao et al. (2016); Zhou et al. (2017)) followed trail and found success in developing high-performing scene text detectors. For example, T. He et al. (2016a) cascaded two FCNs to i) first approximate text region in an image, ii) then cropped the approximate text region and further segment it with higher precision. On the other hand, Yao et al. (2016) trained a FCN to infer three different text properties: text region, character region, and linking orientation maps. Subsequently, characters within the same text region will be arranged into a graph, which will then be further processed by Delaunay triangulation. Characters remain connected after the trimming will be treated as the final detection result.

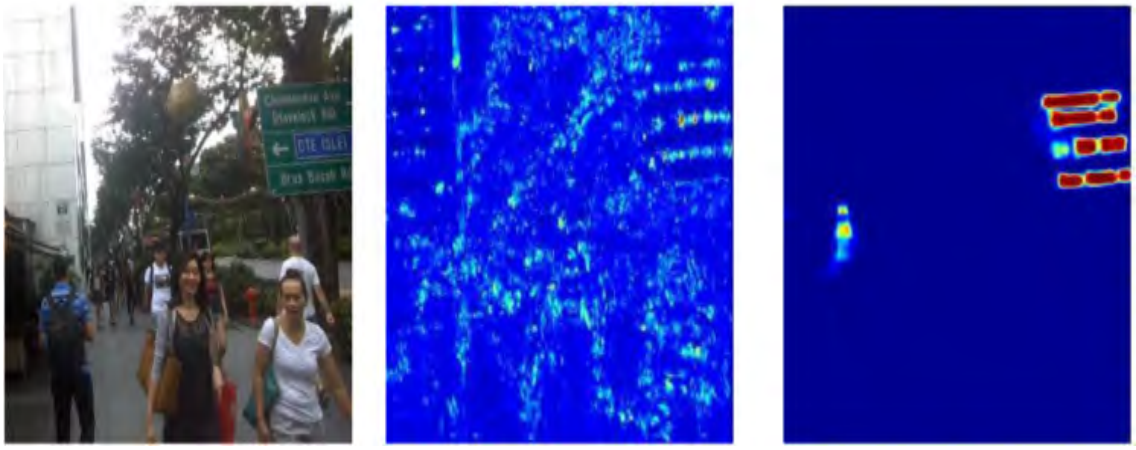


Figure 2.7: Saliency map generated by T. Wang et al. (2012)(mid) and Zhang et al. (2016)(right).

2.3.2 (b) Proposal-Based Solutions

R-CNN (Girshick et al. (2014)), Fast-RCNN (Girshick (2015)), and Faster-RCNN (Ren et al. (2015)) are a series of groundbreaking works for object detection. The core idea is simple – run classification over a large number of probable regions that might contain object of interest with CNN. In the first work (Girshick et al. (2014)), SelectiveSearch (Uijlings et al. (2013)) was used to generate roughly 2K proposals (image patches), which were then fed individually into a CNN to i) classify whether it contains object of interest, and ii) regress the bounding box location of the object. This novelty framework outperformed its counterparts by a large margin[8] on the PASCAL VOC 2010 dataset at the time of publication. However, it has slow execution time – 47s to run an image. Girshick (2015) solved the issue by sharing the feature extraction part of the CNN. Instead of running every proposal through the entire network, Fast-RCNN framework postpones the proposal cropping process to post feature extraction. As a result, the large number of proposals only have to go through the fully-connected layers (prediction layers), which is the last several layers of the entire network. With this, the test time per image is reduced to 2s, which is a significant improvement, but still no real-time speed. Motivated by this,

[8] 13.7% against the then-leading-performer, SegDPM (Fidler et al. (2013)) on the Pascal VOC 2010 test set

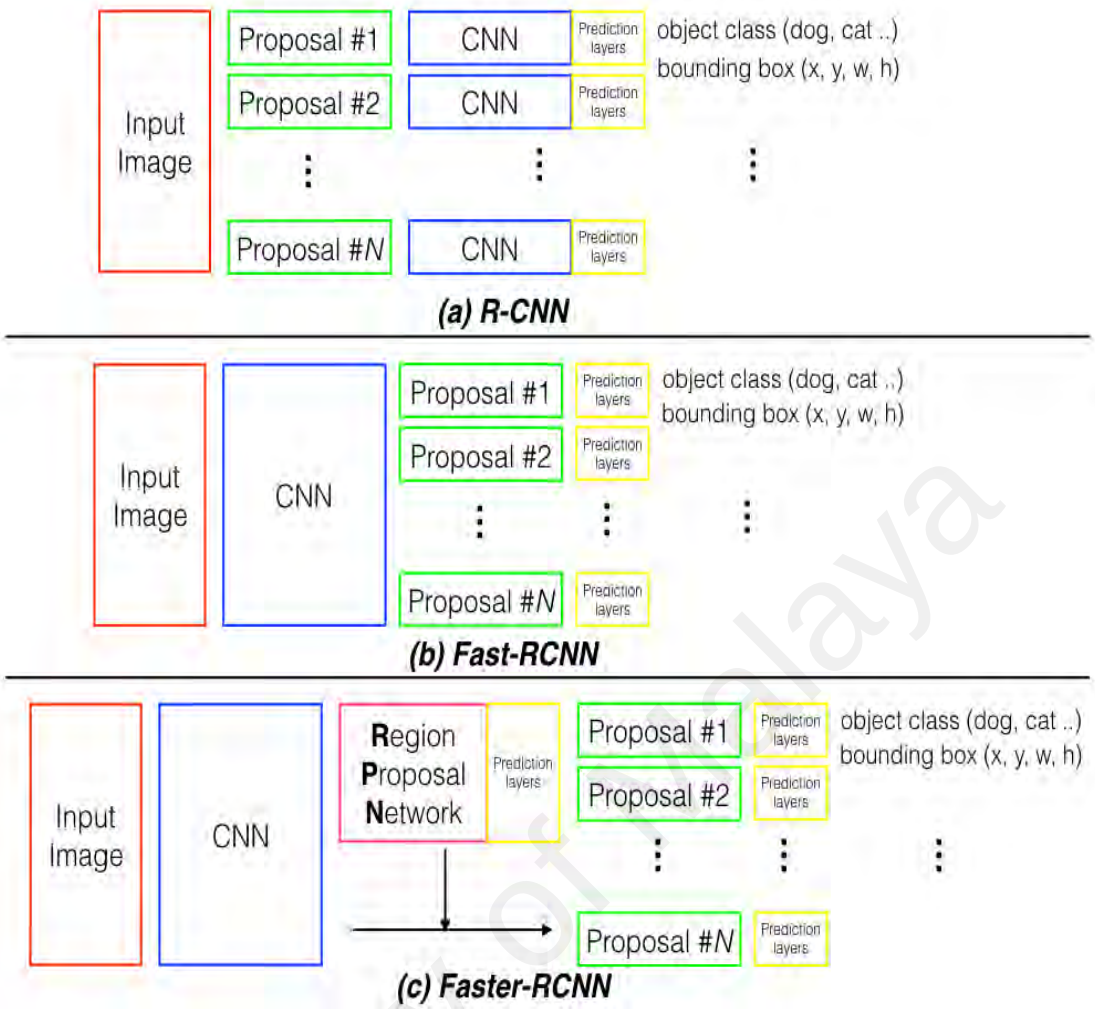


Figure 2.8: The block modules of the R-CNN, Fast-RCNN, and Faster-RCNN framework.

Ren et al. (2015) aimed to optimize the entire pipeline by solving the last bottleneck - the proposal module. Ren et al. (2015) introduced the Region Proposal Network (RPN) to replace the SelectiveSearch proposal module used by its predecessors. Figure 2.9 visualizes the design of RPN and anchor boxes. Anchor boxes were designed to ease the learning of the regression task. Instead of regressing the exact values of *(x, y, w, h)*, the regression layer is expected to regress the offset values to a series of predefined anchor boxes locations. Intuitively, the regression head is tasked to adjust the precision of the location of the anchor boxes. Additionally, the multi-scale aspect of anchor boxes helps in detecting objects with various scales and aspect ratios. By having the proposal module integrated as part of the network, Faster-RCNN is now end-to-end trainable and achieved an even

faster testing speed – 0.2 s per image, yet maintained the same level of performance with Fast-RCNN.

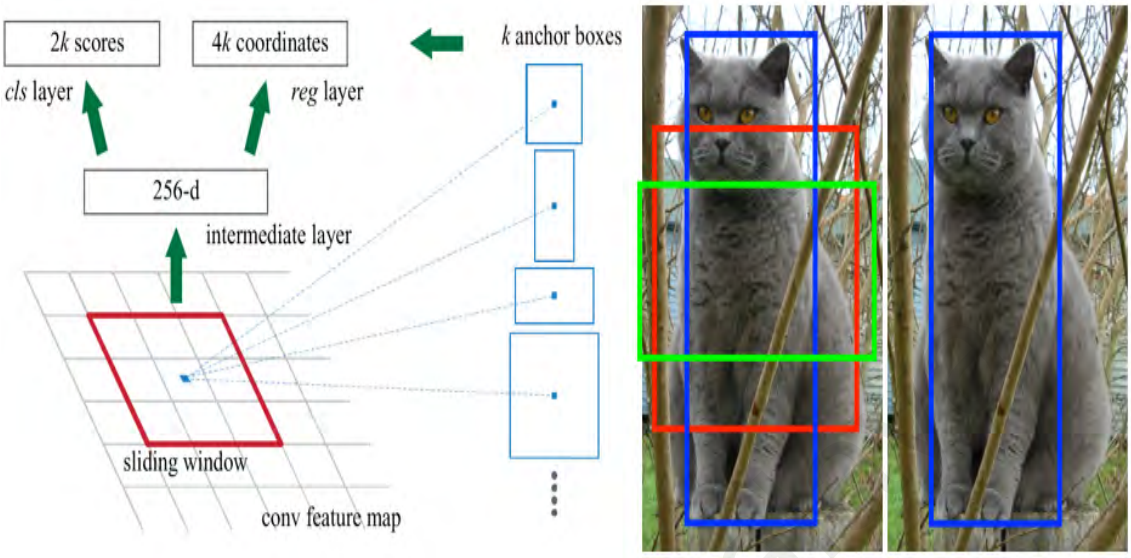


Figure 2.9: Left: Architecture of Region Proposal Network (RPN). Middle: Visualisation of anchor boxes in a real image. Right: The anchor box is too narrow for the object, hence, the regression head is expected to predict an offset of [0, 0, 0.125, 0] *(x, y, w, h)* to increase the width of the box for the final detection result.

The key aspects that contributed to the success of Faster-RCNN are RPN, anchor boxes, and the end-to-end training scheme from feature extraction to region proposal and finally the predictor. Encouraged by the success, Jiang et al. (2017); Ma et al. (2017); Z. Zhong et al. (2017) built their scene text detectors on top of the Faster-RCNN framework. Jiang et al. (2017)'s model, R^2CNN, was designed with an extra regression head to infer inclined box coordinates. The new inclined box coordinates (x_1, y_1, x_2, y_2, h) were designed to overcome the shortcoming of an axis-aligned box (x, y, w, h) in detecting multi-oriented text. Meanwhile, Ma et al. (2017) argued that axis-aligned anchor boxes are not the best fit for scene text that appear in wild orientations. They trained their Rotational-RPN (RPN) to propose rotational bounding box, which is essentially axis-aligned box with an extra bit for the angle information – (x, y, w, h, θ). Figure 2.10 illustrates the difference between axis-aligned bounding box and rotational bounding box (which also applies to the inclined box coordinates in R^2CNN). Z. Zhong et al. (2017)

also attempted to improve the RPN module by incorporating the inception Szegedy et al. (2015) design element. Such design was motivated by the attribute of inception module that is capable of involving both local and global context during the feature extraction process.

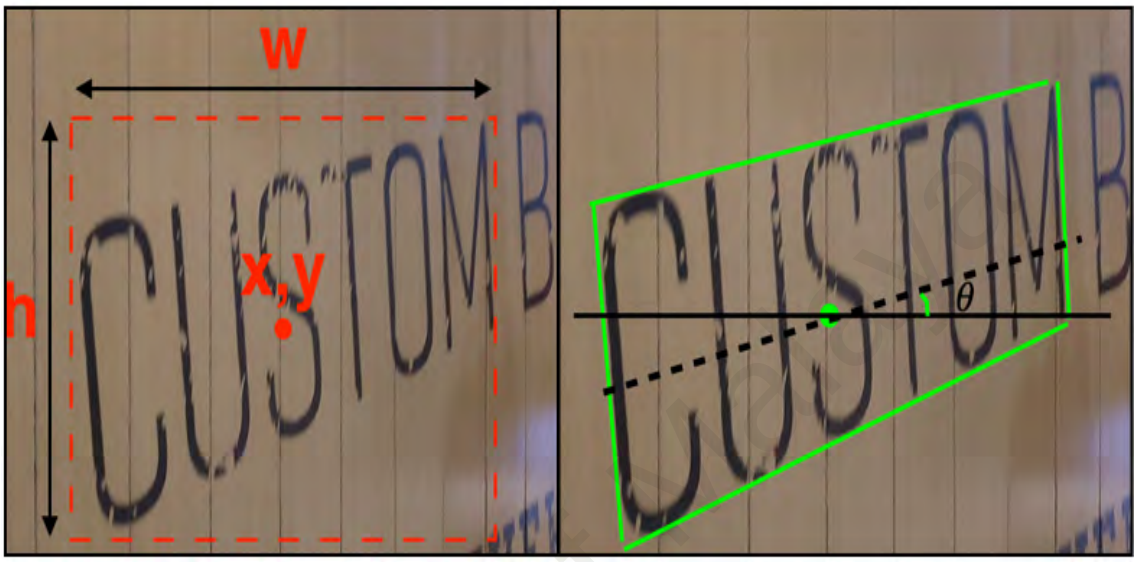


Figure 2.10: In comparison to the encoding of axis-aligned bounding box on the left *(x, y, w, h)*, rotational bounding box has an extra bit of information, θ, which is the angle offset between the text line and the horizontal axis.

2.3.2 (c) Single-Network-Based Solutions

SSD (W. Liu et al. (2016)) and YOLO (Redmon et al. (2016)) on the other hand have no proposal module in their design. It is capable of running at 59 and 122 FPS respectively, the fastest kind of object detection framework we have today. Essentially, such model aims to regress the location of the object by just looking at the input image once. Despite having only single backbone network, both models have their unique design : i) SSD framework has multiple regression heads from feature maps at different levels, while ii) YOLO only has one regression head at the final feature map layer. Based on the observation that text in natural scene could posses various scales, Liao et al. (2017) chose SSD (as illustrated in Figure 2.11) as the base framework to work with. Liao et al. (2017)

argued that the original SSD framework failed to detect text in natural scene as the default boxes (similar to anchor boxes in the Faster-RCNN framework) were designed with generic objects (cars, animal, human, etc.) in mind. Text in natural scene has wider aspect ratio (in other word, longer) than regular objects, hence Liao et al. (2017) integrated a 'text-box' layer on top of the generic SSD, with both i) redesigned default boxes and ii) inception-inspired convolution filters (1×5 in dimensions) to better fit the *longer shape* of text. Figure 2.12 illustrates the difference between the generic SSD's default boxes and TextBoxes's. Upon proving the concept of 'text-box', Liao et al. (2018) extended the original prediction layer with a 5-channel regression head for rotational bounding box (similar to R^2CNN). On a related note, B. Shi et al. (2017)'s model, namely Seglink, is also built on top of the SSD framework. Seglink is different with all other top-down approaches which predict the bounding box location in a direct fashion. Instead, it adopts a bottom-up approach, by modeling the network to regress the location of every character and their linking relationship. Both information are then processed by heuristic rules to form the final word-level detection output. All of these works benefited from the principle of SSD which utilize multi-level feature maps for their own respective regression problems.

2.3.3 Multi-oriented Text Detection Problem

Text detection solutions with multi-oriented text in consideration was rare before the introduction of MSRA-TD500 in 2012. In the past, the main challenge of multi-oriented text has always been the design of complex hueristics in grouping character candidates into a text-line candidates (or word). Most of the works (Epshtein et al. (2010); Kim et al. (2003); Liao et al. (2017); Neumann & Matas (2013); Pan et al. (2011); Yi & Tian (2012, 2013); X. Yin et al. (2012); X.-C. Yin et al. (2013)) were designed with the assumption of text are horizontal oriented. X.-C. Yin et al. (2015)'s algorithm was one of the advanced

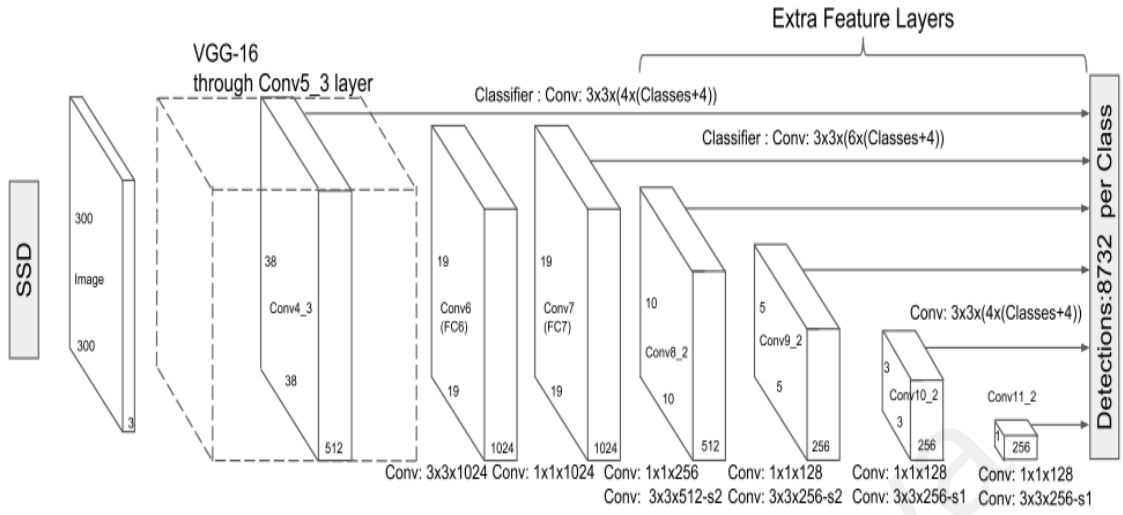


Figure 2.11: The architectural design of Single Shot Multibox Detector (SSD).

Figure 2.12: Left: Default boxes of SSD, which has the aspect ratios of [1, 2, 3, 1/2, 1/3]. Right: Default boxes of the Textboxes, which has the aspect ratios of [1, 2, 3, 5, 7, 10].

multi-oriented text detection works in prior to the CNN era. The work adopted a coarse-to-fine approach in forming text-line candidates as illustrated in Figure 2.13. Firstly, it groups character candidates into clusters based on their color, stroke width, and compactness similarity. Then, the character pairs with similar orientations will be grouped into finer clusters. Lastly, text-line candidates with different orientations will be separated and produce the final detection outputs. Zhang et al. (2016)'s solution is another work that has

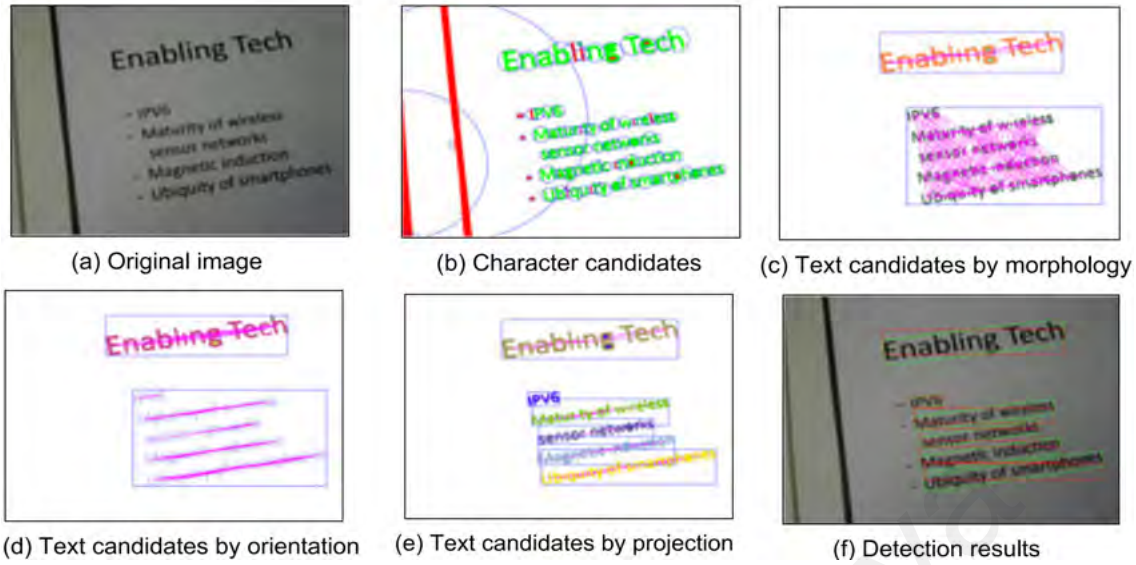


Figure 2.13: Overall pipeline of X.-C. Yin et al. (2015)'s approach. The multi-stage text-line generation process starts from (c) to (e).

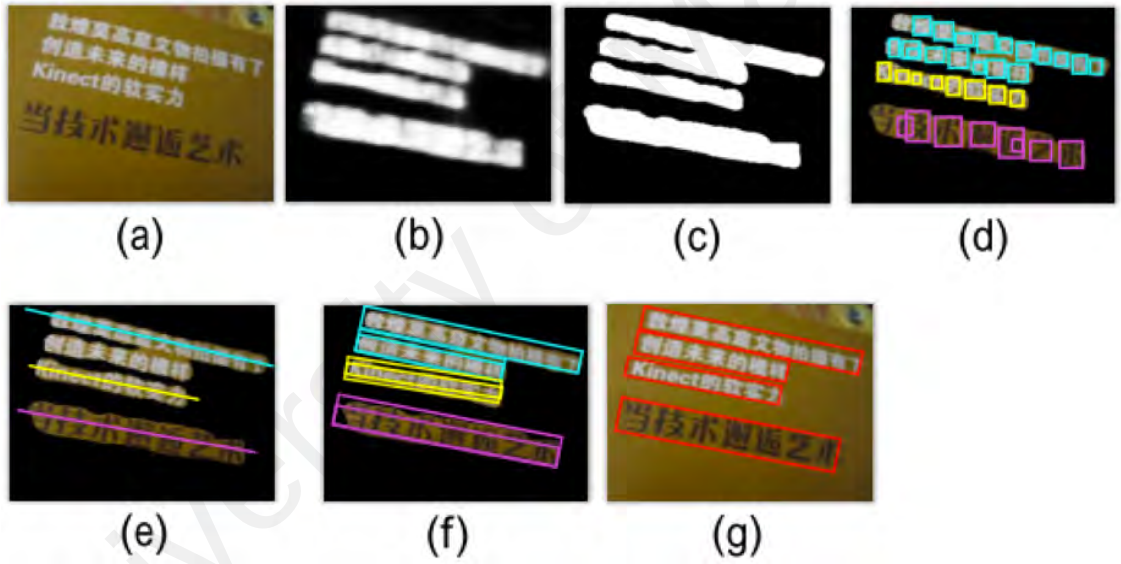


Figure 2.14: Overall pipeline of Zhang et al. (2016)'s scene text detector.

similar rule-based approach in spotting multi-oriented text line candidates. Upon extracting text regions and character candidates with FCN and MSER respectively, the algorithm determines the best angle offset, θ, that will hit the most character blocks within the same text region. As illustrated in Figure 2.14(c) (the connecting top two lines), the spotted text region could potentially contains several text line candidates. The algorithm then use the geometrical information of each character blocks to separate them into different clusters, indicating the potential of multiple text-line candidates (Figure 2.14(d-f)). Another FCN

was employed to remove false positives among these candidates for the final detection stage. The entire pipeline can be seen in Figure 2.14. Such rule-based approaches suffer from three major drawbacks: i) complicated heuristic designs, ii) aggregrated errors from multiple stages, and iii) restricted circumstances coverage (most searching or filtering algorithms assume characters within the same word have unified orientation, which is fine for most text instances but curved text, more details in Section 3.2). These shortcomings are the reasons why they are hard to be adapted for future work.

The top-down approaches are well-received by the community today. Vast majority of the proposal-based and single-network-based solutions directly predict the location of text regions as the detection output. This line of works formulate the detection task as a regression problem, expecting their networks to learn how to regress the encoded bounding box *(x, y, w, h)* from extracted feature maps. *SmoothL1* loss function, as described in Equation 2.1, is the most popular loss function for such model. Intuitively, it calculates the difference between the predicted output (encoded axis-aligned bounding box in this case) and ground-truth, in order to determine the amount of changes the network needs to make for upcoming output predictions.

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} smooth_{L_1}(t_i^u - v_i) \tag{2.1}$$

in which,

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \quad (2.2) \\ |x| - 0.5 & \text{otherwise,} \quad (2.3) \end{cases}$$

As discussed, the conventional axis-aligned bounding box format used in the vanilla

Faster-RCNN and SSD does not fit the shape of multi-oriented text[9]. Hence, a straightforward solution to that would be training the network to regress either rotational bounding box or quadrilateral bounding region that could fit multi-oriented text as visualized in Figure 2.10(right). W. He et al. (2017); Jiang et al. (2017); Liao et al. (2018); Y. Liu & Jin (2017); Ma et al. (2017); Xing et al. (2017); Zhou et al. (2017)'s works can all broadly grouped into this category. Table 2.2 is a summary of different encodings that recent works trained their model to learn. All of the aforementioned works have demonstrated the potential of scaling up either Faster-RCNN or SSD kind of framework to regress more parameters than just the conventional *(x, y, w, h)*. Apart from that, the direct text region spotting mechanism eliminated the needs of heuristic designs and multiple stage of processess, which are usually error prone. Attracted by these attributes, the proposed method of this work is based on the Faster-RCNN framework. Specifically, the effort is focused on the limitation of the existing axis-aligned bounding box, rotational bounding box, and quadrilateral bounding region in detecting curved text instances.

