Evaluating Object and Text Detectors under the Binary Classification Scenario: A Review

by

Yiqin Huang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

Abstract

With the explosively increasing volume of hateful speech presented with images on the Internet, it is necessary to detect hateful speech automatically. Due to the intense demand for computation from the hateful meme detection pipeline, it is vital to classify the text and non-text images for accelerating the speed of the multimodal hateful speech system. This study reviews the recent development of object and text detection architectures and categorizes them into one-stage or two-stage detectors to better compare accuracy and efficiency. Additionally, this study proposes two datasets as the benchmarks for the binary classification scenario to evaluate two representative object detectors and two state-of-art text detectors on the customized datasets with two types of texts embedded in images. The results indicate that one-stage detectors may not necessarily achieve higher throughputs than two-stage detectors, and the performance of detectors varies depending on the type of image texts. This thesis can contribute to further evaluation of detectors in binary detection tasks.

Acknowledgements

This thesis would not have been possible without the mentoring of my supervisor, Professor Mahesh Tripunitara. And I wish to acknowledge the suggestions provided by Professor Seyed Majid Zahedi and Professor Stephen L. Smith. I would also like to extend my sincere thanks to my colleagues in Scrawlr, especially Dr. Jonathan Shahen, for supporting my research.

Dedication

This is dedicated to all the people who encouraged and supported me.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AUROC Area Under the Receiver Operating Characteristic curve 2, 33, 36

CNN Convolutional Neural Networks 7, 8, 17, 18

COCO Common Objects in Context 21, 31, 32

FCN Fully Convolutional Network 14

FPN Feature Pyramid Network 17–19, 24, 26–28, 31, 34

GIF Graphic Interchange Format 1, 4

ICDAR International Conference on Document Analysis and Recognition 13, 32

IoU Intersection over Union 10, 31

NMS Non-Maximum Suppression 11, 14, 26, 28, 30, 35

R-CNNs Region-based Convolutional Neural Networks 7, 8, 11, 12, 18

ReLU Rectified Linear Unit 24, 26, 30

RoI Region of Interest 8, 9, 26, 28

RPN Region Proposal Network 9, 10, 26, 28–30

SVM Support Vector Machine 8

VQA Visual Question Answering 2, 13

Chapter 1

Introduction

This thesis is a report of a study evaluating state-of-art object-detection and text-detection frameworks on a binary text detection task. This study generated two datasets to simulate the binary classification scenarios, implemented selected frameworks, and tested their performance on the proposed datasets. This first chapter of the thesis presents the background of the study, specifies the problem of the study, describes its significance, and presents an overview of the methodology used. The chapter concludes by noting the delimitations of the study and defining some special terms used.

1.1 Background of the Study

Since Richard Dawkins coined the term "meme" to describe small units of culture that spread from person to person by copying or imitation in 1976 [15], netizens apply the tag "Internet meme" to describe the propagation of items such as jokes, rumours, videos, and websites from person to person via the Internet [16]. Internet memes can also be defined as multimodal artifacts remixed by countless participants, employing popular culture for public commentary [58]. As a daily part of our digital lives, Internet memes function as a 'media lingua franca' [58] and appear on our social media news feeds in the form of still images, images with a phrase, Graphic Interchange Format (GIF) and videos [86]. Specifically, the term 'Internet memes' in this thesis refers to the still images as Figure 1.1 shows.