Table 2.2: Recent multi-oriented text detectors with their framework and regression target detail.

Method	Based Framework	Regression Target
Direct Regression (W. He et al. (2017))	FCN	w_1, h_1, w_2, h_2, w_3, h_3, w_4, h_4
EAST (Zhou et al. (2017))	FCN	x, y, w, h, θ x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4
RRPN (Ma et al. (2017))	Faster-RCNN	x, y, w, h, θ
R^2CNN (Jiang et al. (2017))	Faster-RCNN	x_1, y_1, x_2, y_2, h
DMPNet (Y. Liu & Jin (2017))	SSD	x, y, w_1, h_1, w_2, h_2, w_3, h_3, w_4, h_4
Textboxes++ (Liao et al. (2018))	SSD	x_1, y_1, x_2, y_2, h

[9]Based on the groundtruth format annotated in MSRA-TD500 and ICDAR2015

2.3.4 Curved Text Detection Problem

Text detection algorithm with curved text in consideration is rarely seen. Risnumawan et al. (2014)'s system is one of the unconventional cases. Like many other scene text detectors before the CNN era, Risnumawan et al. (2014) designed a symmetry-based feature extraction technique to spot for character candidates in natural images. The novelty of this system is the design of an eclipse growing technique that is capable of connecting neighbouring character candidates with no orientation restictions. Figure 2.15 illustrates two of the grouping examples. The eclipse growing process starts at a random pixel in the feature space, and search through all adjacent pixels for spotted character candidates (obtained in previous stages). The expansion will be terminated when there is no more character candidate in the nearby pixel. The algorithm has demonstrated its effectiveness on the CUTE80 dataset with several notable rooms for improvement: i) the entire system consists of many complicated hueristics (to extract features, generate text-line candidates, and false positives removal) which are prone to errors, ii) rotated bounding box output format which does not fit the curved text instances tightly, and iii) slow execution speed – 16.1s per image.

In contrast, Yuliang et al. (2017)'s CTD (Curved Text Detector) is one of the modern scene text detectors that was designed with curved text in consideration. It was built on top the Faster-RCNN architecture, trained to regress *polygon bounding region* in order to detect and bind text of all orientations tightly. The author emprically found that polygon with fourteen pairs of vertices is sufficient to cover most *line-level text instances* in natural images, hence extended the regression head of CTD to have thirty-two parameters (for x_{min}, y_{min}, x_{max}, y_{max}, w_1, h_1, ... w_{14}, h_{14}). In addition, the authors argued that the relationship between each polygon vertices is sequentially related, hence, employed a variant of Recurrent Neural Network (RNN) – Bidirectional Long Short Term Memory

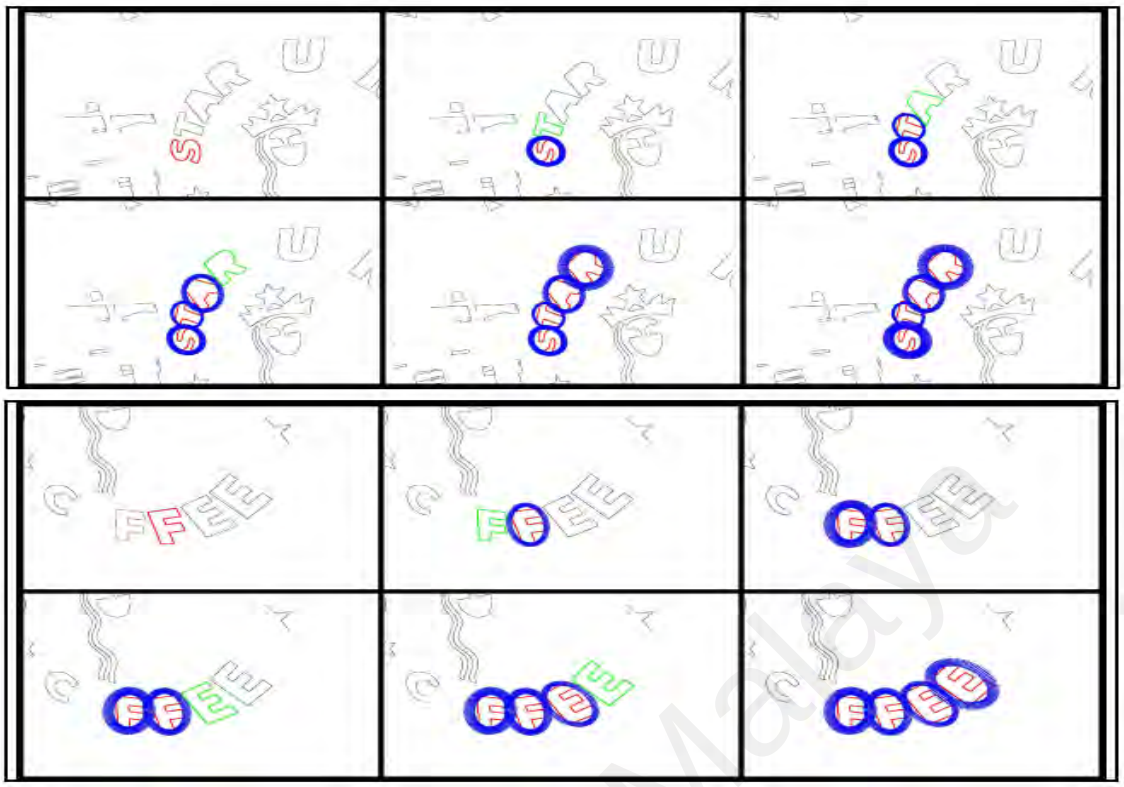


Figure 2.15: Examples of the eclipse growing scheme employed in Risnumawan et al. (2014)'s work.

(BLSTM) to model the relationship between each of them. Meanwhile, the work followed the convention of Faster-RCNN (Ren et al. (2015)) and DMPNet (Y. Liu & Jin (2017)), agreed that relative information (i.e., the *w*idth and *h*eight difference between each vertices to a reference point (x_{min}, y_{min})) are easier targets to be optimized in compared to the absolute information of each vertices (i.e., the exact x and y values for each vertices). CTD inherited the strength of Faster-RCNN – the absence of complicated feature extraction design and multi-stage hueristic processes, and was proven to be both effective and efficient on the CTW1500 dataset. The proposed method is different with the said model in several ways: i) Polygon-FRCNN is modelled to detect text in word-level (inline with the ICDAR2013, ICDAR2015, COCO-Text, MLT datasets), ii) consequently, it has lesser parameters in the regression head to be optimized (i.e., six in compared to fourteen vertices)[10] iii) no need for the intermediate BLSTM module (again, due to the

[10]The best performing model, Poly-FRCNN-3.

smaller amount of vertices).

2.4 Summary

All the details of the aforementioned datasets in this section is summarized in Table 2.1, while the example images can be found in Figure 2.1. Further, there are several noticeable transitions in the launching of these datasets. Firstly, scene text datasets no longer feature only horizontal text, text orientation has been diversified with the inclusion of multi-oriented text. Then, as the performance of text detection algorithms gets better, new datasets like COCO-Text and ICDAR2015 include images with incidental text to increase the level of challenge. Also, the dataset scale has been increasing as well, as more labeled data is required to cater the emergence of data-driven deep learning text detection models. Lastly, the launching of *Total-Text*, CTW1500, and MLT further challenge the robust level of scene text detector with curved-oriented and multi-script text in natural images. All of these transitions are illustrated in Figure 2.2 in chronological order. On a similar note, scene text detection algorithms have seen a considerable amount of evolution in the past few years too. It took the field more than a decade to transcend from low level features like color, edges, texture, etc., to higher level features like MSER and SWT. The quest of seeking for better feature design came to an end when CNN caught the attention of the computer vision community. In the post CNN era, researchers found different ways of incorporating deep learning methods into their scene text detection algorithms, which can be broadly categorised into three clusters: the segmentation-based approach, the proposal-based approach, and the single-network-based approach. Looking at the more related works like the multi-oriented detection algorithms and curved text detection algorithms, similar patterns were found in the higher level of their pipelines. Both problems were addressed in a similar fashion in the early days: researchers designed hueristic algorithms to stitch spotted characters candidates into word level output. In the later days, top-down

proposal approach became popular mainly due to the absence of complicated grouping mechanism and faster execution time, which have largely influenced the design of the proposed method.

CHAPTER 3: THE TOTAL-TEXT SCENE TEXT DATASET

3.1 Introduction

The attributes of datasets establish the nature of a real-world problem for the research community. A well-defined and properly labeled dataset usually sets the trend for the field as well. For instance, the fact that most if not all text instances in ICDAR2013 (and all its variants) are horizontal-oriented has encouraged most scene text detectors throughout the 2000s to early 2010s to incorporate such assumption while solving the scene text detection problem. Then, the introduction of MSRA-TD500 has turned scene text researchers' heads to the multi-oriented text detection problem. The scene text community embraced another transition when ICDAR2015 was introduced. Reseachers benchmarked their scene text detectors on ICDAR2015 to evaluate its effectiveness against text instances that are small, out-of-focused, and randomly placed. All of these have indicated the impact of datasets have on the research field. The neglection on the scene text detection problem was inevitable due to the lack of curved text instances in the existing datasets. As a matter of fact, multiple research works have reported their successful and failure cases on the limited curved text intances in the existing dataset, but failed to evaluate it thorougly due to the absence of sufficient data. Such phenomenom has shaped the first motivation of this thesis work: the collection of a new scene text dataset which prioritise curved text instances, namely *Total-Text*, with the aim of facilitating a new research direction for the scene text community.

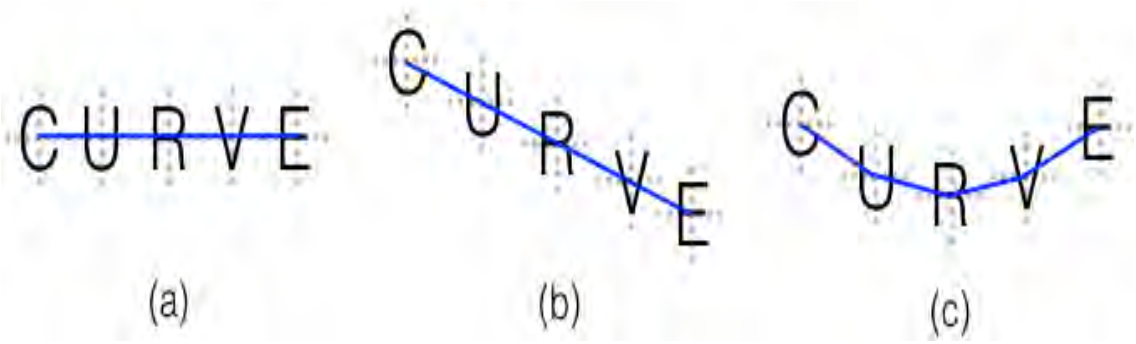


Figure 3.1: Text arranged in different orientations: (a): horizontal text that can be connected by a horizontal line, (b): multi-oriented text that can be connected by a straight line with an unified angle offset with respect to a horizontal line, (c): curved text with no unified angle offset from one character to another.

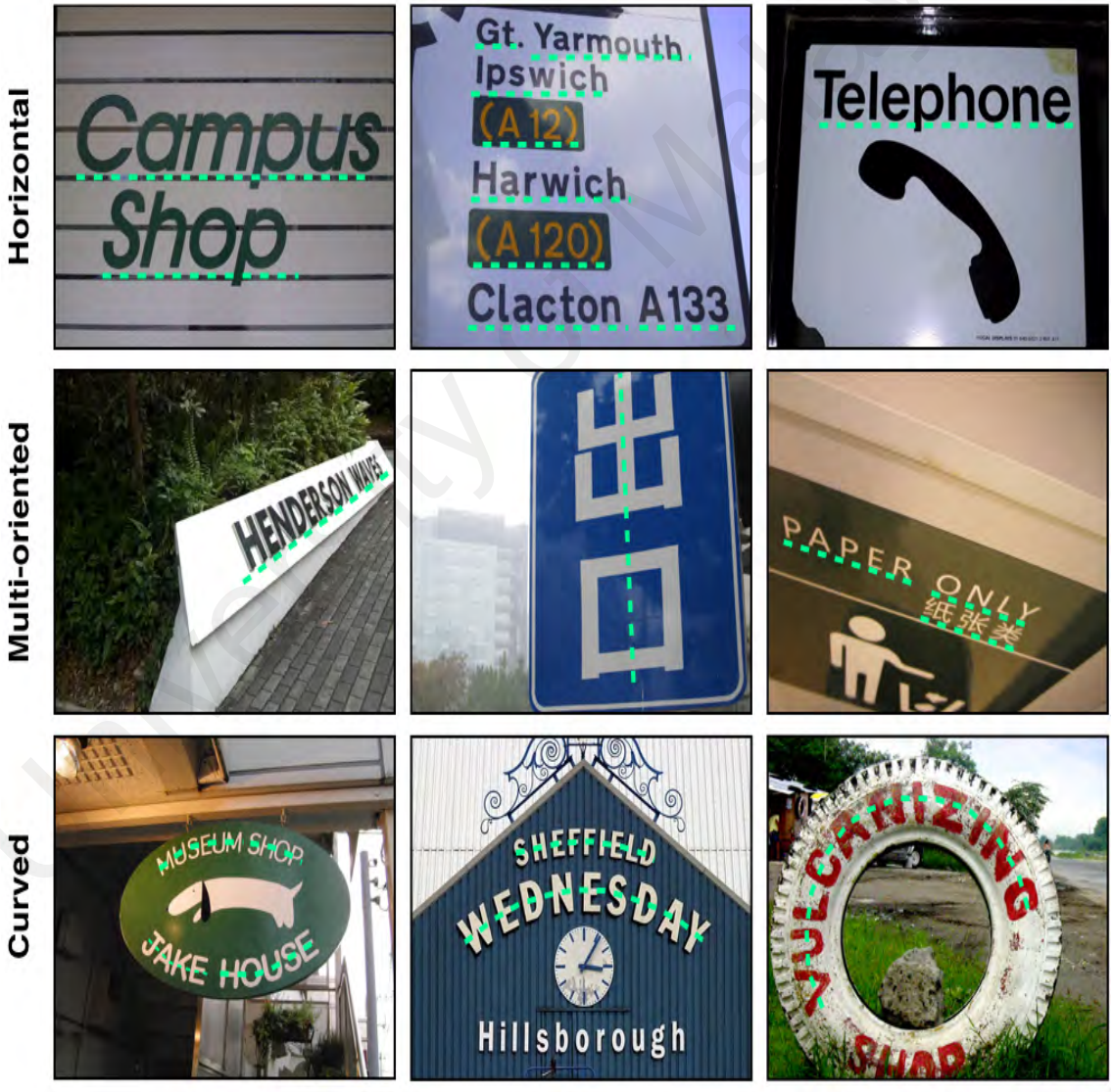


Figure 3.2: Example images of text with different orientations. Top row: horizontal text instances from ICDAR2013, Middle row: multi-oriented text instances from MSRA-TD500 and ICDAR2015, Bottom row: curved text instances from *Total-Text*.

3.2 Orientation of scene text

Text in natural scene comes in a variety of orientations. While horizontal text might be the most common one, it does not represent the entirety of text in wild. Geometrically[1] speaking, a straight line has *no angle variation* along the line, and thus can be described with the linear function, $y = mx + c$. A curved line on the other hand is free of angle variation restriction throughout the line. Shifting to the scene text perspective, it is observed that **horizontal oriented text** or word is a series of characters that can be connected by a straight line (their bottom alignment in particular for most cases). Similarly, **multi-oriented text** in the scene text convention[2] can also be connected by a straight line, given an offset with respect to a horizontal line. Meanwhile, characters in a curved text instance do not have an unified angle offset, in which deemed to fit a polynomial line. Figure 3.1 is the psuedo example to visualize the aforementioned description, from horizontal to curved text, while Figure 3.2 shows the real image examples with different text orientations. Due to the wider definition of curved text, there are variations within it as well. For instance, the example in Figure 3.2 (3rd row) depicts three curved text instances with different curvature: slightly to extremely (from left to right) curved. Further exploration on the difference of curved text will be discussed in Section 4.6.

3.3 Overview of *Total-Text*

3.3.1 Statistics of *Total-Text*

This section will discuss the numbers of Total-Text. The 1555 images of Total-Text is split into two groups, 1255 and 300 for training and testing set respectively. Figure 3.3 shows a series of statistical information of Total-Text. It has a total of 11,459 anno-

[1] In the mathematical context.

[2] Based on the multi-oriented scene text images in existing datasets like MSRA-TD500 and ICDAR2015

tated text instances and 7.37 instances per image on average. These numbers are superior to ICDAR2013 and MSRA-TD500, yet at the same time on par with ICDAR2015 and CTW1500, demonstrating its standard in terms of numbers. Figure 3.4 is a histogram that visualizes the distribution of *Total-Text*'s text instances. It shows that most of the images have roughly 3 to 6 text instances per image. The number goes as high as 117 in one single image, which also can be seen in Figure 3.4. Furthermore, more than half of the images in *Total-Text* have two different text orientations or more. Figure 3.3c illustrates the percentages of different text orientations in Total-Text. Approximately slightly under half of the text instances are curved (42.8% or 4907 in number), and the other half is split almost equally between horizontal (31.3% or 3585 in number) and multi-oriented (25.9% or 2967 in number). It is worthy of note that although all the images were collected with curved text in mind, other orientations still occupy half of the total instances. A closer look into *Total-Text* revealed that curved text usually appears alongside either horizontal or multi-oriented text. With the mixture of text orientations in an image, *Total-Text* challenges text detection algorithms to achieve robustness and generalisation in terms of text orientations.

3.3.2 Attributes of *Total-Text*

Apart from these solid numbers, the dataset was also collected with quality in mind. Scenery complexity such as text-like and low contrast background, different font types and sizes, etc., can be seen in Figure 3.5. In addition, the sceneries in *Total-Text*'s images are well diversified too. Figure 3.6 shows some examples of curved text occurrences, they can be found on business signboards (restaurants, hotel, merchants, etc.), sport clubs logo or jersey, tourist spots, food packaging, fire alarms, bus stops, etc. Another interesting observation is that curved text instances usually exist under the circumstances where it matches the shape containing it (round and arch for instance), as illustrated in the first

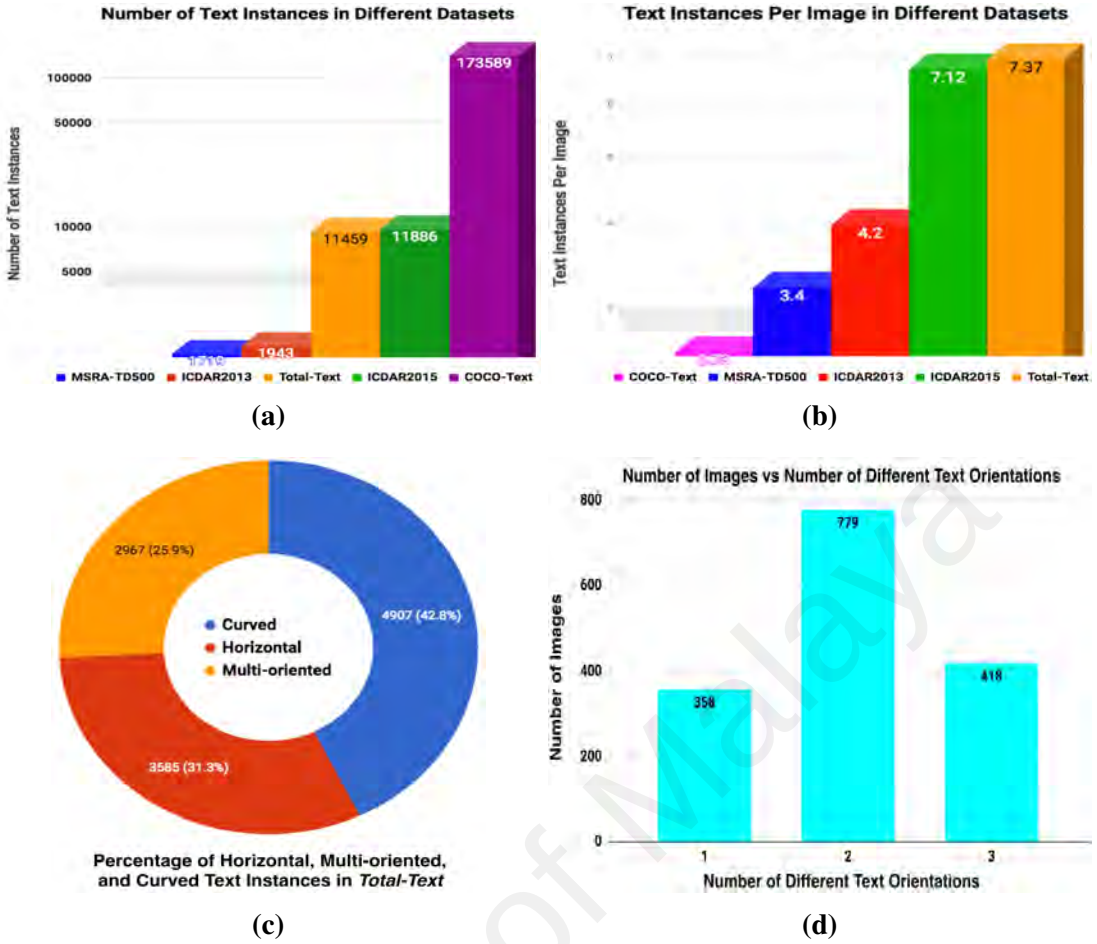


Figure 3.3: Statistics of the *Total-Text* dataset

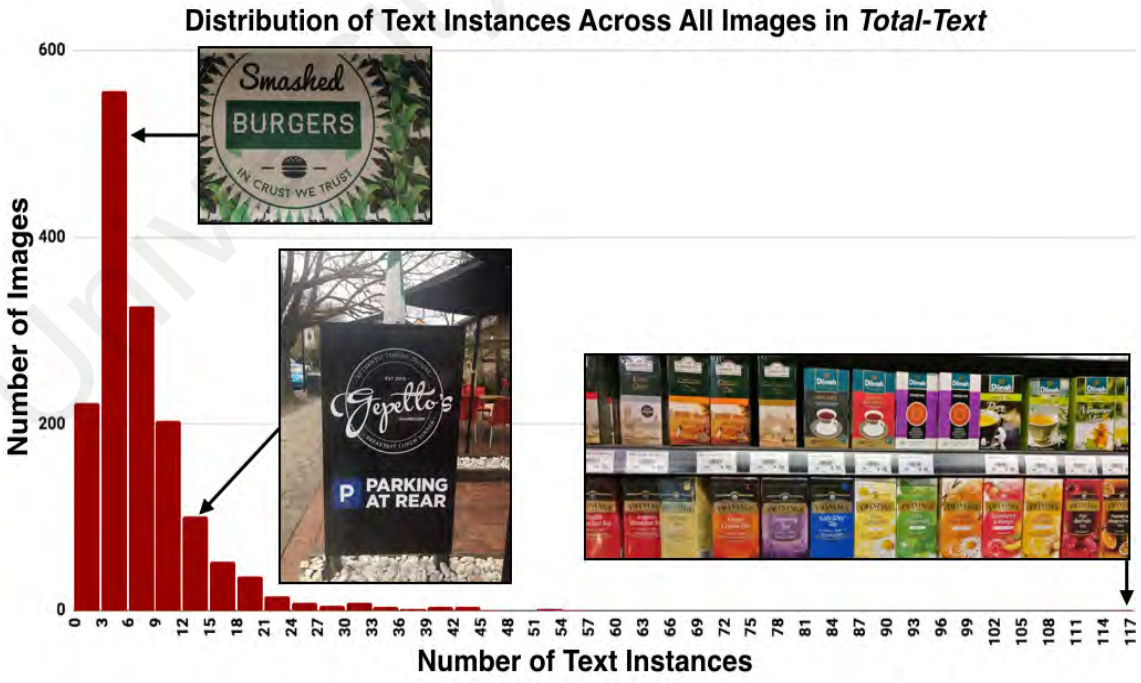


Figure 3.4: Images with different number of text instances.

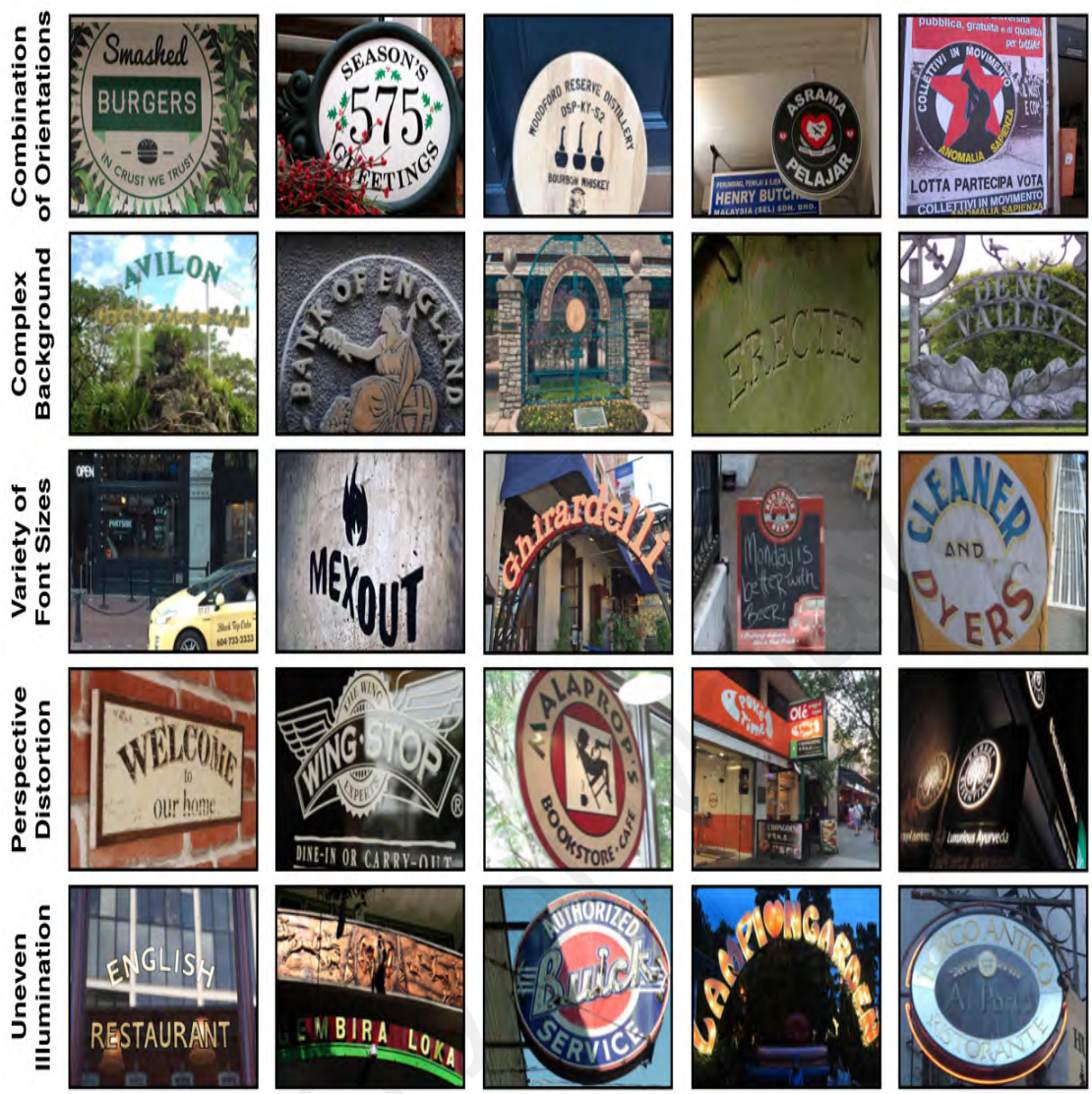


Figure 3.5: Challenges in *Total-Text*. Apart from the commonly seen challenges in the scene text domain such as camouflage background, variations of font size, perspective distortion, and illumination, *Total-Text* challenges text understanding algorithm to handle a combination of multiple oriented text in a single image.

row of Figure 3.7. Such shapes are common in signs and entrances of our world today. On the other hand, curved text instances can also be found on non-round surfaces (as illustrated in the second row of Figure 3.7) albeit not as common. It appears to be more attention-catching than horizontal text in those examples.

3.4 Annotation details of *Total-Text*

Total-Text is annotated for three scene text understanding tasks: detection, recognition, and segmentation. Figure 3.8 shows an example of the ground truths for each of the tasks

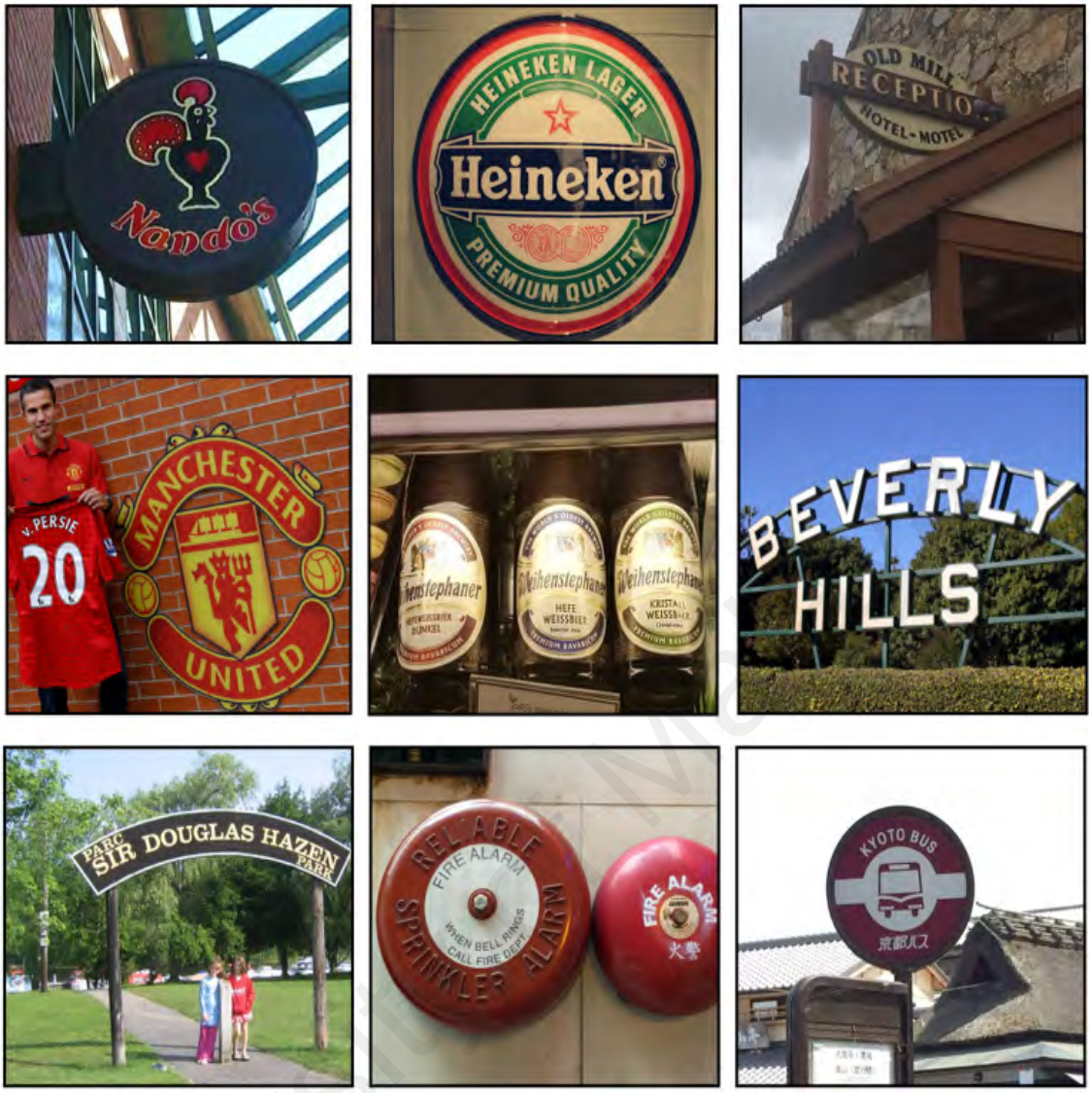


Figure 3.6: Diversified sceneries in the images of *Total-Text*.

(refer to Appendix A for more). The introduction of curved text has no direct impact on the ground truth of recognition and segmentation but detection task. Hence, a new detection ground truth format is introduced to cater the introduction of curved text. All the provided ground truths and their respective annotation process are discussed as follows.

3.4.1 Ground-truth Format for the Detection Task

Axis-aligned bounding box is the most common ground truth format in detection task regardless of its domain (Deng et al. (2009); Everingham et al. (2010); Lin et al. (2014)). It takes the least effort to annotate and deemed to be a good fit for general objects. As

Figure 3.7: Surfaces that contain curved text. Top row: round or arc surfaces which usually define the shape of curved text. Bottom row: normal surfaces which is not as common, the arrangement of curved text appears to be more attention-catching than the other orientations.

shown in Figure 3.9, an axis-aligned bounding box only requires two dots (as represented by '*'). Among the well known scene text datasets that was annotated for detection task, ICDAR2013 and COCO-Text adopted such format. While it is an appropriate format for ICDAR2013 mainly because most of its text instances are horizontal, it is not the best format for COCO-Text owing to the fact that it features text of all orientations. As shown in the first column of Figure 3.9, axis-aligned bounding box could not fit text other than horizontal orientation tightly. On the other hand, MSRA-TD500 uses rotated bounding box as the choice of ground truth format. Rotated bounding box is encoded in five values, (x, y, w, h, θ), which is considered as a good fit for multi-oriented text. However, such ground truth format requires the characters within a text instance to have a unified offset angle, which could not fit text instances with persective distortion or different font sizes properly as illustrated in Figure 3.9 (last two images of second column). ICDAR2015

	label	x-vertices	label	y-vertices	transcription	orientation
1	'x:'	[160,222,281,300,228,190]	'y:'	[212,162,132,172,205,243]	'BORDER'	'c'
2	'x:'	[299,347,402,400,357,304]	'y:'	[124,115,110,153,153,170]	'ROADS'	'c'
3	'x:'	[410,517,604,582,505,406]	'y:'	[113,137,197,233,174,152]	'ORGANISATION'	'c'
4	'x:'	[228,433,432,225]	'y:'	[264,262,306,311]	'SIMPLICITY'	'm'
5	'x:'	[441,478,480,444]	'y:'	[263,263,307,309]	'IS'	'm'
6	'x:'	[485,553,551,496]	'y:'	[260,258,302,303]	'THE'	'm'
7	'x:'	[193,301,304,194]	'y:'	[335,331,378,377]	'PEAK'	'm'
8	'x:'	[311,360,357,313]	'y:'	[334,330,371,379]	'OF'	'm'
9	'x:'	[367,587,589,368]	'y:'	[329,321,363,373]	'CIVILIZATION'	'm'
10	'x:'	[404,489,486,403]	'y:'	[394,391,415,425]	'JESSIE'	'm'
11	'x:'	[489,586,586,491]	'y:'	[386,382,409,416]	'SAMPTER'	'm'
12	'x:'	[143,272,284,143]	'y:'	[458,449,479,494]	'491RMPL'	'm'
13	'x:'	[354,396,398,354]	'y:'	[442,441,475,475]	'54'	'm'
14	'x:'	[398,444,450,401]	'y:'	[440,437,468,473]	'RCC'	'm'
15	'x:'	[493,546,543,499]	'y:'	[425,419,475,477]	'16'	'm'
16	'x:'	[558,613,609,564]	'y:'	[421,421,466,471]	'TF'	'm'

Figure 3.8: Annotation provided in *Total-Text*. Detection: polygon bounding region (*x* and *y* vertices), recognition: transcription, segmentation: pixel level binary map. The orientation information of every text instances are annotated as well.

was the first scene text dataset to employ quadrilateral in annotating text instances. It consists of eight parameters (four vertices, $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$), which has more degree of freedom than the rotated bounding box, hence it is capable of binding any multi-oriented text instances (as depicted in the third row of Figure 3.9). However, none of these bounding region formats fit the shape of curved text tightly, as shown in the last row of Figure 3.9. The usage of such formats would result in i) overlapping of text instances and ii) the inclusion of background regions. Inspired by CUTE80 (Risnumawan et al. (2014)), *Total-Text* employs polygon as the detection ground truth format. As illustrated in the last

column of Figure 3.9, polygon with no vertices restriction is capable of binding text of all orientations tightly.

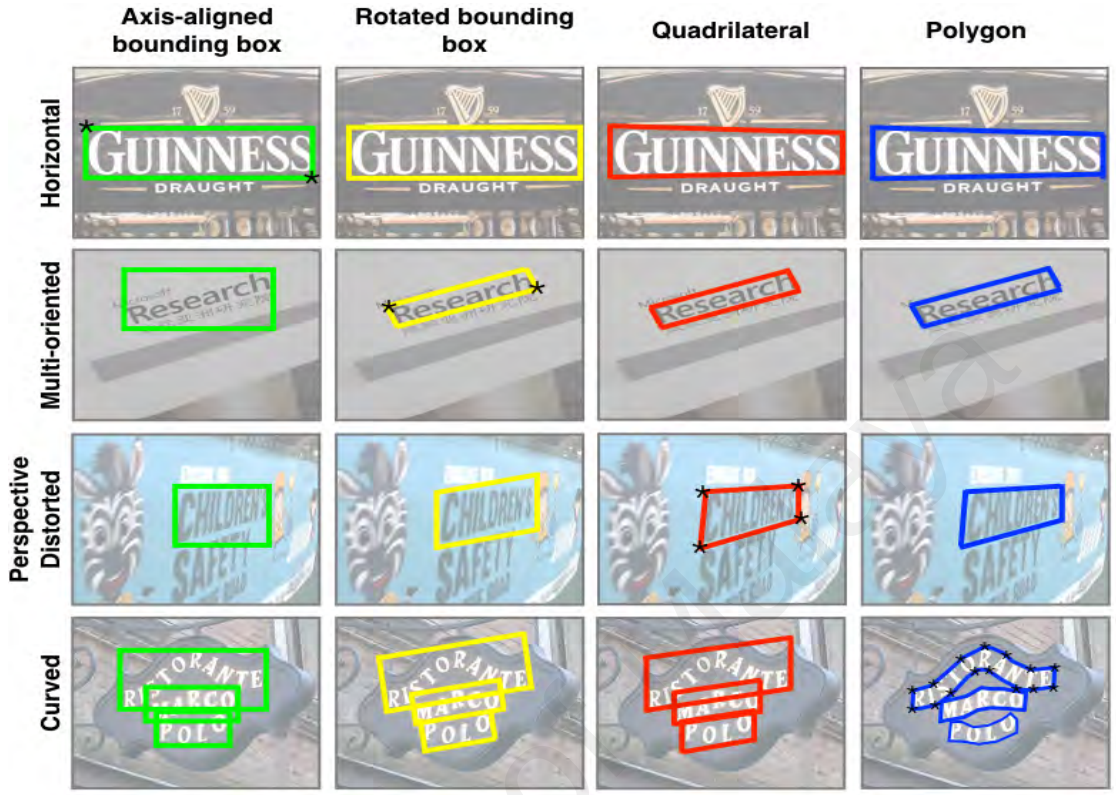


Figure 3.9: Images with text of different orientations, binded by different ground truth format. '*' represents the vertices of the ground truth format. Polygon with no restriction on the number of vertices is able to bind text of all orientations tightly.

3.4.2 Annotation Workflow and Details

This section discusses the annotation workflow employed in annotating the ground truth of *Total-Text*. Specifically, the annotation process of detection, recognition and segmentation will be explained in subsection 3.4.2 (a), 3.4.2 (d), and 3.4.2 (e), with further details in subsection 3.4.2 (b) and 3.4.2 (c).

3.4.2 (a) Detection and Recognition Annotation Process

The process of annotating image with detection ground truth is basically identifying every text instance in an image and provide its location information. As discussed, *Total-Text* employs polygon as the format for the location ground truth. Hence, the ultimate goal

is to locate and bind every text instance tightly with polygon vertices. Meanwhile, the ground truth of the recognition task – the transcription of the text instance, can be carried out at the same time too. The entire workflow is as follows:

i) Display an image in the dataset, and prompt annotator to count the number of text instances available in it. (Figure 3.11a)

ii) Request annotator to bind any of the unannotated text instances with axis-aligned bounding box. (Figure 3.11b to 3.11c)

iii) Crop the image based on the bounding box inputted by annotator, then enlarge the area and display it. (Figure 3.11d)

iv) Prompt annotator to bind the text region with **polygon vertices** (i.e. clicking on the edge of the text instance, as represented by the 'blue dot' in Figure 3.11e to 3.11f).

v) Prompt annotator to enter the **transcription** of the text instance. (Figure 3.11g)

vi) Prompt annotator to enter the **orientation** of the text instance. (Figure 3.11h)

vii) Repeat step (ii) to (vi) until all the text instances are annotated.

Additionally, orientation of every instance were annotated for modularity convenience. For example, if one prefers to evaluate the effectiveness of an algorithm to detect curved text only, one could leverage this annotation to filter out instances with other orientations. The annotation script was written in Matlab programming language (version R2015b), the result of the annotation was saved as the '.mat' file, example in Figure 3.8.

3.4.2 (b) Granularity of Text Instances

In line with ICDAR2013, ICDAR2015, COCO-Text, *Total-Text*'s text instances were annotated with word-level granularity[3]. Adopted from COCO-Text, word level text instances are an *uninterrupted sequence of characters separated by a space*. Figure 3.10

[3]Hence all the usage of 'text instances' in this work means word-level text unless stated otherwise.

illustrates the difference between word and line level text instances. Line-level granularity is often used when the dataset involves languages other than English, like Chinese, Indian, Korean, Japanese etc., since these languages do not have the 'space' element to seperate one word from another.

Figure 3.10: Differences between word and line-level ground truth. Line-level is usually employed in multi-lingual datasets which involve languages such as Chinese, Indian, Korean, etc. which do not have the 'space' element.

3.4.2 (c) Do Not Care Regions

Total-Text's annotation considers only English as text instances; other languages, digital watermarks and illegible text are labelled as *do not care* in the ground truth[4]. *Do not care* area picked up by detection algorithms should be filtered out before evaluating its performance. Equation 3.1 is recommended for such filtering process.

$$Area_o = \frac{Area_g \cap Area_d}{Area_d} \tag{3.1}$$

where detection candidates with $Area_o$ higher than 0.5 should be removed ($Area_g$ is the ground truth area and $Area_d$ is the detection area).

[4]With the '#' sign under the orientation and transcription category.

Figure 3.11: The baseline detection and recognition annotation process, the polygon was annotated with no restriction on the number of vertices. (a): Input image. (b)-(c): Bind the text region with a bounding box. (d): Zoomed-in version of the text region. (e)-(f): Annotate with polygon vertices. (g): Enter the transcription of the text. (h): Enter the orientation of the text.

3.4.2 (d) Regulated Polygon Annotation Process

The initial version of *Total-Text*'s polygon ground truth format as published in Chng & Chan (2017) (Section 3.4.2 (a)) was then found to be impractical for the recent regression-based methods due to its flexible length of vertices. Most of the regression-based methods (as discussed in Subsection 2.3.2 (b) and 2.3.2 (c)) today require a fixed length of regression parameters. Hence, a new version of polygon ground truth is introduced to address such problem. The new version of polygon ground truth (which will be termed as 'regulated polygon' from here on), has two attributes: i) the number of polygon vertices is fixed to ten (which was found emprically to be able to bind all the text instances in *Total-Text* tightly), and ii) the annotation of polygon vertices is regulated by a guiding mechanism to increase its consistency. A new annotation workflow is devised for the new regulated polygon annotation (inspired by the work of Yuliang et al. (2017)), which can be described as follows:

i) Prompt annotator to pick four different vertices that serve as the beginning and ending vertices covering of a text instance. As illustrated in Figure 3.12a, two vertices at the top corner of the word "MARLEY" (red and green dot), and two vertices at the bottom corner (blue and yellow dot).

ii) Calculate the distance between the top two vertices (red and green dot) and generate three equidistant lines (yellow in color) with Algorithm 1. (Figure 3.12b)

iii) Prompt annotator to select the select the interception point (represented as '*') along each yellow guiding line which best binds the top boundary of the word. (Figure 3.12b)

iv) Calculate the distance between the bottom two vertices (yellow and cyan dot) and generate three equidistant lines (green in color) with Algorithm 1. (Figure 3.12c)

v) Prompt annotator to select the select the interception point (represented as '*') along each green guiding line which best binds the bottom boundary of the word. (Figure 3.12c)

(a)

(b)

(c)

Figure 3.12: The workflow of the regulated polygon annotation scheme. (a): Prompt for four vertices. The system will then generate three equidistant guiding lines based on them. (b): Annotate the interception point (represented by the symbol '*') between yellow guiding lines and the top part of the text. (c): Lastly, annotate the interception point between green guiding lines and bottom part of the text.

These five steps are the replacement for step (iv) (as illstrated in Figure 3.11e and 3.11f) in the previous polygon annotation workflow (Section 3.4.2 (a)). The four initial points add to the six vertices from the top and bottom part of the text instances[5] form a polygon with ten vertices. The guiding mechanism, as formulated in Algorithm 1 has the following benefits: i) it relieves the cognitive strains to a certain extent during the annotation process as the annotator has less decision to make, ii) the distance between each polygon vertices are consistent[6], iii) removal of human bias (as the possible outcomes of the polygon with ten vertices is high, the pattern could vary from one human annotator to another). Figure 3.13 shows the comparison between the old and new ground truth.

[5]It could be left or right too, depending on the shape of the text. Regardless, the equidistant lines are generated by the first four initial points, which should be conciously annotated as a pair to cover one side of the text instance each.

[6]Except when the location of the guiding lines are not the best places for a vertex, it happens to some of the curved text intances which are extremely large. When that happens, annotator is required to annotate the vertices location as he/she deems appropriate.

(a) **(b)**

Figure 3.13: (a): Original ground truth with flexible length of polygon vertices. (b): New regulated ground truth with a fixed length of ten vertices.

Algorithm 1 Algorithm to generate the guiding lines in the regulated polygon annotation process.

Let the pair of input vertices be (x_1, y_1) and (x_2, y_2)
$x_{min} = \min(x_1, x_2)$
$x_{max} = \max(x_1, x_2)$
$y_{min} = \min(y_1, y_2)$
$y_{max} = \max(y_1, y_2)$
$width = x_{max} - x_{min}$
$height = y_{max} - y_{min}$
if width > height **then**
  equidistant interval = $(x_{max} - x_{min}) \ /4$
  $Line1_{x_1} = x_{min}$ + equidistant interval
  $Line2_{x_2} = Line1_{x_1}$ + equidistant interval
  $Line3_{x_3} = Line2_{x_2}$ + equidistant interval
else if height > width **then**
  equidistant interval = $(y_{max} - y_{min}) \ /4$
  $Line1_{y_1} = y_{min}$ + equidistant interval
  $Line2_{y_2} = Line1_{y_1}$ + equidistant interval
  $Line3_{y_3} = Line2_{y_2}$ + equidistant interval
end if

3.4.2 (e) Pixel Level Annotation Process

The ground truth for text segmentation task is a binary map labeled with text and non-text pixels. Its annotation process is the most time consuming process of all annotations. In order to ease the effort in annotating such ground truth, multiple preprocessing schemes are used in the workflow. The entire workflow can be described as follows:

i) From the original image, crop an image patch that contains text instance based on the annotated detection ground truth. (Figure 3.14a).

ii) Prompt annotator to adjust the threshold for all three color channels[7]. This is based on the observation that the color of the same text instances is usually in constrast to the background region. Adjusting the thresholds would be able to 'zero-out' the pixel (which is the background pixels in this context) below the adjusted threshold. (Figure 3.14b to 3.14c).

iii) Prompt annotator to bind the regions with the remaining background pixels in order to 'zero-out' them. (Figure 3.14d to 3.14f)

iv) Repeat step (i) to (iii) until all the text instances are annotated.

However, there are cases where the aforementioned threshold adjustment technique would not be able to help in preprocessing the text patches. Text regions with uneven illumination or color that is similar to the background region are two of such cases. Under such circumstances, annotator has the option to bind the character region from scratch, as visualized in Figure 3.15. At the end of the process, a binary map with text pixel labeled as '1' and background pixel labeled as '0' is produced and saved as a '.png' file.

[7]This is achieved by 'colorThreshold' function in Maltlab.

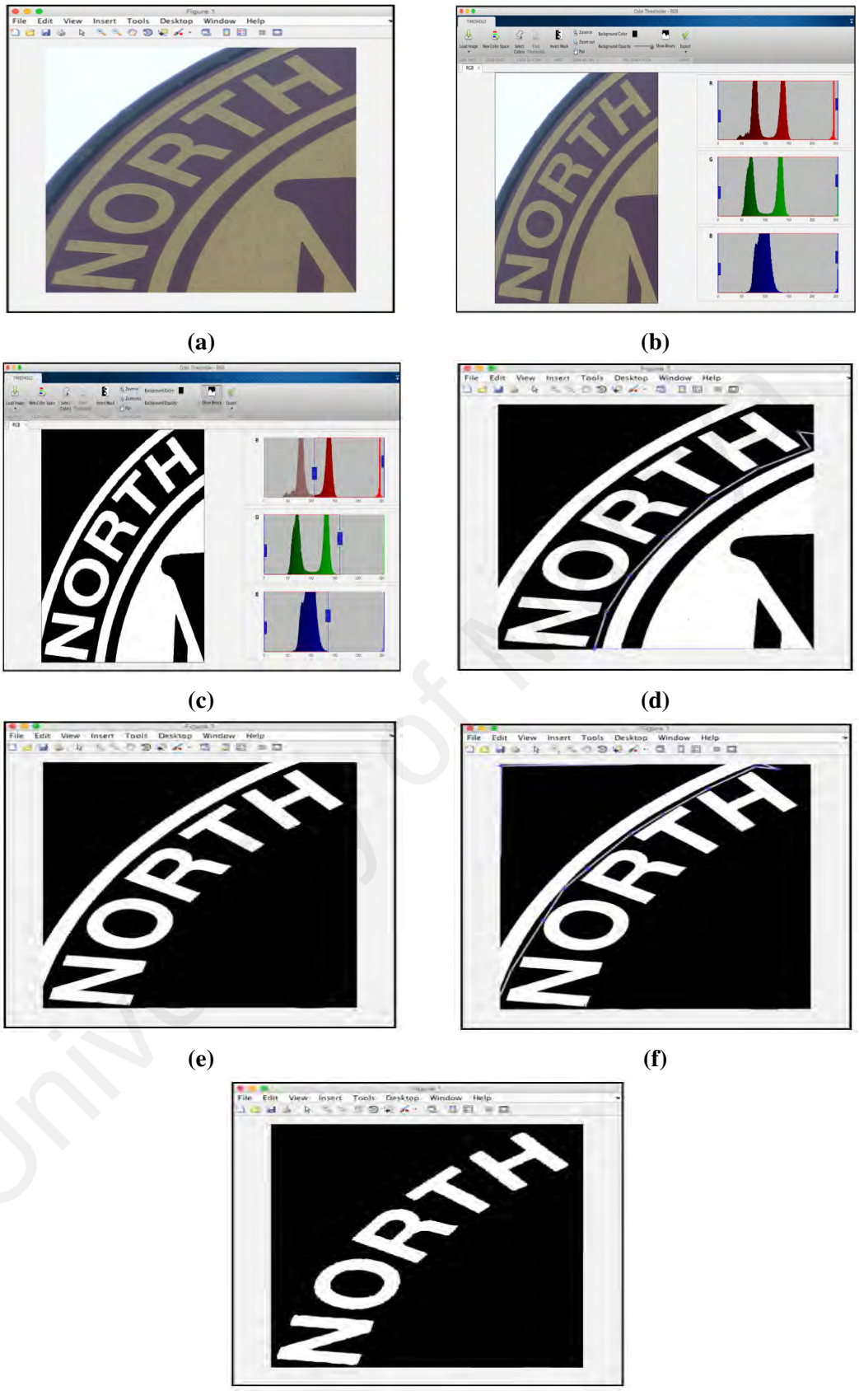


Figure 3.14: Pixel level annotation process. (a): Input image patch. (b)-(c): Adjust the color thresholds to seperate text from background region. (d)-(f): Remove 'non-text' region. (g): Final result.

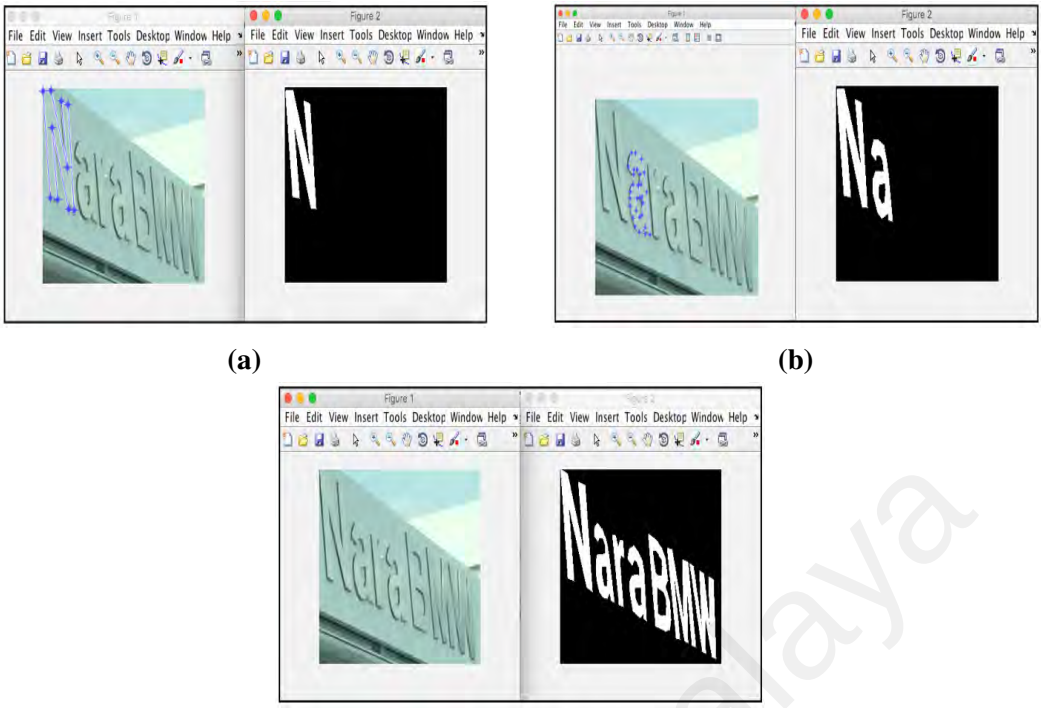


Figure 3.15: Pixel level annotation from scratch. (a) and (b) shows the process of binding the character region, while (c) depicts the final result.

3.5 In-depth Exploration of Curved Text

There are different kinds of curved text, which could be described in layman's terms like: *'the curvature of this text is more than that'*, *'the arc of this curved text is intruding, while that one is protruding'*, etc. However, such interpretation is subjective and of no help in large number analysis. Hence, this work believes that there is a need of being able to describe the variation of curved text objectively in concerete numbers. This section will reveal an encoding method to represent the differences of curve text. In addition, several insights regarding the curved text instances in *Total-Text* will be shown and discussed.

3.5.1 Curved Text Encoding

As explained in Section 3.2, a single straight line is not capable of connecting all the characters of a curved text. It requires a polynomial line with changing gradients. *Two straight lines with different slopes*, however, are able to do the trick coarsely. The encod-

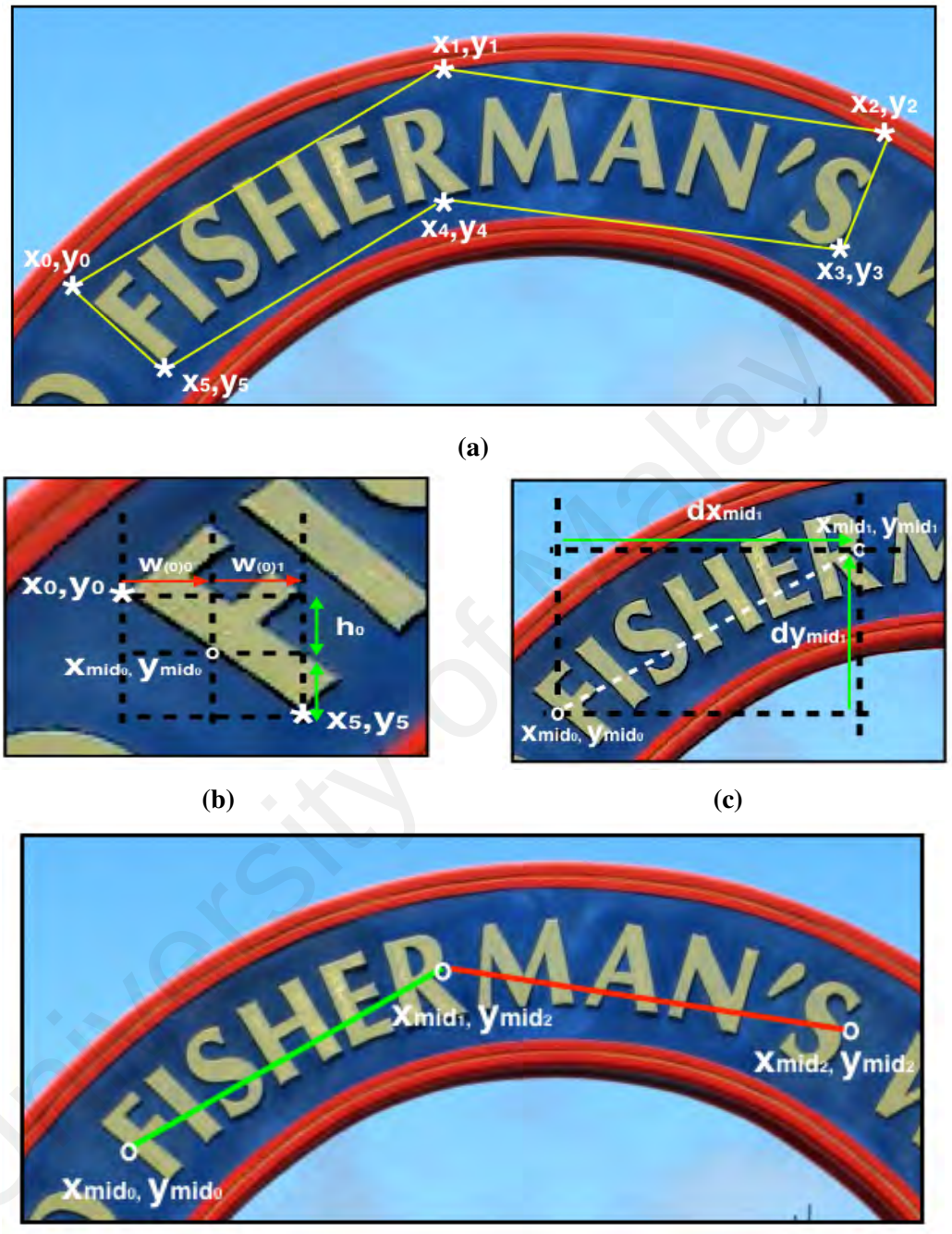


Figure 3.16: The proposed encoding method to represent curved text with two independent gradients. (a): Polygon ground truth format of Total-Text. (b): Encoding visualisation of $x_{mid(0)}$, $y_{mid(0)}$, $h_{(0)}$, $w_{(0)0}$, $w_{(0)1}$. (c): Encoding visualisation of $dx_{mid(i)}$ and $dy_{mid(i)}$. (d): Representation of curved text with two straight lines.

ing method can be described as follows. Firstly, a standardize ground truth format for all the curved text instances is required: 6-vertex polygon format is chosen as it is the minimum amount of vertex required to represent a curved text instance with two straight lines. Apart from the fixed number of vertices, the vertices were annotated in the following pairing manners: $\{x_0, y_0\}$ pairs with $\{x_5, y_5\}$, $\{x_1, y_1\}$ pairs with $\{x_4, y_4\}$, and $\{x_2, y_2\}$ pairs with $\{x_3, y_3\}$, while following a clockwise rotation, all of these descriptions can be observed in Figure 3.16a. Upon converting all the ground truths to polygons with six vertices, Equation 3.2 are used to determine three middle points from them, as visualized in Figure 3.16b.

$$
\begin{aligned}
x_{mid(i)} &= min(x_{(i)}, x_{((N-1)-i)}) + abs(x_{(i)} - x_{((N-1)-i)})/2 \\
y_{mid(i)} &= min(y_{(i)}, y_{((N-1)-i)}) + abs(y_{(i)} - y_{((N-1)-i)})/2 \\
&For\ i = [0, N/2)
\end{aligned}
\tag{3.2}
$$

The 'opposite pair' nature is embedded in the notion of i and $((N-1)-i)$, where N is the total number of vertices. Taking the $\{x_0, y_0\}$ and $\{x_5, y_5\}$ pair as example,

i) $x_{mid(0)}$ is the middle point between x_0 and x_5, and

ii) $y_{mid(0)}$ is the middle point between y_0 and y_5.

These middle points are then used to form two straight lines, which can be represented by its gradients ($dx_{mid(i)}$ / $dy_{mid(i)}$, for i $\in [1,2]$), as formulated and shown with Equation 3.3 and in Figure 3.16c and 3.16d. As mentioned, two straight lines with independent gradient are what is needed to represent curved text in the proposed encoding method. This encoding method can be extended to fully represent a polygon, which will be discussed in

Section 4.2.3 (a), as the regression target for the proposed curved text detection method.

$$
\begin{aligned}
dx_{mid(i)} &= x_{mid(i)} - x_{mid(i-1)} \\
dy_{mid(i)} &= y_{mid(i)} - y_{mid(i-1)} \\
For\ i &= [1, N/2)
\end{aligned}
\tag{3.3}
$$

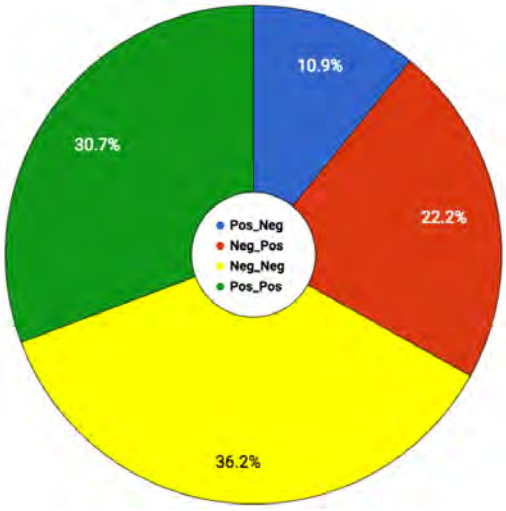
3.5.2 Signs of Gradient

The combination of the signs of the two gradients is the first helpful feature to describe different curved text. For instance, the curved text (i.e. FISHERMAN'S) in Figure 3.16 has a negative $dy_{mid(1)}/dx_{mid(1)}$ (i.e. the 'FISHER' part of the word), followed by a positive $dy_{mid(2)}/dx_{mid(2)}$ (i.e. the 'MAN'S' part of the word), mainly because $y_{mid(1)}$ is higher than $y_{mid(0)}$, and $y_{mid(2)}$ is lower than $y_{mid(1)}$[8]. Curved text in *Total-Text* can be categorized into four different groups based on this representation: *Neg_Pos*, *Pos_Neg*, *Pos_Pos*, and *Neg_Neg*, based on the sequence of their signs (*Neg_Pos* as in the curved text has a negative gradient followed by a positive gradient, similar notation for *Pos_Neg*, *Pos_Pos*, and *Neg_Neg*). Figure 3.17a shows the weightage of each group in *Total-Text*. Intuitively, curved text in the *Neg_Pos* and *Pos_Neg* group has a change of direction, forming what looks like an arc. *Pos_Pos* and *Neg_Neg* on the other hand have no change of direction but the magnitude difference between their gradient pairs makes them a curved instead of slanted text.

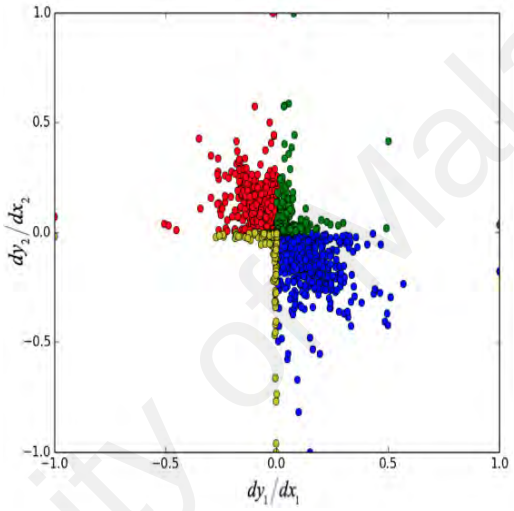
3.5.3 Magnitude of Gradient

Figure 3.17b is a 2D projection of every curved text instance with the magnitude of gradients representing the x and y axes. The first apparent observation is that the afore-

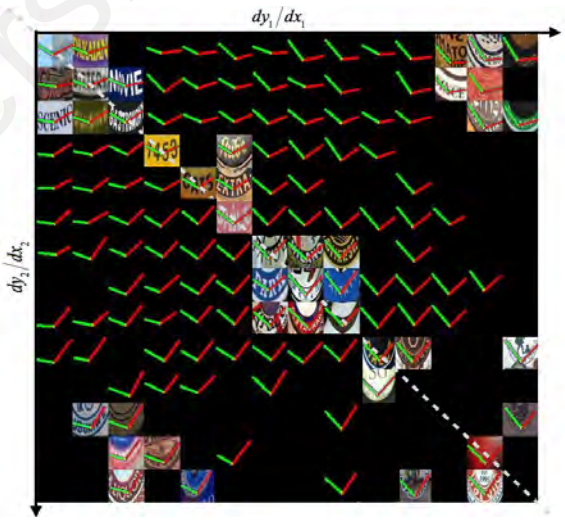
[8]Note that the origin of the y-axis starts from the top of the image



(a)



(b)



(c)

Figure 3.17: In-depth exploration of the curved text in Total-Text. (a): Percentage of different curved text in Total-Text. (b): Distribution of *Neg_Pos* (red), *Pos_Pos* (green), *Pos_Neg* (blue), and *Neg_Neg* (yellow), projected with their normalized gradients as *x* and *y* axis. (c): A closer look into the curved text images distribution at the *Pos_Neg* quadrant. The first and second line are color coded with green and red line for better observation. (Image patches were sampled)

mentioned curved variants (i.e. *Pos_Neg*, *Neg_Pos*, *Pos_Pos*, and *Neg_Neg*) split the plot into four quadrants with different sign combinations. Secondly, there are obvious gaps in the symmetrical zone of the *Pos_Pos* and *Neg_Neg* quadrants owing to the fact that proportional gradients would make them a straight line instead of curved. Meanwhile, Figure 3.17c is a close-up visualization of the *Pos_Neg* quadrant. Image patches with *Pos_Neg*'s curved text instances were cropped and mapped based on their coordinates in Figure 3.17c. The first interesting observation is that the curved text instances lie on the white diagonally dotted line have symmetrical arc. This is mainly due to the proportional gradients between the first and second line (i.e., $dy_{mid(1)}/dx_{mid(1)}$ is equal or close to $dy_{mid(2)}/dx_{mid(2)}$). Moving along the diagonal line, curved text intances in those images are seen to transit from slightly to extremely curved as the magnitude of both gradients increase. Moving away from the center line, the curved text images are observed to have higher gradient in one of the lines than the other depending on the axis it is leaning towards (i.e., steeper green lines close to the x-axis ($dy_{mid(1)}/dx_{mid(1)}$), steeper red lines close to the y-axis ($dy_{mid(2)}/dx_{mid(2)}$)). Figure 3.18 is the same kind of visualization but with all four quadrants, the 'zoomed-out' version. The image patches were ommited for better visualization. All the mentioned observations can be seen in all other quadrants. At the same time, each quadrants are observed to possess unique patterns: i) *Neg_Pos*'s curved text instances form a *'n'*-like arc, ii) while *Pos_Neg*'s curved text instances form a *'v'*-like arc, iii) *Pos_Pos*'s curved text instances are projecting downwards, and finally iv) *Neg_Neg*'s curved text instances are pointing upwards.

As a summary, the sign of gradients provides a high level overview on the curved text in *Total-Text*, while the magnitude of gradients allows us to navigate through different curved text with finer details. With this, the example in Figure 3.16 can be described as a curved text with a negative followed by a positive gradient. Its $dy_{mid(1)}/dx_{mid(1)}$ to $dy_{mid(2)}/dx_{mid(2)}$ ratio is 7.4, a curved text with steeper start than its end. Such represen-

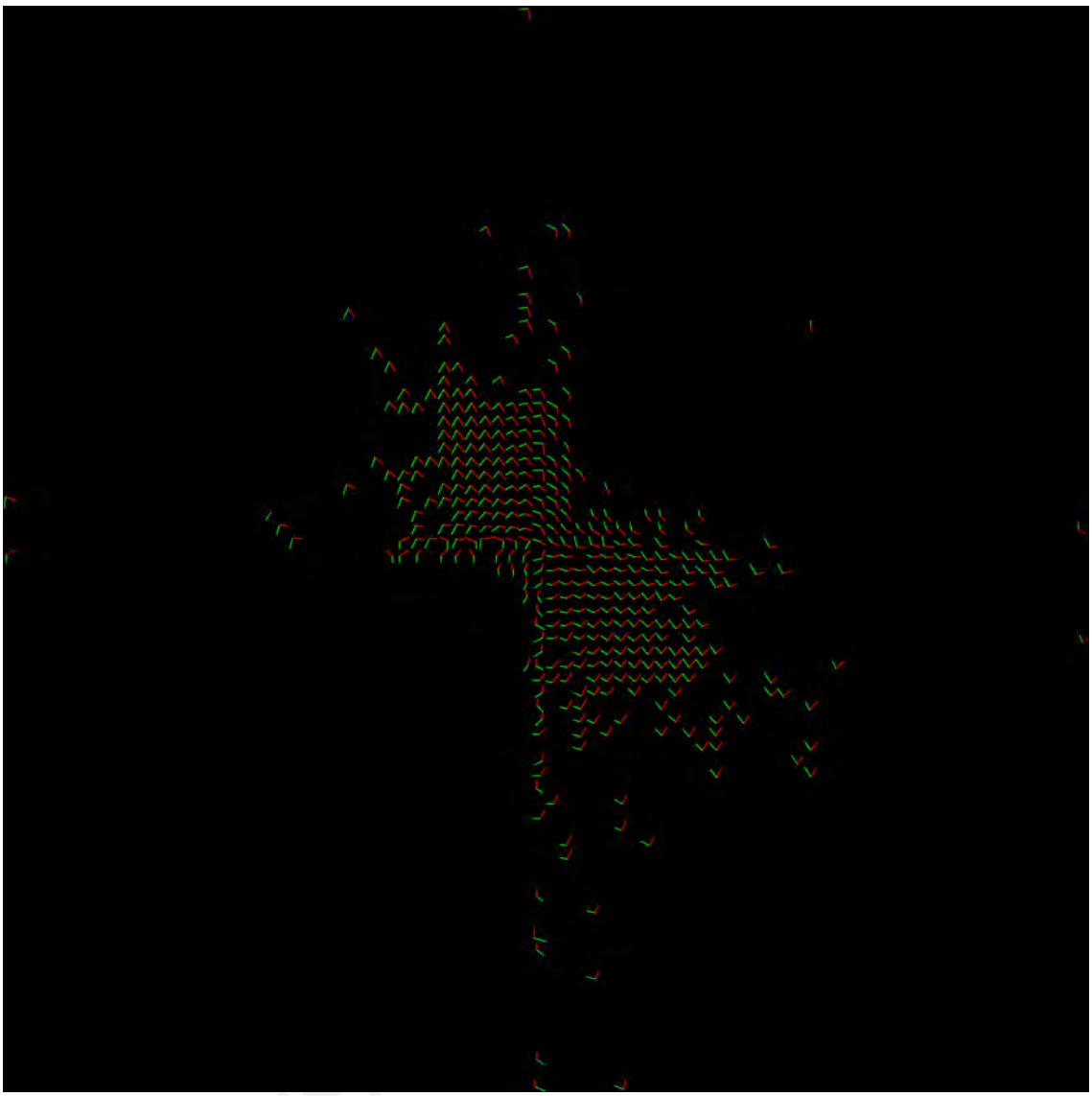


Figure 3.18: Visualization of all the curved text instances represented by two straight lines. Various patterns emerge for intepretation.

tation describes curved text instances in numerical terms, and hence made the in-depth explorations above possible.

3.6 Evaluation Protocol

There are two commonly used evaluation protocols in the community of scene text detection, namely Pascal VOC and DetEval. Both of them were modified and implemented for the evaluation of *Total-Text*. Pascal VOC's evaluation protocol was introduced in the Pascal VOC object detection competition, which employs IoU between detection and

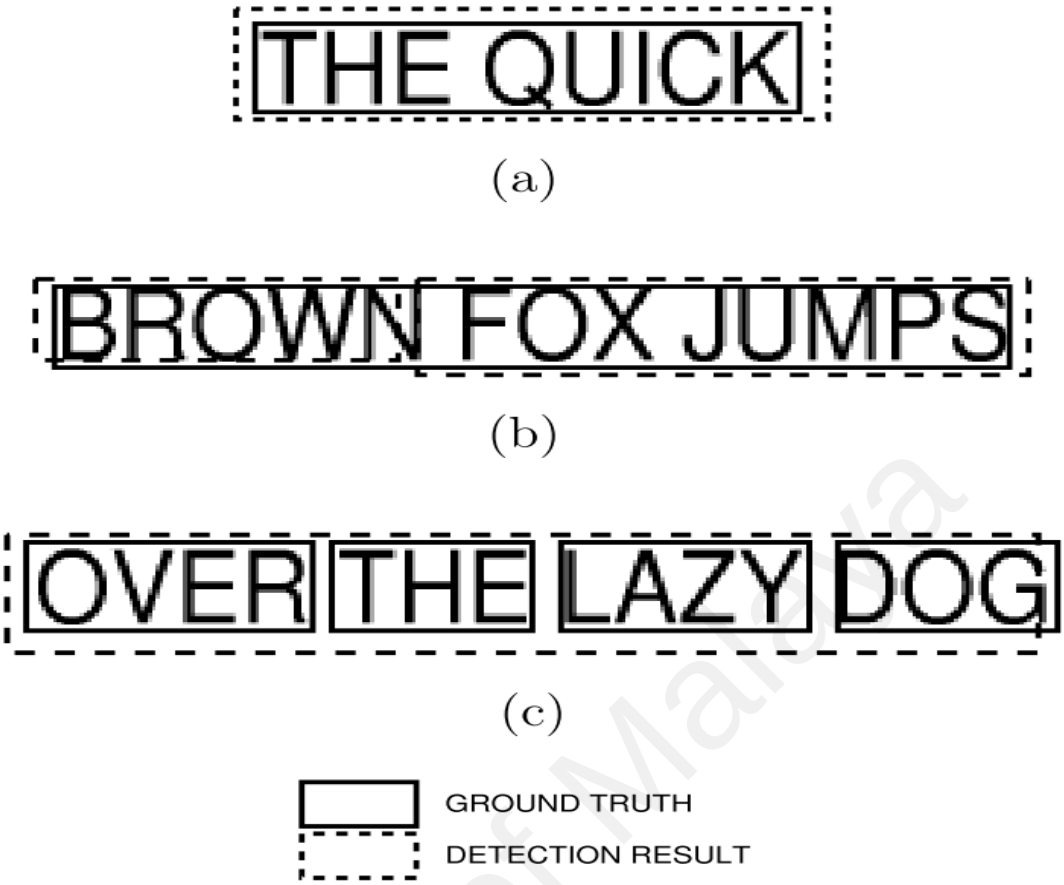


Figure 3.19: Multiple cases handled by DetEval. (a): One-to-One match, (b): One-to-Many match, (c): Many-to-One match. All of these are acceptable in the scene text understanding context.

ground truth regions as the metric of evaluation (Everingham et al. (2010)). Detection with more than 0.5 IoU will contribute to the accumulated summation of True Positive (TP). Any detection with lower than the IoU threshold will be regarded as False Positive (FP) and missed ground truth will be penalised with an increment in the False Negative (FN). DetEval on the other hand is specifically devised for the scene text detection problem (Wolf & Jolion (2006)). Unlike object detection, one ground truth being detected multiple times (One-to-Many), or one detection encapsulate multiple ground truth regions (Many-to-One) is acceptable in the application of scene text detection (Wolf & Jolion (2006)). Figure 3.19 illustrates three of the acceptable scenarios: One-to-One, One-to-Many, and Many-to-One; while Many-to-Many is a rare combination of One-to-Many and Many-to-One case. One of the main reasons behind this is the notion that

multiple words being detected as one, or the vice versa, is still useful in the context of text recognition, which usually is the next stage of text detection in a complete scene text understanding system. Unlike Pascal VOC, DetEval utilises two metrices to measure the quality of detection, which are τ and σ as fomulated in Equation 3.4 and 3.5 respectively. Moreover, both of these metrices are regulated by threshold tp and tr respectively. Only detection with higher τ and σ than the threshold values will be considered as a 'match'. After a thorough investigation, the convention tp (0.4) and tr (0.8) values, being used in the DetEval evaluation protocol were found to be inappropriate for *Total-Text*.

3.6.1 Precision of Detection, τ

τ is a metric in DetEval which measures *how precise is a detection*, as described in Equation 3.4 ($Area_g$ is the ground truth area and $Area_d$ is the detection area). Taking Figure 3.20 for example, it can be observed that the detection result of Poly-FRCNN is much tighter than Box-FRCNN's (both the detail of Poly-FRCNN and Box-FRCNN will be discussed in Section 4); such observation is also consistent with the actual precision calculation as tabulated in Table 3.1. As observed, the output of Poly-FRCNN consistently outperforms the loosely bounded output from Box-FRCNN. However, such performance difference is not reflected in the final calculation. Since the tp value is set as low as 0.4, all the detections of Poly-FRCNN and Box-FRCNN eventually contributed the same to the final score as all of them surpass the threshold despite the significant difference. In other words, the significant gained by the Poly-FRCNN is diminished by the low tp value. On the global scale, Figure 3.21a illustrates the performance of both methods (benchmarked on the test set of *Total-Text*) across all values of tp. It shows that Box-FRCNN's performance drops significantly as the threshold increases, while Poly-FRCNN's performance remained consistent. All of these suggest that an increment to the tp threshold is deemed necessary. As illustrated in Figure 3.21a, 0.6 is the point where Box-FRCNN's perfor-

Figure 3.20: Detection outputs of Box-FRCNN(red) and Poly-FRCNN(yellow) on one of the test images of *Total-Text*.

Table 3.1: τ and σ scores of both Box-FRCNN and Poly-FRCNN's output on Figure 3.20

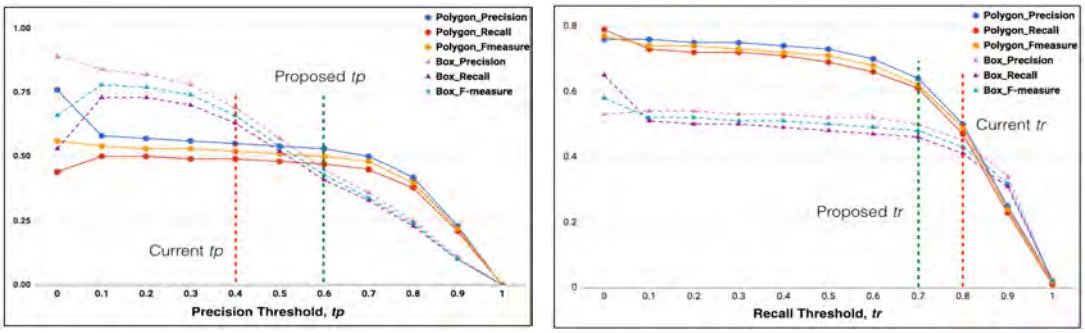
DetEval metric	τ		σ	
Detection | Methods	Box-FRCNN	Poly-FRCNN	Box-FRCNN	Poly-FRCNN
KAPILI	0.6	**0.98**	**0.99**	0.91
KISMET	0.51	**0.85**	**0.99**	0.92
HAN	0.4	**0.76**	**1.0**	0.78
10	0.8	**0.89**	**1.0**	0.95

mance drops beneath Poly-FRCNN, hence, this work recommends 0.6 as the default tp threshold for the evaluation on *Total-Text*.

$$\tau = \frac{Area_d \cap Area_g}{Area_d} \tag{3.4}$$

3.6.2 Coverage of Detection, σ

On the other hand, σ measures *how much of the ground truth area is covered*, as formulated in Equation 3.5. With the same example in Figure 3.20, Box-FRCNN's prediction boxes score almost or equal to full mark in all of the detection boxes while Poly-FRCNN's polygon detections are penalised as it try to match the ground truth precisely (see Table



(a) Sweeping of tp with tr fixed at 0.8. **(b) Sweeping of tr with tp fixed at 0.6.**

Figure 3.21: Performance of Box-FRCNN and Poly-FRCNN across different tr and tp with the DetEval evaluation protocol.

3.1). This is mainly due to the nature of Equation 3.5, which favours detection with large detection area. Furthermore, polygon ground truth has much more vertices (hence much more offset potential) than an axis-aligned bounding box, which is not an easy task to meet at a high level of agreement. Further experiment in Section 3.7 validates this by showing that even human annotation cannot perfectly match each other in multiple attempts. Hence, with the recommended sigma threshold, tr, as high as 0.8, the third detection Poly-FRCNN's prediction (corresponds to the 'HAN' ground truth) is forfeited in the final calculation. Figure 3.21b shows that by lowering tr by 0.1, to 0.7, the performance gap between the two became significant and much fairer.

$$\sigma = \frac{Area_d \cap Area_g}{Area_g} \tag{3.5}$$

3.7 Assisted Annotation Tool for Scene Text Detection

Ground-truth annotation is the biggest bottleneck in scaling up a dataset. Karatzas et al. (2017) introduced an on-line annotation platform that stresses on quality control and database management. However, the mundane and painstaking annotation task alone has much room for improvements. Quality annotation format such as the proposed regulated

polygon comes with a higher cost than the conventional axis-aligned bounding box and quadrilateral format. Inspired by Castrejón et al. (2017), this work proposes a new assisted annotation tool to ease the time consuming annotation process. The assisted annotation tool, namely Total-Text-Tool (*T3*), is capable of reducing annotation time by 25% with an aggreement rate of 84% with human annotators. The main differences between *T3* and Polygon-Recurrent Neural Network (RNN) (Castrejón et al. (2017)) are: i) the latter requires user to input image patches to the system, while *T3* takes the whole image, and ii) the suggestion (polygon vertices) in *T3* is designed to be aligned with the regulated polygon format as explained in Section 3.4.2 (d), which deemed to fit text instances better.

3.7.1 Total-Text-Tool (*T3*)

Training a modern scene text detector requires a lot of annotated data and time. In the scene text research domain, most of these invested resources reach the end-of-life when the performance tables are tabulated. The main idea of *T3* is to reuse the well trained and good performing scene text detectors to predict the potential text regions ahead of the annotation process. The predictions will serve as a suggestion for the human annotator.

The ideal flow of a good quality annotation tool should consist of i) cropping out text region with bounding box, ii) select a series of vertices for either quadrilateral or polygon that will bind the text instance tightly. This workflow is used in Karatzas et al. (2017) and Yuliang et al. (2017), which will be compared to as a baseline[9] in the following experiment. *T3* aids annotator with both of the processes in this workflow. The annotation process of *T3* is as follows:

i) Firstly, new scene image will be processed by a scene text detector to output both bounding box and polygon shaped predictions.

[9]The same workflow introduced in Section 3.4.2 (d)

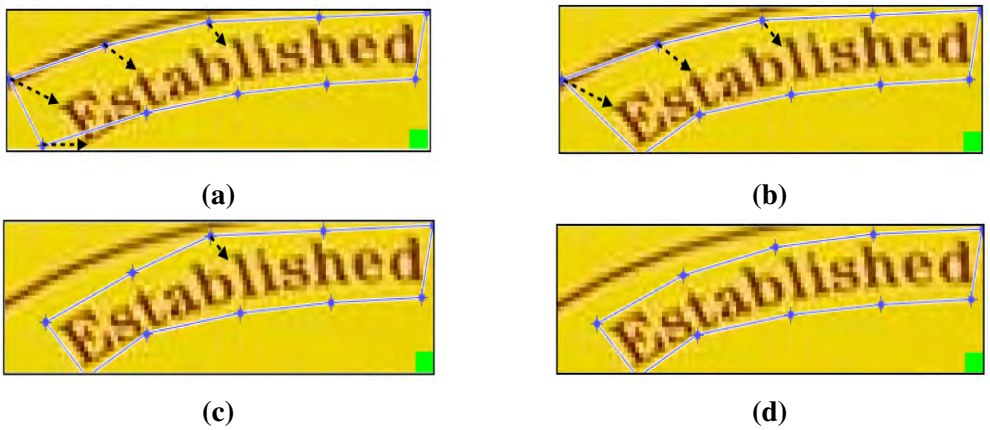


Figure 3.22: The annotator interaction part of *T3* – adjusting the suggested polygon vertices by *T3*. These figures are arranged sequentially, (a) - (d), black dotted lines represent the adjustment that took place.

ii) Secondly, *T3* crops image patches with the provided bounding boxes.

iii) Thirdly, display the cropped image patch with *interactive* polygon vertices for annotator to adjust and fit the text region tightly, as illustrated in Figure 3.22.

At the same time, annotator has the option to discard poorly suggested bounding box or polygon and manually re-annotate it from scratch using the baseline method. Figure 3.23 illustrates the annotation process of *T3*, the proposed vertices can be seen in the image patches with blue polygons, while the green ones are the examples post annotation.

3.7.2 Efficiency of *T3*

3.7.2 (a) Experiment Setup

An experiment was carried out in order to measure the efficiency using *T3*. The scene text detector used in the experiment is Poly-FRCNN-5 (see Section 4.3.1 for details). 100 images from the testing set was selected according to the actual F-measure distribution (depicted by the scale of x-axis of Figure 3.24) of Poly-FRCNN-5 to ensure the legitimacy of the experiment. The human annotator was informed and trained on both baseline and the *T3* annotation tool before the experiment. The human annotator was given the freedom to take a break whenever he feels like to, in order to ensure he did not suffer from fatigue

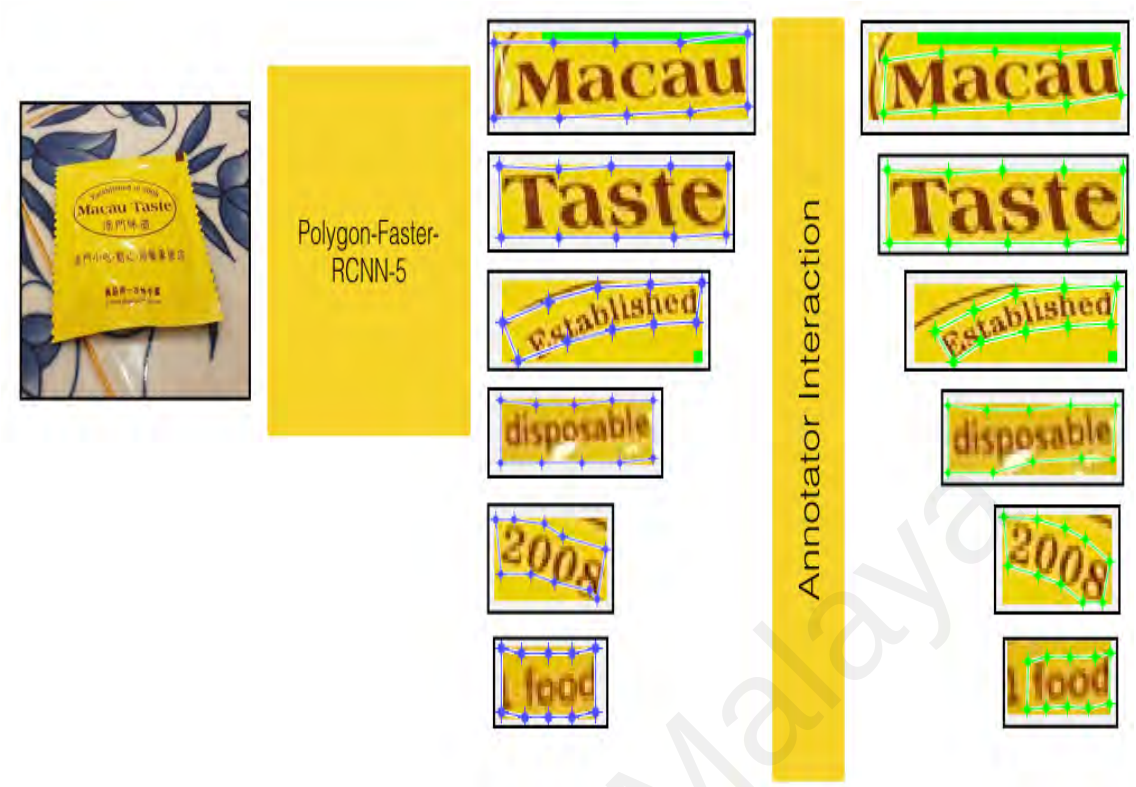


Figure 3.23: Overview of the detection ground truth annotation process with the aid of *T3*.

which would in turn introduce bias to the experiment. Both time and the annotation quality were measured internally (within the script) and individually to each image.

3.7.2 (b) Performance Analysis

In overall, it took the human annotator a total of 8989.58 seconds (2.5 hrs) to annotate all the 100 images with the baseline annotation tool. With *T3*, the total time taken is reduced by 25% to 6832.34 seconds (1.9 hrs). The quality of both annotation results are in an agreement of 84%, with the standard deviation of 4.6% in IoU. Figure 3.24 illustrates the comparison of time taken (normalized to one) between both methods across all 100 images. As observed in the mentioned figure, the better the performance of the detector on the image, the lesser time it takes for the human annotator to complete the annotation with the aid of *T3*. Note that on those images where the detector scores as low as 0 to 0.4, those suggestions were mostly bad, therefore, the human annotation was required to annotate

it from scratch. Hence the time taken for those images are almost identical or sometime worse with *T3* due to the extra decision required to discard the suggestions. Even in such cases, the agreement of annotation between the two methods could not match perfectly. This is mainly due to i) as pointed out in Castrejón et al. (2017), IoU is a strict metric for small object instances, which is the nature of scene text in general, and ii) polygon has much more vertices (ten in this case) hence comes with smaller margin for offset. The *T3* framework is generic, thus any scene text detector can be part of the workflow. The performance of Poly-FRCNN-5 (F-measure of 0.6 on *Total-Text*) has much more room for improvement, a better performing detector would futher reduce the annotation time with *T3*.

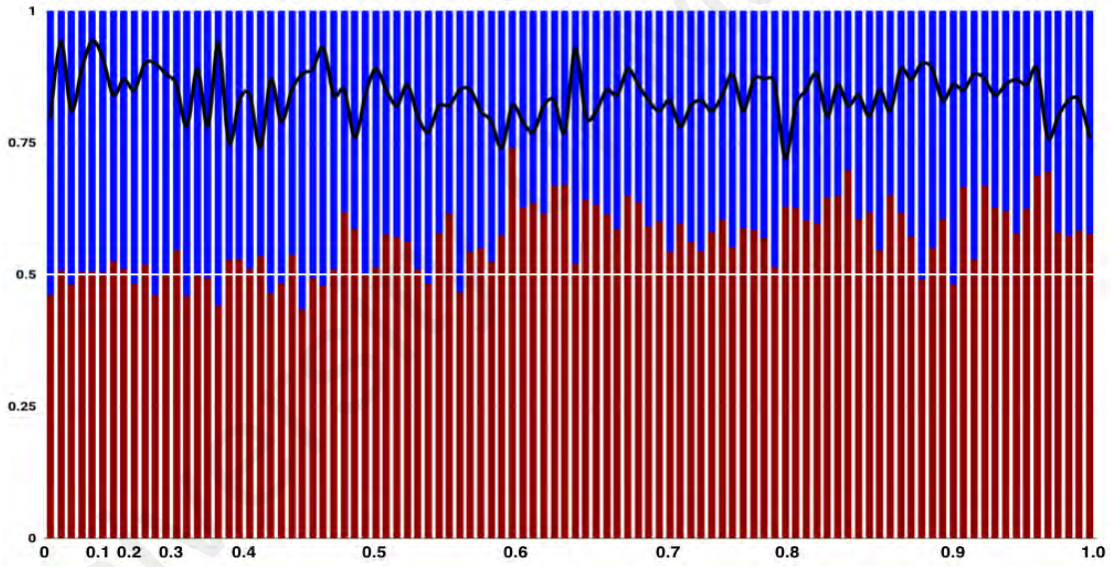


Figure 3.24: Comparison of time taken to annotate with *T3* (in blue) and without (in red). The better the performance of Poly-FRCNN on the image, the lesser time it takes for human annotator to complete the annotation with the help of *T3*. The agreement of annotation (black line) with the aid of *T3* is remained as high as the one without (left end of the chart), averaged at 84% IoU with a standard deviation of 4.6% across the 100 images used in this experiment.

3.8 Summary

In summary, the proposed dataset of this work, *Total-Text*, is introduced to fill the void of missing curved text. Curved text instances occupy more than 40% of its total 11459 text

instances. *Total-Text* has similar scale as existing scene text datasets like ICDAR2015 and CTW1500, which is considered as a properly scaled dataset. Apart from solid numbers, it consists of numerous challenging aspects like camouflage background, variation of font size, combination of multiple text orientations, etc. It was annotated for three different tasks: detection, recognition, and segmentation. A new detection ground truth format, regulated polygon, is introduced to cater the curved text region in *Total-Text*. Apart from being able to bind curved text region tightly, it has the following properties: i) number of vertices is fixed to ten, and ii) a guided algorithm is involved in generating equidistant polygon vertices. Such properties are desirable because it is practical for the existing regression model and the equidistant polygon has lesser human annotation bias. In addition, the curved text instances in *Total-Text* are analysed with the proposed polygon encoding method. All the curved text instances in the proposed dataset can be grouped into four main categories according to the encoding method. Next, both Pascal VOC and DetEval are the official evaluation protocols of *Total-Text*. The default threshold values were proven to be not optimised with the usage of polygon ground truth and detection format. A new set of thresholds values are proposed for more accurate evaluation result on the proposed dataset. Last but not least, a generic assisted annotation tool for the detection ground truth is described in the last part of this chapter. It demonstrates promising results with a reduction of 25% in the annotation time. With the availability of *T3*, it is hoped that the future annotation process can be sped up.

CHAPTER 4: CURVED TEXT DETECTOR – POLYGON-FASTER-RCNN

4.1 Introduction

Scene text detection with curved text in consideration is under-researched at the moment due to the lack of available data and ground truth (Chng & Chan (2017)). Upon collecting *Total-Text*, this work explored the possibility of extending the current state-of-the-art object detection framework – Faster-RCNN (Ren et al. (2015)), to a baseline model that is capable of detecting text of all orientations. This chapter discusses the design elements of the proposed method, Polygon-FRCNN, and its performance on various datasets including *Total-Text* to illustrates its effectiveness against text of all orientations.

4.2 Polygon-Faster-RCNN

Inspired by the success of (Jiang et al. (2017); Y. Liu & Jin (2017); Ma et al. (2017); Yuliang et al. (2017)) in building scene text detectors on top of the Faster-RCNN architecture, the proposed method of this work, is too built based on Faster-RCNN. Polygon-FRCNN, as teased in the name, is designed to detect text instances with the polygon format. The main motive of modeling Polygon-FRCNN to produce polygon output is in line with the reason of employing polygon as the ground truth format in *Total-Text* – to bind text of all orientations tightly. Figure 4.1 depicts the overall architecture of Polygon-FRCNN. Inherits from Faster-RCNN, Polygon-FRCNN has three major blocks: feature extraction, region proposal network, and polygon classifer network (originally box classifier network), all of which will be discussed in the following subsections.

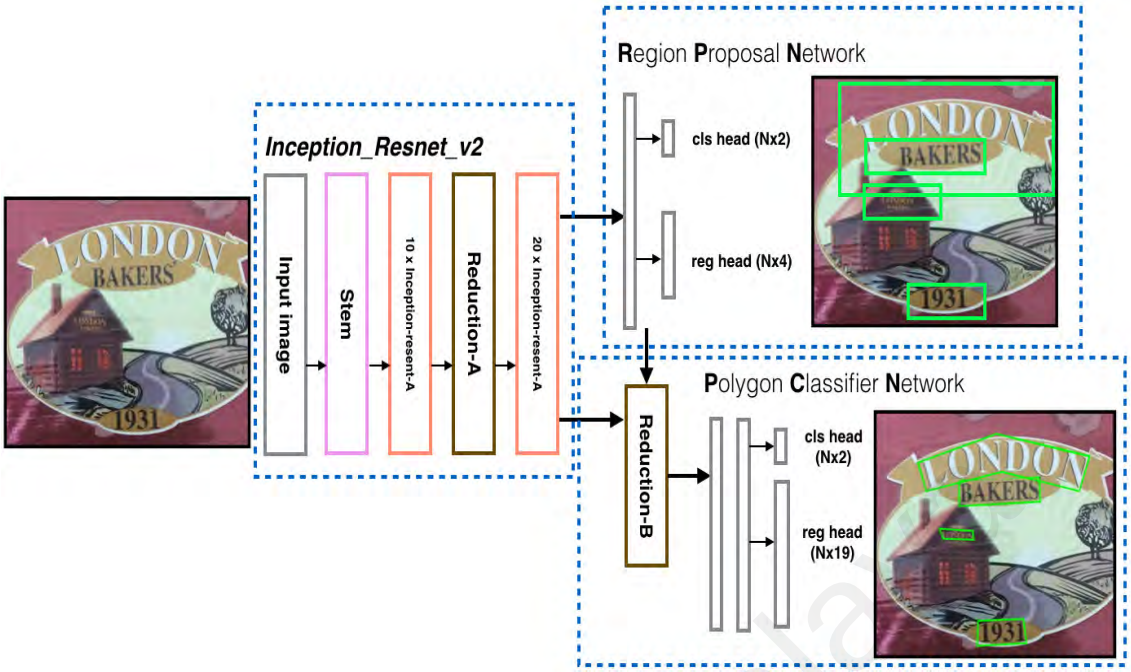


Figure 4.1: The overview of Polygon-FRCNN. It is based on Faster-RCNN with the modification in the second stage regression head. The regression head was extended from four to nineteen to include fifteen encoded polygon parameters.

4.2.1 Feature Extraction

The feature extraction part of a network is important as good features usually translate to good performance in computer vision problems. Polygon-FRCNN adopts the Inception-ResNet-v2 (Szegedy et al. (2017)) model as the feature extractor of the network. The rationale behind the option are i) Inception-ResNet-v2 has demonstrated its strength in the studies of Szegedy et al. (2017) and J. Huang et al. (2016) ii) pre-trained weights (on the MSCOCO (Lin et al. (2014)) dataset) were made available by J. Huang et al. (2016)[1], which eliminates the need of training the network from scratch that would take weeks and many computing resources. The network architecture of Inception-ResNet-v2 can be seen in Figure 4.2. It combines the strength from both ResNet (K. He et al. (2016)) and the Inception modules (Szegedy et al. (2015)) which allows i) large number of layers within the network, and ii) faster running speed with parameter factorization. Both of these

[1] Available in https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Table 4.1: Network details of the feature extractor (based on Inception-ResNet-v2). Note that $Mixed\_5b$, $Reduction-A$, and $Inception-resnet-B$ are made up of the combination of convolution and pooling operations (which can be referred to in Figure 4.2) with multiple sets of parameters (i.e. kernel size, padding, and stride). Hence, they are not tabulated for the sake of the consistency of the table.

Layer/block	input size	kernel size, k ($h \times w \times c \times num\_filters$)	padding, p	stride, s	output size
`Conv2d_1a`	$299 \times 299 \times 3$	$3 \times 3 \times 3 \times 32$	0	2	$149 \times 149 \times 32$
`Conv2d_2a`	$149 \times 149 \times 32$	$3 \times 3 \times 32 \times 32$	0	1	$147 \times 147 \times 32$
`Conv2d_2b`	$147 \times 147 \times 32$	$3 \times 3 \times 32 \times 64$	0	1	$147 \times 147 \times 64$
`MaxPool_3a`	$147 \times 147 \times 64$	$3 \times 3 \times 64$	0	2	$73 \times 73 \times 64$
`Conv2d_3b`	$73 \times 73 \times 64$	$1 \times 1 \times 64 \times 80$	0	1	$73 \times 73 \times 80$
`Conv2d_4a`	$73 \times 73 \times 80$	$3 \times 3 \times 80 \times 192$	0	1	$71 \times 71 \times 192$
`MaxPool_5a`	$71 \times 71 \times 192$	$3 \times 3 \times 192$	0	2	$35 \times 35 \times 192$
`Mixed_5b`	$35 \times 35 \times 192$	–	–	–	$35 \times 35 \times 320$
Inception-resnet-A	$35 \times 35 \times 320$	–	–	–	$35 \times 35 \times 320$
Reduction-A	$35 \times 35 \times 320$	–	–	–	$35 \times 35 \times 1088$
Inception-resnet-B	$35 \times 35 \times 1088$	–	–	–	$17 \times 17 \times 1088$

advantages made the construction and training of a very deep network possible, which is essential in extracting high quality features for the later regression and classification tasks.

The input image of the network is restricted to 600 - 1024 pixels, input image exceeding the range will be resized with the aspect ratio intact. At the end of the feature extraction module, a feature map of dimension ($h_{feature} \times w_{feature} \times 1088$) will be generated, where $h_{feature}$ and $w_{feature}$ are modelled with Equation 4.1, where dim_{in} and dim_{out} represent input and output dimension of feature maps respectively. Table 4.1 contains all the details of the Inception-ResNet-v2 based feature extractor. Meanwhile, it provides a demo calculation based on Equation 4.1 assuming the input image has a spatial dimension of (299×299). The extracted feature maps will then be shared between the region proposal network and polygon classifier network.

$$dim_{out} = [(dim_{in} + 2p - k)/s] + 1 \tag{4.1}$$

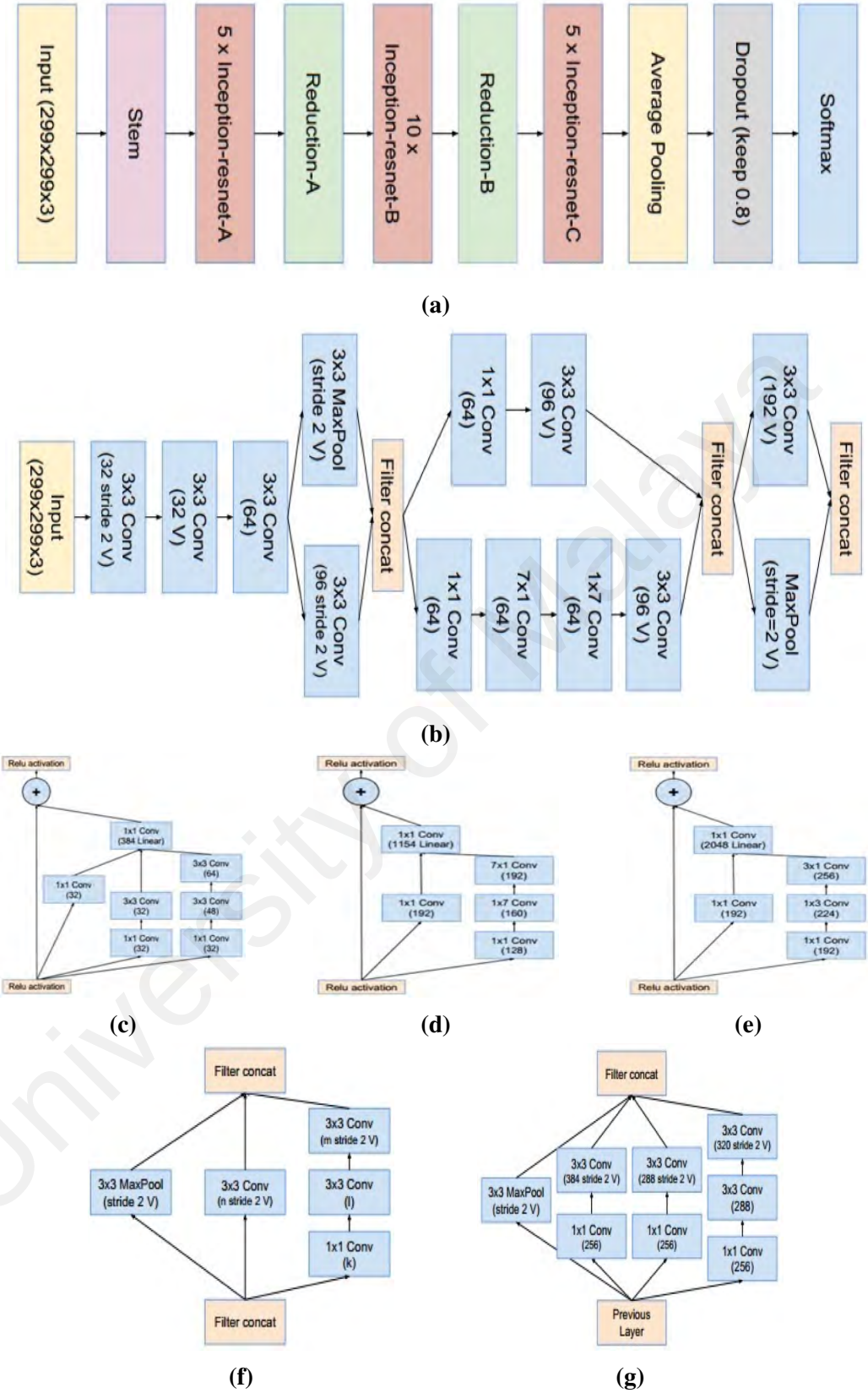


Figure 4.2: Overall architecture and its internal network details of Inception-ResNet-v2. (a): Overall architecture, (b): *'Stem'* block, (c): *'Inception-resnet-A'* block, (d): *'Inception-resnet-B'* block, (e): *'Inception-resnet-C'* block, (f): *'Reduction-A'* block, (g): *'Reduction-B'* block.

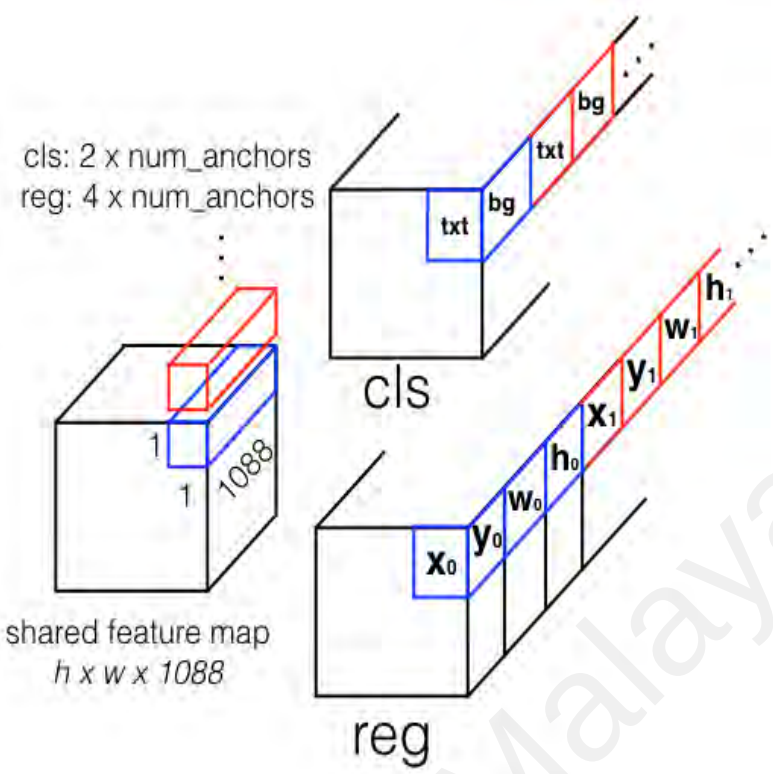


Figure 4.3: Classification (cls) and regression (reg) head of the Region Proposal Module.

4.2.2 Region Proposal Network

Upon transforming the input image to high dimensional feature maps (with the dimensions of ($h_{feature} \times w_{feature} \times 1088$)), a classification and regression head are attached to the feature map with the objective of i) classifying whether text exists, and ii) regress its location in axis-aligned bounding box format. Fundamentally, the classification 'head' is a convolutional operation with the filter size of $1 \times 1 \times 1088 \times (num\_of\_anchors \times 2)$ over every pixel in the extracted feature maps. In other words, the filter will learn to predict an input-dependent size of two-layer probability maps, indicating the likelihood of each pixel being 'text' or 'background' region. The parameter $num\_of\_anchors$ is associated with the design of anchors, which will be explained soon. Similarly, the regression 'head' is also a convolutional operation, but with the filter size of $1 \times 1 \times 1088 \times (num\_of\_anchors \times 4)$. The '4' here represents the encoded axis-aligned bounding box,

(x, y, w, h). Figure 4.3 illustrates both the classfication and regression heads with several details that will be referred to in the following part.

4.2.2 (a) Anchors

Layment description of anchors can be found in Section 2.3.2 (b), this section attempts to explain it in terms of technical implementation. Firstly, every spatial location in the feature map are associated with a number of predefined anchors, which is one of the hyperparameters in the Faster-RCNN framework. These anchors come in different scales and aspect ratios. The work of Ren et al. (2015) claimed that anchors with the scales of 128^2, 256^2, and 512^2, and the aspect ratios of 1:1, 1:2, and 2:1 helped to achieve good performance in general object detection task. The mentioned work ended with nine anchors per location (three scales with three aspect ratios each) on the final layer of the feature map.

Multiple studies (Liao et al. (2018, 2017); Ma et al. (2017)) have shown that anchors with the default scales and aspect ratio did not work well with text in general. Inspired by their findings, the anchors in Polygon-FRCNN is designed with the scales and aspect ratio of 16^2, 32^2, 64^2, 128^2, and 256^2 and 2:1, 1:1, 1:2, and 1:3 respectively. Such smaller and elongated shape of anchors fits better to the nature of text instances in general as discussed in Section 2.3.2 (c).

4.2.2 (b) Training Strategy

The training objective of Polygon-FRCNN's RPN module is consistent with the study of J. Huang et al. (2016)[2]: optimizing the multi-task loss function as described in Equation

[2]Multiple optimization were introduced since the publication of the original Faster-RCNN study (Ren et al. (2015)), hence the proposed model opts for the optimized version of the instead.

4.2:

$$L(p_i,t_i) = \alpha \frac{1}{N_{mini-batch}} \sum_i L_{cls}(p_i, p_i^*) + \beta \frac{1}{N_{mini-batch}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (4.2)$$

where i is the index of anchor in a mini-batch, while p_i and p_i^* are the predicted probability and ground truth label of anchor i being a text instance. On the other hand, t_i represents the predicted parameterized target (t_x, t_y, t_h, and t_w) as described in Equation 4.7 and t_i^* is its ground truth counterpart. $N_{mini-batch}$ is the mini-batch number, acts as a normalizer for both of the loss functions. α and β are the regularization terms to control the weightage of both loss terms. All of these notions will be detailed as the explanation proceed. Firsly, at each iteration, the anchors go through a matching process with the ground truth (that is associated with the input image at that particular iteration) to determine which anchors are positive and which are negative based on the following conditions:

$$Matching(box_a, box_{gt}) = \begin{cases} Positive & \text{if IoU}(box_a, box_{gt}) >= 0.7 \quad (4.3) \\ Negative & \text{if IoU}(box_a, box_{gt}) < 0.3 \quad (4.4) \\ Neutral & \text{otherwise} \quad (4.5) \end{cases}$$

Both positive and negative labelled anchors contribute to the classification loss (L_{cls}) as described in Equation 4.6. Meanwhile, the neutral anchors do not contribute to the loss calculation at all.

$$L_{cls}(p, p^*) = -(p^* log(p) + (1 - p^*) \log(1 - p)) \quad (4.6)$$

where p^* is the ground truth label (positive anchors are assigned with '1' and negative anchors are assigned with '0'), and p is the predicted probability ranged between 0 to 1. Such detail can be seen in Figure 4.3 as well. On the regression end, only positive anchors contribute to the regression loss as described in Equation 2.1. It can be thought of as an implicit switching mechanism, which is controlled by p_i^* in the second part of

Equation 4.2. All the positive labelled anchors and its matching ground truths are then parameterized with Equation 4.7 to form the regression targets for the training process. Intuitively, the network is supervised with the offset between the anchor and the actual location of text instance.

$$\begin{aligned} t_x &= (x - x_a)/w_a, \\ t_y &= (y - y_a)/h_a, \\ t_h &= \log(h/h_a), \\ t_w &= \log(w/w_a), \end{aligned} \tag{4.7}$$

where variable x and xa represent the ground truth box and anchor box respectively (same applies to y, h, and w). Both the L_{cls} and L_{reg} losses are normalized by $N_{mini-batch}$. Consistent with the study of J. Huang et al. (2016), $N_{mini-batch}$ is set to 256 for each image. In a natural image, there are usually more background regions than text regions. A subsampling mechanism is implemented in the framework to control the balance between both, ensuring both text and background have at most 128 training samples each. Lastly, α and β are set to 1 and 2 respectively as the L_{reg} has more variables, hence was allocated with more weightages in the final loss calculation.

4.2.2 (c) Proposal

The objective of RPN is to propose regions of interest (ROIs) for the next stage of the network. The number of proposal is another hyperparameter of the Polygon-FRCNN model, as inherited from Faster-RCNN. Following the studies of Ren et al. (2015) and J. Huang et al. (2016), the proposed method is configured to produced 300 proposals for each input image. Both studies agreed that increasing the number of proposals does not improve the performance but would slow down the execution speed instead. The

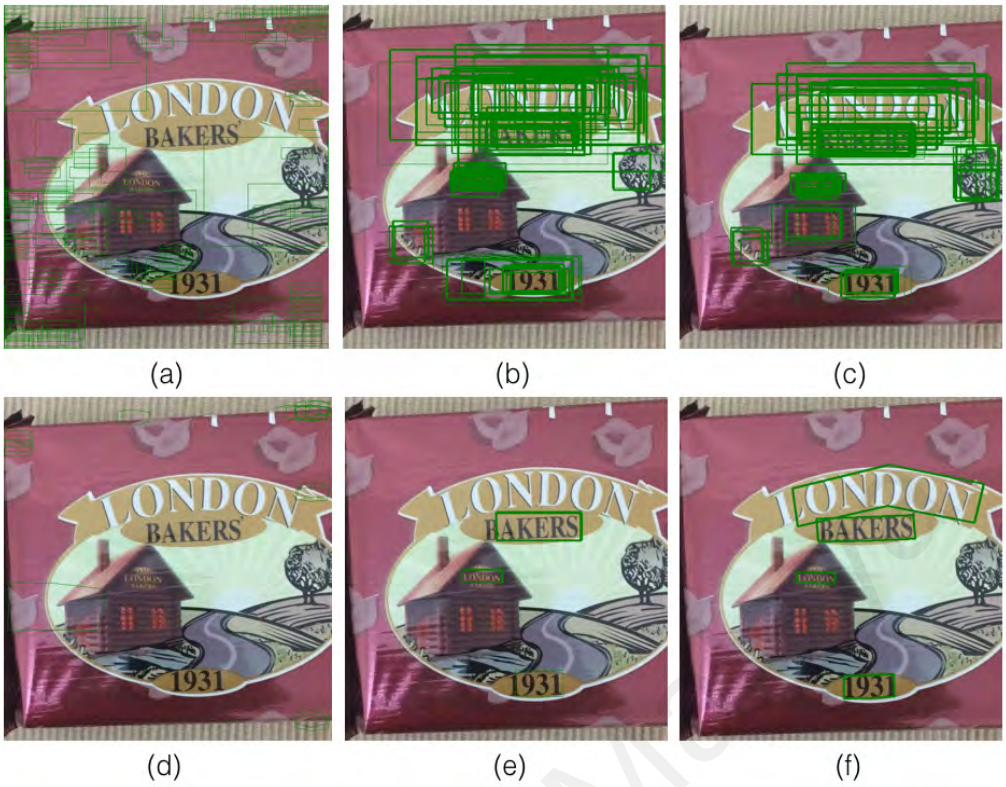


Figure 4.4: Proposals ((a) - (c)) and detections ((d) - (f)) produced from RPN and PCN respectively at the end of different training iterations. (a) and (d): 0 iteration, (b) and (e): 100,000th iteration, (c) and (f): 200,000th iteration. The width of the bounding lines indicates the confidence level of each proposal or prediction, wider the line, higher the confidence.

raw proposals are first filtered by the Non-Maximal Suppression (NMS) algorithm then clipped with their classification scores. The NMS algorithm is a straightforward way to eliminate overlapping proposal boxes, which is explained in the Algorithm 2. In the case where there are more than 300 proposal left after the NMS filtering process, the top-300 proposals based on their classification scores will be retained for the next stage. Figure 4.4(c) visualizes the proposals produced by RPN (after completing the training process) upon feeding in an input image.

4.2.3 Polygon Classifier Network

The objective of Polygon Classifier Network (PCN) is to refine coarsely proposed boxes to produce precise polygons. The design of PCN is based on Box Classifier Network

Algorithm 2 Non-maximum-surpression algorithm for the proposal boxes filtering process.

Let the input proposal boxes be $proposal_n$
Let the input classification scores be $scores_n$
Let the processed proposal be $processedProposals$
Let iouThreshold = 0.6
Sort $scores_i$ descendingly
Sort $proposal_i$ based on the ID of sorted $scores_i$
Remove $proposal_0$ from $proposal_i$
while length($proposal_i$) > 1 **do**
  for all all i such that $0 < i < n$ **do**
    $IoU = (proposal_0 \cap proposal_i)/(proposal_0 \cup proposal_i)$
    if IoU >= iouThreshold **then**
      Remove $proposal_i$ from $proposal_n$
    else
      Continue
    end if
    Append the $processedProposals$ with $proposal_0$
  end for
end while
Append the $processedProposals$ with the last proposal in $proposal_n$

(BCN) from the original Faster-RCNN architecture. On top of the traditional bounding box regression head, PCN is modelled to regress polygon shaped bounding region as well. The lower level details to adapt such change are detailed in this section. As illustrated in Figure 4.1, the PCN takes both the feature maps and proposal boxes as input. Firstly, the shared feature is cropped based on the proposed regions from the RPN. Since proposal boxes come in arbitrary size, a resize operation is required before feeding them into the fully-connected layers. Hence, all the cropped regions will be resized to 17x17 spatially, while retaining the third dimension size. Next, the cropped and resized proposal regions will be fed into the average pooling layer, through the fully-connected layer to the final prediction layer. The final output is two vectors representing i) the confidence score of the particular proposal being a text instance and ii) the combined parameters of a bounding box and N-vertex polygon bounding region. All of these operations are detailed in Figure 4.5 for better visualization as well.

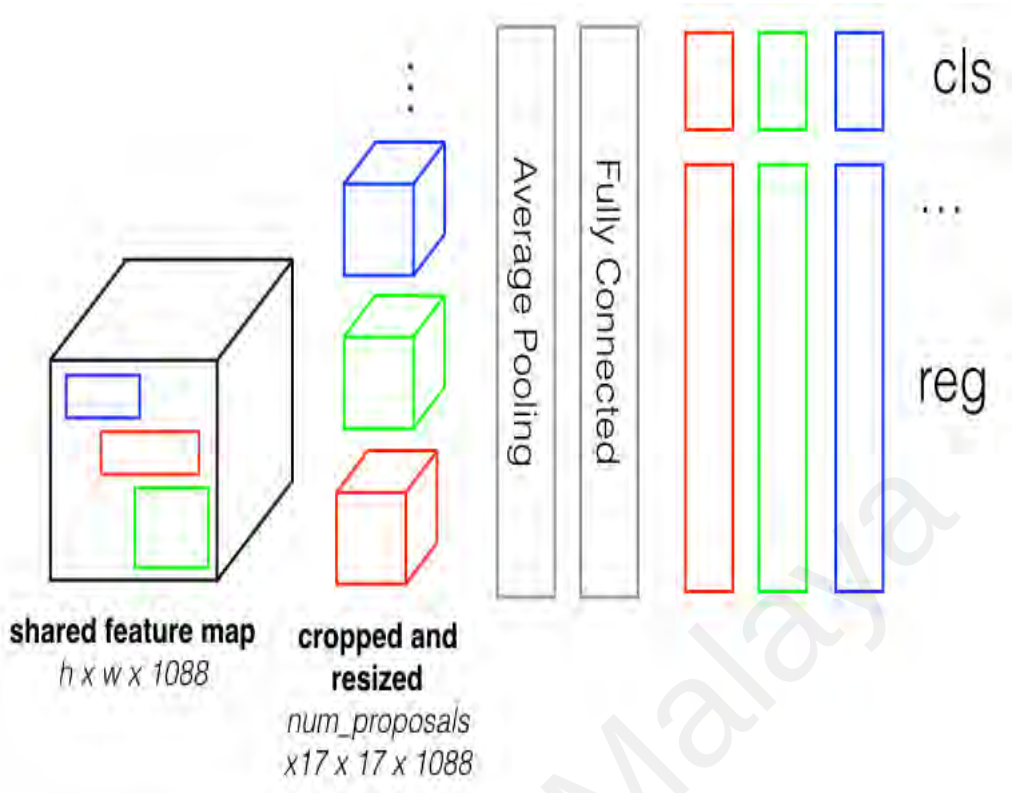


Figure 4.5: Architecture of Polygon Classifier Network.

4.2.3 (a) Encoded Polygon Regression Target

The convention regression target (x_m y_m, w, h)[3] used in Faster-RCNN, SSD, and YOLO only applies to axis-aligned rectangular boxes, but not polygon. Y. Liu & Jin (2017) introduced an encoding method for quadrilateral and claimed that relative information is easier to train instead of absolute values of quadrilateral vertices (i.e., (x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)). In fact, most of the regression-based scene text detectors were modelled to regress one kind of encoded quadrilateral or another. Table 2.2 tabulates all the existing state-of-the-art scene text detectors and their regression target. Inspired by these studies, this work introduces a polygon encoding method, which translates absolute polygon vertices into relative information for Polygon-FRCNN's regression training.

Encode

The proposed encoding is an extension to the encoding used to represent curved text with

[3]Subscripts 'm' stands for the middle point between x_{min}, y_{min} and x_{max}, y_{max}

two straight lines in Section 3.5.1 earlier. The mentioned section introduced Equation 3.2 and 3.3, with the function of converting polygon vertices into $(x_{mid_0}, y_{mid_0}, dx_1, dy_1, dx_2, dy_2)$. However these parameters are not reversible (i.e., decode back to proper polygon vertices) due to the loss of information. On top of Equation 3.2 and 3.3, Equation 4.8 is needed to form a reversible encoding method. The said equation is also visualized in Figure 3.16b.

$$\begin{aligned} h_{(i)} &= abs(y_{(i)} - y_{((N-1)-i)}) \\ w_{(i)1} &= x_{(i)} - x_{mid(i)} \\ w_{(i)2} &= x_{((N-1)-i)} - x_{mid(i)} \\ For\ i &= [0, N/2) \end{aligned} \tag{4.8}$$

Similar to the notation in Equation 3.2, 'i' and $((N-1)-i)$ translates to the opposing pairs of vertices, where N is the total number of vertices. Note that the encoding method is compatible to any polygon with *even* number vertices. However, for the sake of simplicity, all the explanation henceforth assume the 6-vertex polygon, as introduced in Section 3.5.1, unless stated otherwise. Hence, utilizing the same example as earlier (i.e., taking the $\{x_0, y_0\}$ and $\{x_5, y_5\}$ pair as example):

1) $h_{(i)}$ is the height between y_0 and y_5,

2) $w_{(0)1}$ is the width between $x_{mid(0)}$ and x_0, and

3) $w_{(0)2}$ is the width between $x_{mid(0)}$ and x_5.

The reason for two width values ($w_{(i)1}$ and $w_{(i)2}$) is to ensure that the model is flexible enough to cover the cases of $x_i > x_{N-i-1}$ and $x_i < x_{N-i-1}$, which are pretty common in the shape of curved text. As a result, the polygon ground truth with the original form of (x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5) is encoded into fifteen relative information

($y_{mid(0)}$, $x_{mid(0)}$, $h_{(0)}$, $w_{(0)1}$, $w_{(0)2}$, $dy_{mid(1)}$, $dx_{mid(1)}$, $h_{(1)}$, $w_{(1)1}$, $w_{(1)2}$, $dy_{mid(2)}$, $dx_{mid(2)}$, $h_{(2)}$, $w_{(2)1}$, $w_{(2)2}$). Intuitively, the network is tasked to learn and regress the text line location as depicted in Figure 3.16d.

Decode

Decoding process is needed to convert the predictions to actual polygon vertices. During decoding, all the middle points from second points onwards (i.e., $i = [1, N/2)$) will first be determined with Equation 4.9.

$$\begin{aligned} x_{mid(i)} &= x_{mid(i-1)} + dx_{mid(i)} \\ y_{mid(i)} &= y_{mid(i-1)} + dy_{mid(i)} \\ For\ i &= [1, N/2) \end{aligned} \tag{4.9}$$

Then h_i, $w_{(i)1}$, and $w_{(i)2}$ will be used to transform these middle points to N vertices using Equation 4.10-4.11.

$$\begin{aligned} x_{(i)} &= x_{mid(i)} - w_{(i)1} \\ y_{(i)} &= y_{mid(i)} - h_{(i)}/2 \\ For\ i &= [0, N/2) \end{aligned} \tag{4.10}$$

$$\begin{aligned} x_{(i)} &= x_{mid(i)} + w_{(i)2} \\ y_{(i)} &= y_{mid(i)} + h_{(i)}/2 \\ For\ i &= [N/2, N) \end{aligned} \tag{4.11}$$

Again with the example in Figure 3.16a, $y_{(0)}$, $y_{(1)}$, and $y_{(2)}$ will always be the opposite of $y_{(5)}$, $y_{(4)}$, and $y_{(3)}$ respectively with the height values: $h_{(0)}$, $h_{(1)}$, and $h_{(2)}$ (likewise for $x_{(i)}$

with $w_{(i)1}$ and $w_{(i)2}$).

4.2.3 (b) Parameterization of Anchor Polygon

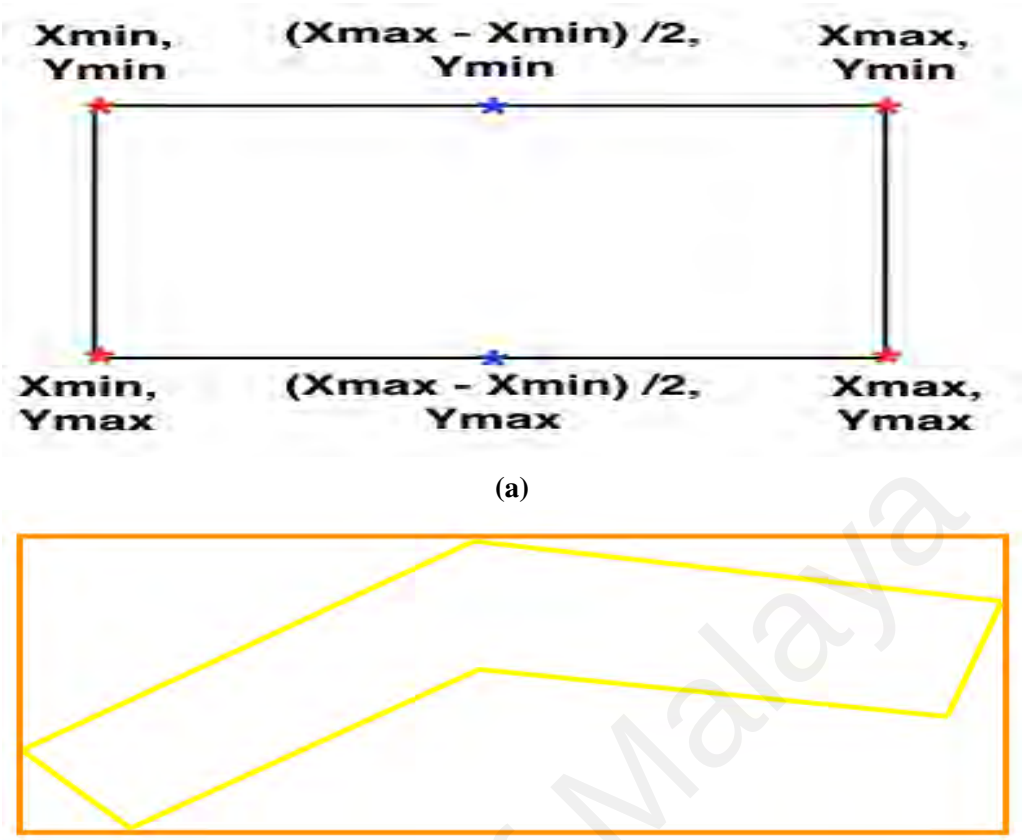
In contrast to RPN, PCN outputs polygon vertices. Implementation wise, the ground truths in polygon format are first converted to the fifteen encoded values as explained in Section 4.2.3 (a). Then, in line with the parameterization with Equation 4.7, the ground truths will be matched and parameterized with the proposal boxes from RPN (instead of the anchor boxes). However, there are two problems in this process: i) the matching between polygons requires much more complicated process than the rectangular box, and ii) the axis-aligned proposal boxes is not compatible with the polygon ground truths for the parameterization.

Matching between Axis-aligned Proposal Boxes and Ground-truth Polygons

The algorithms of identifying overlapping region that involves a polygon is complicated (Sutherland & Hodgman (1974); Vatti (1992); Weiler & Atherton (1977)). They usually consist of heuristic rules and multiple processes which would slow down the execution time significantly. Such trait is not desirable in the design of a modern scene text detector as minimal execution speed is often demanded. The design of Polygon-FRCNN sidesteps this problem with an approximation solution: converting polygon to a circumscribed rectangle. Figure 4.6b shows the said circumscribed rectangle of a polygon. Consequently, the problem is simplified to just finding the overlapping region between two rectangles. Note that such conversion only applies to the matching process, the ground truths remain in polygon format for the following process.

Parameterization between Proposal and Ground-truth Polygons

Upon identifying the matching proposal and ground truth pairs, the next step is to parameterized them. Unlike the original parameterization equation as stated in Equation



(a)

(b)

Figure 4.6: Methods employed in the PCN matching and parameterization process: (a): increasing the number of vertices of proposal box to match the 6-vertex polygon ground truth. (b): converting polygon (yellow) to a circumscribed rectangle (orange) for the matching process.

4.7, which was derived for encoded axis-aligned box only, this work extended it to fit the proposed encoded polygon. Firstly, the axis-aligned bounding box outputs (four vertices, $(x_{min}, y_{min}, x_{max}, y_{max})$) from the RPN need to be converted to N-vertex polygon so that both have equal variables to be parameterized. The conversion process is a simple operation which adds interval points to the original axis-aligned proposal box. The said process can be visualized in Figure 4.6a, the original 4-vertex bounding box (represented by four red '*') is added with two middle points (represented by two blue '*'), which in turn became compatible with the 6-vertex polygon ground truth for the paramerization

process as formulated in Equation 4.12.

$$
\begin{aligned}
t_{x_{mid(0)}} &= (x_{mid(0)} - xa_{mid(0)})/w_a \\
t_{y_{mid(0)}} &= (y_{mid(0)} - ya_{mid(0)})/h_a \\
t_{h_{(i)}} &= \log(h_{(i)}/h_{a_{(i)}}) \\
t_{w_{(i)1}} &= (w_{(i)1} - w_{a_{(i)1}})/w_{a_{(i)1}} \\
t_{w_{(i)2}} &= (w_{(i)2} - w_{a_{(i)2}})/w_{a_{(i)2}} \\
For\ i &= [0, N/2) \\
t_{dx_{mid(j)}} &= (dx_{mid(j)} - dx_{a_{mid(j)}})/w_a \\
t_{dy_{mid(j)}} &= (dy_{mid(j)} - dy_{a_{mid(j)}})/h_a \\
For\ j &= [1, N/2)
\end{aligned}
\tag{4.12}
$$

where x and x_a represent the encoded predicted polygon (also the encoded ground truth polygon) and encoded anchor polygon respectively (same applies to y, h, w_1, w_2, dx, and dy).

4.3 Implementation Details

Polygon-FRCNN was built on top of the Tensorflow object detection API (J. Huang et al. (2016)). All of the training and experiments were ran on a machine with Intel 6-core Xeon chip, 32GB of RAM, Nvidia Titan X Pascal Architecture GPU, and Ubuntu OS v14.04.

4.3.1 Variants of Polygon-FRCNN

As discussed in Section 3.5.1, six vertices are the minimum amount of vertices required to cover a curved text in *Total-Text*. Hence, Polygon-FRCNN-3 is designed to produce six vertices polygon bounding region. The proposed encoding method splits six vertices

into three opposing pairs, hence the notation '3'. The overview of Polygon-FRCNN-3 architecture can be seen in Figure 4.1. It is not perfect as some of the bigger curved text instances require more than six vertices to fit. In such cases, the bounding region will be looser in order to fit in the entire text region. The 6-vertex polygon as explained in Section 3.5.1 is used as the training ground truth for Polygon-FRCNN-3. On the other hand, Polygon-FRCNN-5 is the scaled-up version of 3, which has ten more parameters in the regression head. Polygon-FRCNN-5 regresses five pairs of opposing vertices. It was trained with the controlled polygon groundtruth mentioned in Section 3.4.2 (d). Polygon-FRCNN-5 is the scene text detector in the *T3* assisted annotation tool as described in Section 3.7.

4.3.2 Training Process

4.3.2 (a) Pre-Training

The weights of Polygon-FRCNN was initialized with the ImageNet-pretrained weights.[4] Such initialization is important as it gives the network a general idea on what to look for since both ImageNet and scene text datasets are made up of images in natural scene. Additionally, training such a deep network from scratch is very time consuming (weeks of time), hence initializing a network with pre-trained model became a norm in the deep learning community (Sharif R. et al. (2014); Yosinski et al. (2014)).

Upon initialization, Polygon-FRCNN was trained on COCO-Text for 100K iterations. It is a good practice to pre-train a deep network on a larger dataset first, before the actual training data which is smaller in scale (*Total-Text* in this case). Doing so would prevent overfitting, and help the network to converge faster (Yosinski et al. (2014)) during the actual training. Utilizing the extra annotation of COCO-Text, roughly 15K images

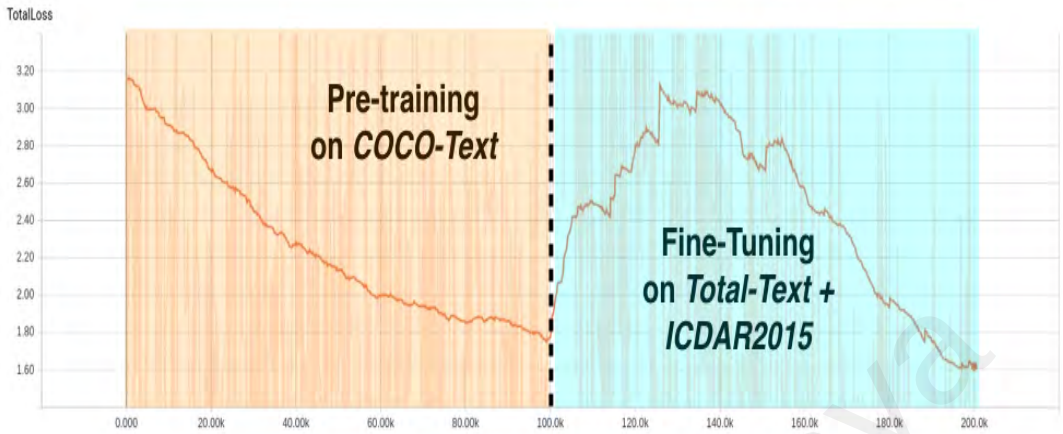
[4]Available in https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

with at least one legible text instance were retrieved for the pre-training process. Consistent with J. Huang et al. (2016)'s study, the learning rate was set to 0.003 during the pre-training stage.

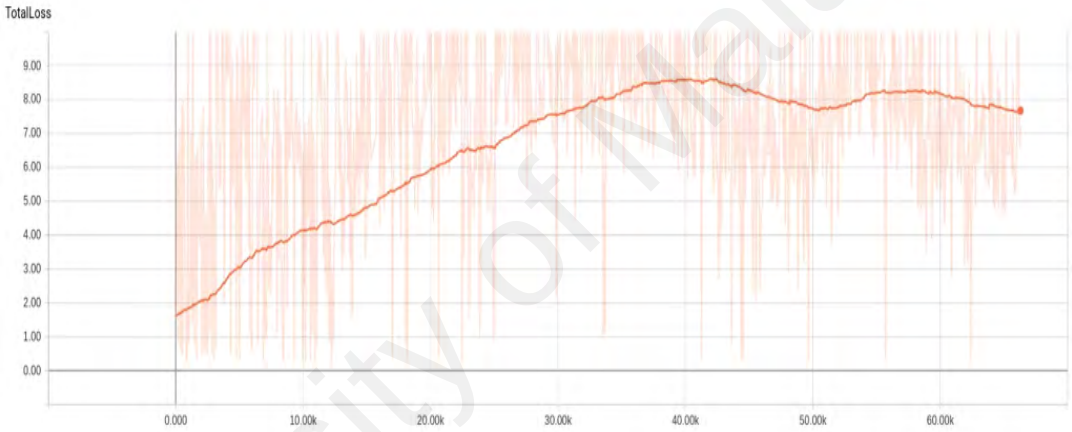
The first part of Figure 4.7a (shaded in orange) shows the training loss of the first 100K pre-training steps. The total loss, as modelled in Equation 4.2, decreases as the training proceeds indicates that the network was getting better at the task. The performance graph in Figure 4.7c validates this with the increasing score as well. Meanwhile, the importance of pre-training process is illustrated in Figure 4.7b, which shows high training loss (stagnant at the range of 8-9 in compared to 2-3(in Figure 4.7a)) with Polygon-FRCNN training on the combination of *Total-Text* and ICDAR2015 without going through the pre-training process. The pre-training process was terminated at 100K iteration when the evaluation score stopped improving further at roughly 0.44 IoU in average across the validation set of COCO-Text. On top of that, Figure 4.4(a-b) and (d-e) visualises the outputs from RPN and PCN before(a,d) and after(b,e) this pre-training process. It can be observed that the network has learnt to propose better regions and produce better predictions. However, the final prediction after this training process is not ready for the curved text region yet. As we can see in Figure 4.4(d), it failed to detect the curved text instance ('LONDON') in this example. More results can be found in Appendix B, as observed, when it successfully predicts the location of the curved text, the outputs still take the shape of normal rectangular. This is mainly because the ground truth format employed in COCO-Text is axis-aligned bound box, the network has yet to see the actual polygon shaped ground truth.

4.3.2 (b) Fine-Tuning

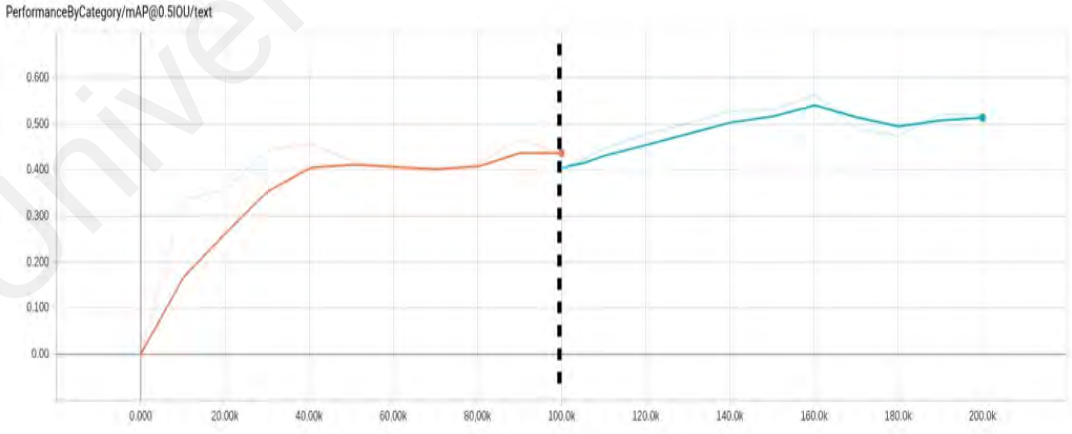
After the pre-training process, Polygon-FRCNN was fine-tuned on the combination *Total-Text* and ICDAR2015 training set for another 100K iterations. The merging of dataset



(a) Training loss of Polygon-FRCNN.



(b) Training loss of Polygon-FRCNN without pre-training it on COCO-Text.



(c) Performance of Polygon-FRCNN throughout the training

Figure 4.7: Losses and performance of Polygon-FRCNN throughout the training process.

was done to increase the scale of training data, which was proven benefitial to the training of deep neural network models (LeCun et al. (1998)). The combined dataset has 2000 training images in total, with 1000 images each from the training set of *Total-Text* and ICDAR2015. 255 images from the *Total-Text* training set was withhold to serve as the validation set, which was used to monitor the performance of the network during the training process, as plotted in Figure 4.7c. Yosinski et al. (2014)'s work suggested to reduce the learning rate when fine-tuning a network on a much smaller scale dataset, hence it was decreased to 0.0003 for this stage of the training process. As shown in Figure 4.7a, the loss increases abruptly at the 100K mark, when the fine-tuning process starts. That sudden increase was caused by the newly introduced training data, the network has to adjust to the emergence of curved text training data. The loss started to decrease again around the 130K mark and reached a plateau around 195K iteration. The performance of the network on the validation set is seen to improved gradually until the 160K iteration mark, and saturated around 0.5-0.6 for the rest of the training. Figure 4.4 demonstrates the importance of the availability of the curved text data in *Total-Text*. As seen in Figure 4.4(e), after the fine-tuning process, the network has learnt to produce polygon bounding region.

4.4 Experiments

Several experiments were carried out to evaluate the effectiveness of Polygon-FRCNN. This section describes all the experiment configurations, results, and some of the insights drawn upon them.

4.4.1 Datasets

Polygon-FRCNN was benchmarked on ICDAR2013, ICDAR2015, and *Total-Text*. ICDAR2013 and ICDAR2015 were chosen to demonstrate the performance of Polygon-

FRCNN on horizontal and multi-oriented text respectively.

4.4.2 Evaluation Protocol

In order to ensure the legitimacy of the comparison carried out in this section, all the benchmarkings were done following the standard evaluation protocol and configuration of ICDAR2013 and ICDAR2015. Firstly, all the results on Table 4.2 were obtained using the DetEval protocol with the default thresholds (i.e. tr=0.8 and tp=0.4) (Wolf & Jolion (2006)). On the other hand, the Pascal VOC evaluation method is the official protocol for ICDAR2015. The results on Table 4.3 were recorded with the standard 0.5 IoU threshold. Both DetEval and Pascal VOC methods were made available for *Total-Text*. The threshold values for DetEval, tr and tp were configured to 0.7 and 0.6 respectively based on the findings as described in Section 3.6. Meanwhile, for the Pascal VOC evaluation method wise, the standard 0.5 IoU threshold was chosen.

4.4.3 Quantitative Result

Table 4.2 and 4.3 show that the performance of Polygon-FRCNN-3 (our best performing model) is on par with contemporary state-of-the-art works like Jiang et al. (2017); Liao et al. (2017); Y. Liu & Jin (2017); Ma et al. (2017); B. Shi et al. (2017). On the other hand, Table 4.4 tabulates the performance of Polygon-FRCNN on *Total-Text*. In order to measure the effectiveness Polygon-FRCNN againsts of text with different orientations, especially curved text, two subsets of *Total-Text* were created with the orientation annotation. The 'curved-set' in Table 4.4 consists of only curved text; while the 'non-curved-set' is made up of horizontal and multi-oriented text. There are three methods being compared on *Total-Text*: Box-FRCNN, Polygon-FRCNN-3, and Polygon-FRCNN5. Box-FRCNN is the **axis-aligned box** output from the same network as Polygon-FRCNN-3, being compared to show the effectiveness of the **polygon** predicting Polygon-FRCNN-3. Mean-

Table 4.2: Evaluation Result on ICDAR2013

Methods	Precision	Recall	F-measure
TextBoxes (Liao et al. (2017))	0.89	**0.83**	0.86
R^2CNN (Jiang et al. (2017))	**0.94**	**0.83**	**0.88**
Seglink (B. Shi et al. (2017))	0.88	**0.83**	0.85
RRPN (Ma et al. (2017))	0.90	0.72	0.80
DIRECT (W. He et al. (2017))	0.92	0.81	0.86
Polygon-FRCNN-3	0.89	0.81	0.85

while, Polygon-FRCNN-5 is the scaled up version as explained in Section 4.3.1.

As observed in Table 4.4, Polygon-FRCNN-3 outperforms its box predicting counterpart (Box-FRCNN) by 0.16 (0.12) in terms of F-measure overally (wholeset). Their performance differences became much more significant on the curved text only subset of *Total-Text*, with a gap of 0.3 (0.2). This result demonstrates that the design of the proposed method is effective in detecting curved text. The performance of Polygon-FRCNN-5 drops slightly by 0.04 (0.03) in terms of F-measure in compared to Polygon-FRCNN-3. The scaled up version is expected to bind curved text region smoothly with more vertices; however, the increase in number of vertices also reduces the error margin of each vertices, which causes the loss in performance.

Another worthy observation is that the performance of the proposed method is lower on *Total-Text* than on both of the ICDAR2013 and ICDAR2015 datasets. Visualized comparison can be seen in Figure 4.8. This observation shows that *Total-Text* is more challenging than both of the well-received scene text datasets in the text community. Lastly, the execution speed of Polygon-FRCNN is 1.47s in average across all 300 testing images in the proposed dataset.

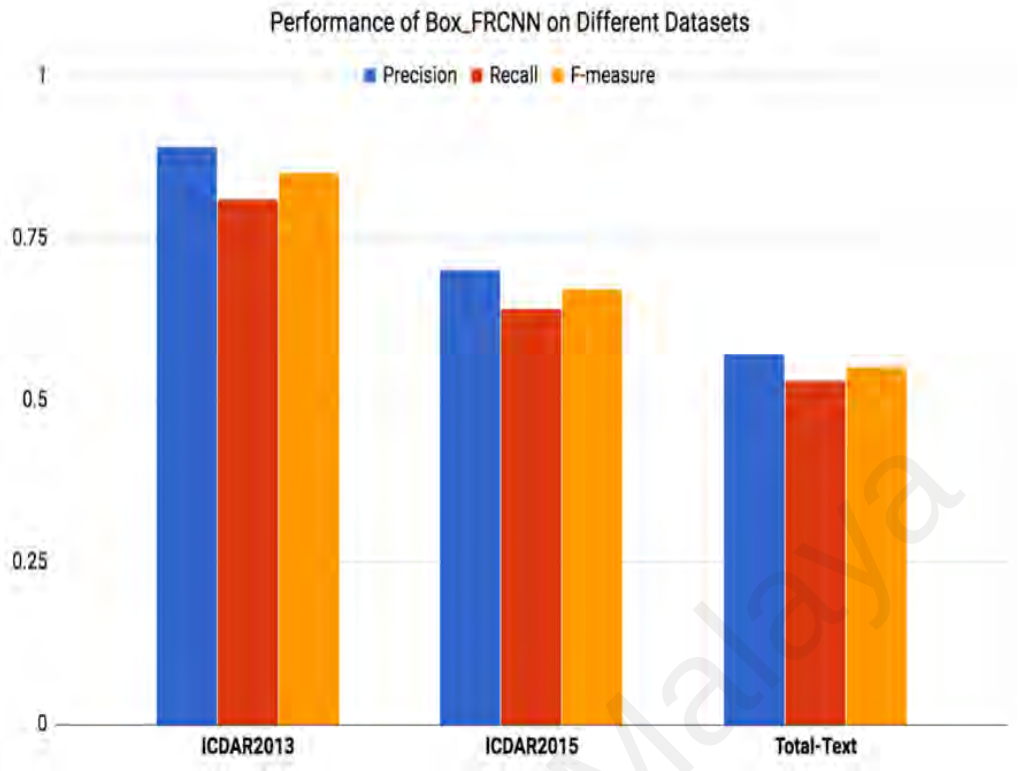


Figure 4.8: Performance of Box-FRCNN on different datasets. It achieves the best scores in terms of all three metrices on ICDAR2013, followed by ICDAR2015, lowest on *Total-Text*.

Table 4.3: Evaluation Result on ICDAR2015

Methods	Precision	Recall	F-measure
DMPNet (Y. Liu & Jin (2017))	0.73	0.68	0.70
R^2CNN (Jiang et al. (2017))	**0.86**	0.80	**0.83**
Seglink (B. Shi et al. (2017))	0.73	0.77	0.75
RRPN (Ma et al. (2017))	0.73	**0.82**	0.77
DIRECT (W. He et al. (2017))	0.82	0.80	0.81
Polygon-FRCNN-3	0.76	0.67	0.72

4.4.4 Qualitative Result

The detection examples of Polygon-FRCNN-3 can be seen in Figure 4.10, 4.11, 4.12, 4.13, and 4.14. The contrary examples of text instances in both Figure 4.10 and 4.11 illustrate that Polygon-FRCNN-3 is robust against text with different sizes, orientations and perpestive distortion.

Table 4.4: Evaluation of Polygon-FRCNN on Total-Text. All the models were evaluated on both DetEval and Pascal VOC (in the bracket) evaluation protocol. Legends: P = Precision, R = Recall, F = F-measure.

Methods	wholeset		
	P	R	F
Box-FRCNN	0.50(0.57)	0.46(0.53)	0.48(0.55)
Polygon-FRCNN-3	**0.67**(**0.69**)	**0.61**(**0.64**)	**0.64**(**0.67**)
Polygon-FRCNN-5	0.64(0.69)	0.56(0.6)	0.6(0.64)
	curved-set		
Box-FRCNN	0.19(0.3)	0.17(0.29)	0.18(0.3)
Polygon-FRCNN-3	**0.51**(**0.52**)	**0.45**(**0.48**)	**0.48**(**0.50**)
Polygon-FRCNN-5	0.44(0.5)	0.38(0.44)	0.41(0.47)
	non-curved-set		
Box-FRCNN	0.50(0.51)	0.58(0.61)	0.54(0.55)
Polygon-FRCNN-3	0.63(0.64)	**0.59**(**0.62**)	**0.61**(**0.63**)
Polygon-FRCNN-5	**0.63**(**0.65**)	0.57(0.59)	0.59(0.62)

On top of that, Figure 4.12 shows that the polygon output of Polygon-FRCNN-3 is capable of binding text of all orientations tightly. Meanwhile, Figure 4.9 emphasizes the effectiveness of polygon output in compared to box output when it comes to curved text instances. Both of these visual examples validate the performance advantage of Polygon-FRCNN-3 against Box-FRCNN as tabulated in Table 4.4.

On the other hand, Figure 4.13 visualizes the difference between Polygon-FRCNN-3 and the scaled up Polygon-FRCNN-5. The detection output of Polygon-FRCNN-5 is observed to bind curved text with larger curvature more tightly as expected.

Lastly, Figure 4.14 shows that the proposed baseline for Total-Text, Polygon-FRCNN-3 is not perfect. Polygon has more vertices than axis-aligned bounding box and quadrilateral (i.e., six for Polygon-FRCNN-3, in compared to two and four for axis-aligned bounding box and quadrilateral respectively), which makes it more prone to error. As

Figure 4.9: Comparison between detection examples of Box-FRCNN and Polygon-FRCNN.

observed in Figure 4.14, although the proposals were mostly right about the location of the text, but several errors in the vertices regression caused the polygon to miss the proper text region. Other failure cases include the shortcoming of Polygon-FRCNN-3 to bind text with larger curvature tightly (1st row, 2nd column), and complete miss detections (1st row, 3rd column).

4.5 Summary

As a summary, Polygon-FRCNN is a scene text detector that was designed with curved text in consideration. It was built on top of the groundbreaking object detection framework, Faster-RCNN. Changes of design were made to equip it with the ability to produce polygon bounding region, which is capable of detecting curved text. One of the main changes is the introduction of the polygon encoding method to prepare for the network regression target. Multiple experiments were conducted to evaluate the performance of the

Figure 4.10: Detection examples of Polygon-FRCNN-3 on ICDAR2013.

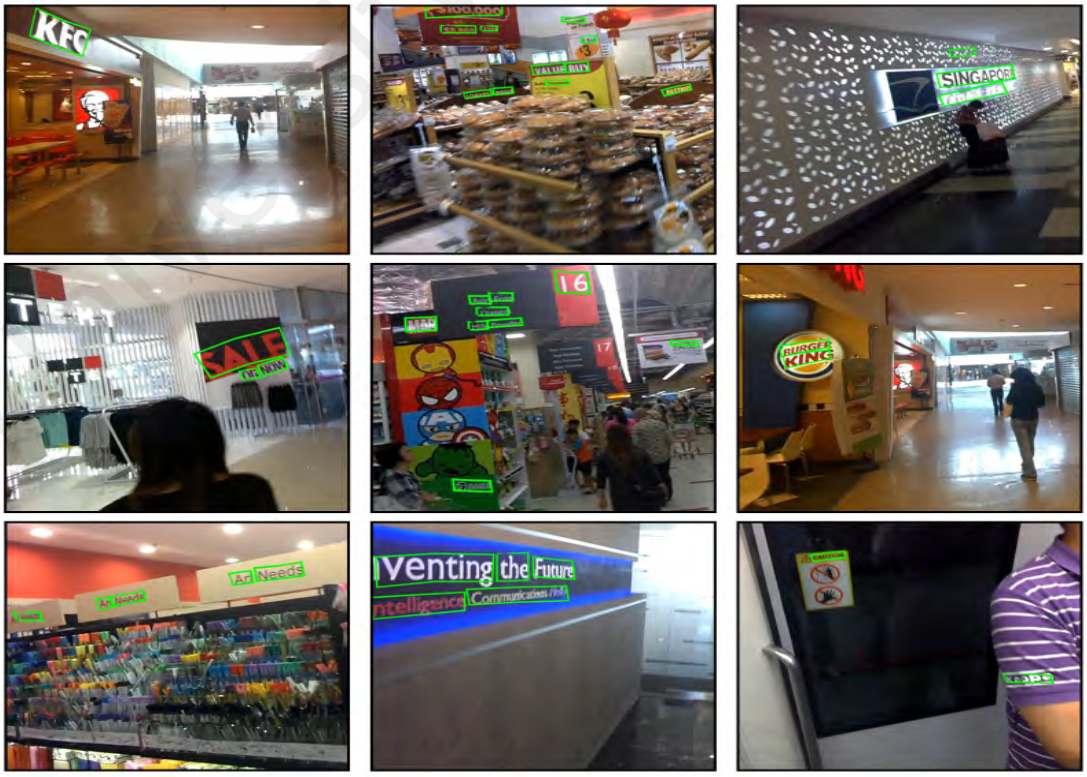


Figure 4.11: Detection examples of Polygon-FRCNN-3 on ICDAR2015.

Figure 4.12: Successful detection examples of Polygon-FRCNN-3 on Total-Text. It shows that polygon shaped output is effective against text of all orientations.

Figure 4.13: Left: Polygon-FRCNN-3; Right: Polygon-FRCNN-5. Polygon-FRCNN-5's output has more vertices, is able to bind curved text with larger curvature more tightly.

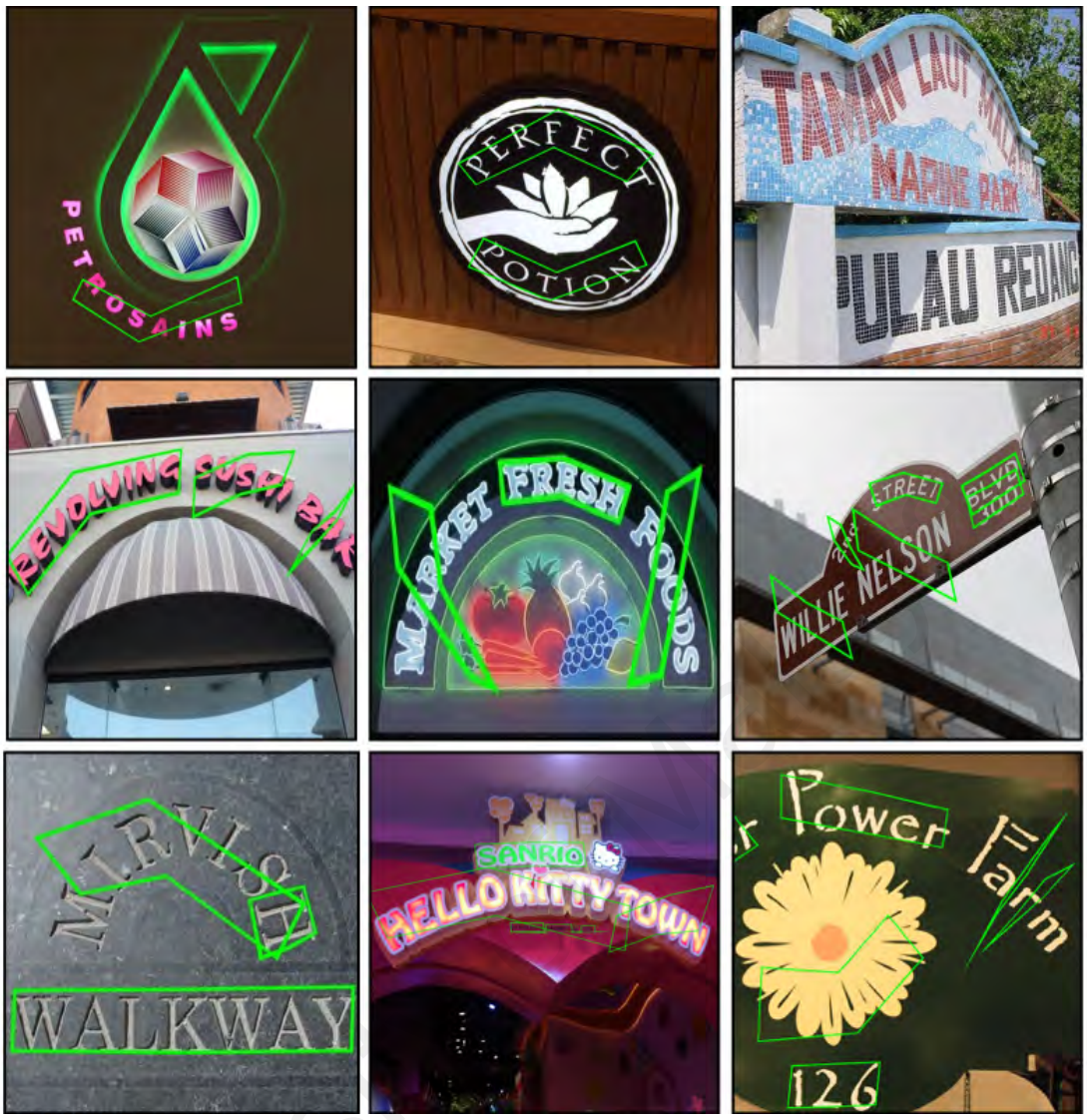


Figure 4.14: Failure detection examples of Polygon-FRCNN-3 on Total-Text.

proposed method. The benchmark result on ICDAR2013 and ICDAR2015 have shown that Polygon-FRCNN is on par with the state-of-the-art scene text detectors. Moreover, it has proven to be a strong baseline for the proposed dataset, *Total-Text*. The effectiveness of polygon bounding output against text of all orientations is shown with the result breakdown of the *Total-Text* dataset as well. Last but not least, the significance of the curved text data in the proposed dataset was presented as well.

CHAPTER 5: CONCLUSIONS

This chapter concludes this thesis work with a summary of what was done and presented, current limitations, and potential future research works.

5.1 Summary

This research set out to fill up the gap of the existing scene text community: the lack of curved text data in the scene text datasets. A new scene text dataset, *Total-Text*, was collected to achieve this objective. All 1555 images were carefully annotated with three kinds of ground truth for three different tasks: detection, recognition, and segmentation. The entire dataset is currently available at `https://github.com/cs-chan/Total-Text-Dataset`. The availability of *Total-Text* hopes to catch the field's attention to this under-research problem.

Apart from collecting images and ground truth, there are several more aspects that are crucial in facilitating a new research direction. The second objective of this work is to make sure every of these aspects are carefully chosen and tuned to lay a concrete foundation to kick start this new research direction. Firstly, the proposed regulated polygon that is used for the ground truth of detection task is capable of binding text of all orientations tightly. On top of that, the proposed annotation process for the regulated polygon reduces human subjectiveness with a guidance mechanism. Next, a polygon encoding method was introduced to represent curved text numerically for analysis purposes. It is observed that the curved text in *Total-Text* can be broadly categorised into four large clusters. Finer details within each cluster can also be observed through the differences of each curved text in terms of its curvature. Next, the default thresholds of Deteval evaluation protocol were fine-tuned for a fairer evaluation on *Total-Text*. The fine-tuning process were empirically validated with experiment results. Lastly, an assisted annotation framework was

introduced to reduce annotation time, hoping to aid in future dataset scaling up effort.

The third and last objective of this thesis is to propose a scene text detector that is robust against text of all orientations that natural scenes exhibits. *Polygon-FRCNN* is a straightforward extension of the *Faster-RCNN* object detection framework. Its performance on the renowned datasets like ICDAR2013 and ICDAR2015 is on par with the existing state-of-the-art scene text detectors. In addition, the evaluation of it on *Total-Text* have demonstrated its ability in detecting and binding text of all orientations, including curved text which is a challenge for the exisiting higher performing scene text detectors.

5.2 Limitations

Both the proposed dataset and scene text detector possess several limitations. Firstly, although the scale of *Total-Text* is similar to ICDAR2015 and CTW1500, but it is significantly smaller in compared to COCO-Text. Such limitation is caused by one main bottleneck: the annotation effort in producing polygon ground truth is more time consuming than the conventional axis-aligned bounding box. Since deep learning based methods appeared to be the direction which the community has embarked on, the lack of such training data would greatly hindled the generalization of these data-dependent model. On top of that, the evaluation carried on small dataset can only be taken seriously to a limited extent.

As presented in Section 4, Polygon-FRCNN has rooms for improvement. The Polygon-FRCNN-3 variant's six vertices output is not tight enough for larger curve text; while Polygon-FRCNN-5 shows improvement qualitatively in that respect, the higher count of parameters to be regressed make it more prone to error. Apart from that, the running time of the proposed model is 1.5 second in average, which is not real-time speed. Real-time speed is an important element real-life application.

5.3 Future Works

There are several potential future works to address the limitation of this thesis work. As mentioned in 5.2, *Total-Text* at best a moderate-sized dataset among all the renowned scene text datasets today. As potrayed by large scale datasets in other domain like ImageNet (Deng et al. (2009)) (with more than 1 million images), scaling up the dataset would definitely attract more researchers in working towards the curved text detection problem. *T3* as presented in Section 3.7 is the first initiative towards scaling up the dataset. The framework addresses this bottleneck by reducing annotation time. While there is a potential in further reducing the annotation time with better performing detector; the image collection part of the process can be helped with the integration of automation too. For instance, a image crawling tool that is capable of recognizing the existence of curved text could greatly increase the efficiency of the image collection process. Besides, Gupta et al. (2016)'s synthetic data generation method could be another way to increase the training data. The current method can only generate axis-aliged bounding box as ground truth, it is definitely worth an exploration on the potential of generating polygon shaped ground truth as it would increase the data with the least amount of effort.

There are two potential ways to improve the performance of the proposed scene text detector. Firstly, the increment of training data would help as suggested in LeCun et al. (1998)'s study. The higher the diversity of the training data, the better the model generalize which would in turned translates to better performance. Besides, the stability of Polygon-FRCNN deemed to reduce as the vertices increased. One potential solution is to replace the polygon regression module. The latest advancement of the Faster-RCNN framework, Mask R-CNN (K. He et al. (2017)) has reported an unprecendented result in the instance segmentation field. The dense level pixel prediction module is a good replacement to the regression head module of *Polygon-FRCNN* owing to the fact that the

output mask is not constrained by text of any orientations. While segmentation-based detection algorithm is not new, the Mask R-CNN framework offers an hybrid solution. Instead of demanding the network to produce pixel-wise prediction directly from an input image, Mask R-CNN feeds cropped proposal region to the mask predicting module. Such strategy greatly reduce the complexity of the task which contributes to the success of this framework. Besides, the training of mask prediction greatly reduces the potential bias and subjectivess possessed by the ground truth.[1]

[1]The mentioned bias and subjectivess are referring to the sequence of the polygon vertices, which could varied across different shape of text instances.

REFERENCES

Andreas, V., Tomas, M., Lukas, N., Jiri, M., & Serge, B. (2016). Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*.

Bai, B., Yin, F., & Liu, C.-L. (2012). A fast stroke-based method for text detection in video. In *Document analysis systems (das), 2012 10th iapr international workshop on* (pp. 69–73).

Castrejón, L., Kundu, K., Urtasun, R., & Fidler, S. (2017). Annotating object instances with a polygon-rnn. In *Proceedings of the ieee conference on computer vision and pattern recognition* (Vol. 1, p. 2).

Chen, H., Tsai, S. S., Schroth, G., Chen, D. M., Grzeszczuk, R., & Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *Image processing (icip), 2011 18th ieee international conference on* (pp. 2609–2612).

Chen, X., & Yuille, A. L. (2004). Detecting and reading text in natural scenes. In *Computer vision and pattern recognition, 2004. cvpr 2004. proceedings of the 2004 ieee computer society conference on* (Vol. 2, pp. II–II).

Chng, C. K., & Chan, C. S. (2017). Total-text: A comprehensive dataset for scene text detection and recognition. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, *01*, 935-942.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., ... Ng, A. Y. (2011). Text detection and character recognition in scene images with unsupervised feature learning. In *Document analysis and recognition (icdar), 2011 international conference on* (pp. 440–445).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer vision and pattern recognition, 2009. cvpr 2009. ieee conference on* (pp. 248–255).

Dinh, V. C., Chun, S. S., Cha, S., Ryu, H., & Sull, S. (2007). An efficient method for text detection in video based on stroke width similarity. In *Asian conference on computer vision* (pp. 200–209).

Epshtein, B., Ofek, E., & Wexler, Y. (2010). Detecting text in natural scenes with stroke width transform. In *Proceedings of the ieee conference on computer vision and pattern recognition.*

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

Fan, Q., Brown, L., & Smith, J. (2016). A closer look at faster r-cnn for vehicle detection. In *Intelligent vehicles symposium (iv), 2016 ieee* (pp. 124–129).

Fidler, S., Mottaghi, R., Yuille, A., & Urtasun, R. (2013). Bottom-up segmentation for top-down detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3294–3301).

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, *1*(2), 119–130.

Garcia, C., & Apostolidis, X. (2000). Text detection and segmentation in complex color images. In *Acoustics, speech, and signal processing, 2000. icassp'00. proceedings. 2000 ieee international conference on* (Vol. 4, pp. 2326–2329).

Girshick, R. (2015). Fast r-cnn. In *International conference on computer vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic data for text localisation in natural images. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2315–2324).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *International conference on computer vision* (pp. 2980–2988).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

He, T., Huang, W., Qiao, Y., & Yao, J. (2016a). Accurate text localization in natural image with cascaded convolutional text network. *arXiv preprint arXiv:1603.09423*.

He, T., Huang, W., Qiao, Y., & Yao, J. (2016b). Text-attentional convolutional neural network for scene text detection. *IEEE Transactions on Image Processing*, *25*(6), 2529–2541.

He, W., Zhang, X.-Y., Yin, F., & Liu, C.-L. (2017). Deep direct regression for multi-oriented scene text detection. In *International conference on computer vision.*

Hou, R., Chen, C., & Shah, M. (2017). Tube convolutional neural network (t-cnn) for action detection in videos. In *Ieee international conference on computer vision.*

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... others (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*.

Huang, R., Shivakumara, P., & Uchida, S. (2013). Scene character detection by an edge-ray filter. In *Document analysis and recognition (icdar), 2013 12th international conference on* (pp. 462–466).

Huang, W., Qiao, Y., & Tang, X. (2014). Robust scene text detection with convolution neural network induced mser trees. In *European conference on computer vision.*

ICDAR. (2018). *Ranking table of icdar2013 localization task.* Retrieved 2018-04-18, from `http://rrc.cvc.uab.es/?ch=2&com=evaluation&task=1`

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Deep features for text spotting. In *European conference on computer vision.*

Jiang, Y., Zhu, X., Wang, X., Yang, S., Li, W., Wang, H., ... Luo, Z. (2017). R2cnn: Rotational region cnn for orientation robust scene text detection. *arXiv preprint arXiv:1706.09579*.

Karatzas, D., Gómez, L., & Rusiñol, M. (2017). The robust reading competition annotation and evaluation platform. *arXiv preprint arXiv:1710.06617*.

Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L. G., Mestre, S. R., ... de las Heras, L. P. (2013). Icdar 2013 robust reading competition. In *International conference on document analysis and recognition.*

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1725–1732).

Karsch, K., Hedau, V., Forsyth, D., & Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)*, *30*(6), 157.

Kiefer, J., & Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 462–466.

Kim, K. I., Jung, K., & Kim, J. H. (2003). Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(12), 1631–1639.

Koo, H. I., & Kim, D. H. (2013). Scene text detection via connected component clustering and nontext filtering. *IEEE Transactions on Image Processing*, *22*(6), 2296–2305.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, J.-J., Lee, P.-H., Lee, S.-W., Yuille, A., & Koch, C. (2011). Adaboost for text detection in natural scene. In *Document analysis and recognition (icdar), 2011 international conference on* (pp. 429–434).

Lee, S., Cho, M. S., Jung, K., & Kim, J. H. (2010). Scene text extraction with edge constraint and text collinearity. In *Pattern recognition (icpr), 2010 20th international conference on* (pp. 3983–3986).

Li, H., Doermann, D., & Kia, O. (2000). Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, *9*(1), 147–156.

Liao, M., Shi, B., & Bai, X. (2018). Textboxes++: A single-shot oriented scene text detector. *arXiv preprint arXiv:1801.02765*.

Liao, M., Shi, B., Bai, X., Wang, X., & Liu, W. (2017). Textboxes: A fast text detector with a single deep neural network. In *Association for the advancement of artificial intelligence* (pp. 4161–4167).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*

(pp. 21–37).

Liu, X., Fu, H., & Jia, Y. (2008). Gaussian mixture modeling and learning of neighboring characters for multilingual text extraction in images. *Pattern Recognition*, *41*(2), 484–493.

Liu, Y., & Jin, L. (2017). Deep matching prior network: Toward tighter multi-oriented text detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (Vol. 2, p. 8).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., & Young, R. (2003). Icdar 2003 robust reading competitions. In *Document analysis and recognition, 2003. proceedings. seventh international conference on* (pp. 682–687).

Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng, Y., & Xue, X. (2017). Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*.

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, *22*(10), 761–767.

Mosleh, A., Bouguila, N., & Hamza, A. B. (2012). Image text detection using a bandlet-based edge detector and stroke width transform. In *British machine vision conference* (pp. 1–12).

Nagy, R., Dicker, A., & Meyer-Wegener, K. (2011). Neocr: A configurable dataset for natural image text recognition. In *International workshop on camera-based document analysis and recognition* (pp. 150–163).

Nayef, N., Yin, F., Bizid, I., Choi, H., Feng, Y., Karatzas, D., ... others (2017). Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *Document analysis and recognition (icdar), 2017 14th iapr international conference on* (Vol. 1, pp. 1454–1459).

Neumann, L., & Matas, J. (2011). Text localization in real-world images using efficiently pruned exhaustive search. In *Document analysis and recognition (icdar), 2011 international conference on* (pp. 687–691).

Neumann, L., & Matas, J. (2013). Scene text localization and recognition with oriented stroke detection. In *Computer vision (iccv), 2013 ieee international conference on* (pp. 97–104).

Nistér, D., & Stewénius, H. (2008). Linear time maximally stable extremal regions. In *European conference on computer vision* (pp. 183–196).

Pan, Y.-F., Hou, X., & Liu, C.-L. (2011). A hybrid approach to detect and localize texts in natural scene images. *IEEE Transactions on Image Processing*, *20*(3), 800–813.

Pan, Y.-F., Liu, C.-L., & Hou, X. (2010). Fast scene text localization by learning-based filtering and verification. In *Image processing (icip), 2010 17th ieee international conference on* (pp. 2269–2272).

Phan, T. Q., Shivakumara, P., & Tan, C. L. (2012). Text detection in natural scenes using gradient vector flow-guided symmetry. In *International conference on pattern recognition.*

Polzounov, A., Ablavatski, A., Escalera, S., Lu, S., & Cai, J. (2017). Wordfence: Text detection in natural images with border awareness. *arXiv preprint arXiv:1705.05483*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Risnumawan, A., Shivakumara, P., Chan, C. S., & Tan, C. L. (2014). A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, *41*(18), 8027–8048.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Saleh, K., Hossny, M., & Nahavandi, S. (2016). Kangaroo vehicle collision detection using deep semantic segmentation convolutional neural network. In *Digital image computing: Techniques and applications (dicta), 2016 international conference on* (pp. 1–7).

Sharif R., A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 806–813).

Shi, B., Bai, X., & Belongie, S. (2017). Detecting oriented text in natural images by linking segments. In *Proceedings of the ieee conference on computer vision and pattern recognition* (Vol. 3).

Shi, B., Wang, X., Lyu, P., Yao, C., & Bai, X. (2016). Robust scene text recognition with automatic rectification. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4168–4176).

Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., & Zhang, Z. (2013). Scene text recognition using part-based tree-structured character detection. In *Computer vision and pattern recognition (cvpr), 2013 ieee conference on* (pp. 2961–2968).

Shivakumara, P., Huang, W., & Tan, C. L. (2008). Efficient video text detection using edge features. In *Pattern recognition, 2008. icpr 2008. 19th international conference on* (pp. 1–4).

Shivakumara, P., Phan, T. Q., & Tan, C. L. (2010). New fourier-statistical features in rgb space for video text detection. *IEEE Transactions on Circuits and Systems for Video Technology*, *20*(11), 1520–1532.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sutherland, I. E., & Hodgman, G. W. (1974). Reentrant polygon clipping. *Communications of the ACM*, *17*(1), 32–42.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Association for the advancement of artificial intelligence* (pp. 4278–4284).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... others (2015). Going deeper with convolutions..

Tian, S., Pan, Y., Huang, C., Lu, S., Yu, K., & Lim Tan, C. (2015). Text flow: A unified text detection system in natural scene images. In *Proceedings of the ieee international conference on computer vision* (pp. 4651–4659).

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective

search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171.

Vatti, B. R. (1992). A generic solution to polygon clipping. *Communications of the ACM*, *35*(7), 56–63.

Wang, K., Babenko, B., & Belongie, S. (2011). End-to-end scene text recognition. In *Computer vision (iccv), 2011 ieee international conference on* (pp. 1457–1464).

Wang, T., Wu, D. J., Coates, A., & Ng, A. Y. (2012). End-to-end text recognition with convolutional neural networks. In *Pattern recognition (icpr), 2012 21st international conference on* (pp. 3304–3308).

Weiler, K., & Atherton, P. (1977). Hidden surface removal using polygon area sorting. In *Acm siggraph computer graphics* (Vol. 11, pp. 214–222).

Wolf, C., & Jolion, J.-M. (2006). Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal on Document Analysis and Recognition*, *8*(4), 280–296.

Xing, D., Li, Z., Chen, X., & Fang, Y. (2017). Arbitext: Arbitrary-oriented text detection in unconstrained scene. *arXiv preprint arXiv:1711.11249*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Yao, C., Bai, X., & Liu, W. (2014). A unified framework for multioriented text detection and recognition. *IEEE Transactions on Image Processing*, *23*(11), 4737–4749.

Yao, C., Bai, X., Liu, W., Ma, Y., & Tu, Z. (2012). Detecting texts of arbitrary orientations in natural images. In *Proceedings of the ieee conference on computer vision and pattern recognition.*

Yao, C., Bai, X., Sang, N., Zhou, X., Zhou, S., & Cao, Z. (2016). Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*.

Ye, Q., & Doermann, D. (2014). Text detection and recognition in images and video : a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(7), 1-20.

Yi, C., & Tian, Y. (2011). Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, *20*(9), 2594–2605.

Yi, C., & Tian, Y. (2012). Localizing text in scene images by boundary clustering, stroke segmentation, and string fragment classification. *IEEE Transactions on Image Processing*, *21*(9), 4256–4268.

Yi, C., & Tian, Y. (2013). Text extraction from scene images by character appearance and structure modeling. *Computer Vision and Image Understanding*, *117*(2), 182–194.

Yin, X., Yin, X.-C., Hao, H.-W., & Iqbal, K. (2012). Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *Pattern recognition (icpr), 2012 21st international conference on* (pp. 725–728).

Yin, X.-C., Pei, W.-Y., Zhang, J., & Hao, H.-W. (2015). Multi-orientation scene text detection with adaptive clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1930–1937.

Yin, X.-C., Yin, X., Huang, K., & Hao, H.-W. (2013). Accurate and robust text detection: A step-in for text retrieval in natural scene images. In *Proceedings of the 36th international acm sigir conference on research and development in information retrieval* (pp. 1091–1092).

Yin, X.-C., Yin, X., Huang, K., & Hao, H.-W. (2014a). Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(5), 970-983.

Yin, X.-C., Yin, X., Huang, K., & Hao, H.-W. (2014b). Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(5), 970–983.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems* (pp. 3320–3328).

Yuliang, L., Lianwen, J., Zhang, S., & Sheng, Z. (2017). Detecting curve text in the wild: New dataset and new solution. *arXiv preprint arXiv:1712.02170*.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).

Zhang, Z., Shen, W., Yao, C., & Bai, X. (2015). Symmetry-based text line detection in natural scenes. In *Proceedings of the ieee conference on computer vision and pattern recognition.*

Zhang, Z., Zhang, C., Shen, W., Yao, C., Liu, W., & Bai, X. (2016). Multi-oriented text detection with fully convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition.*

Zhao, X., Lin, K.-H., Fu, Y., Hu, Y., Liu, Y., & Huang, T. S. (2011). Text from corners: a novel approach to detect text and caption in videos. *IEEE Transactions on Image Processing*, *20*(3), 790–799.

Zhao, Y., Lu, T., & Liao, W. (2011). A robust color-independent text detection method from complex videos. In *Document analysis and recognition (icdar), 2011 international conference on* (pp. 374–378).

Zhong, Y., Zhang, H., & Jain, A. K. (2000). Automatic caption localization in compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(4), 385–392.

Zhong, Z., Jin, L., & Huang, S. (2017). Deeptext: A new approach for text proposal generation and text detection in natural images. In *Acoustics, speech and signal processing (icassp), 2017 ieee international conference on* (pp. 1208–1212).

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: An efficient and accurate scene text detector..

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Journal

Ch'ng, C. K., & Chan, C. S. & Liu, C. L. (Under reviewed). Total-Text: Towards Orientation Diversity in Scene Text Detection. *International Journal on Document Analysis and Recognition (IJDAR)*. (ISI, IF: 1.445)

Conference

Ch'ng, C. K., & Chan, C. S. (2017). Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (pp. 935-942).*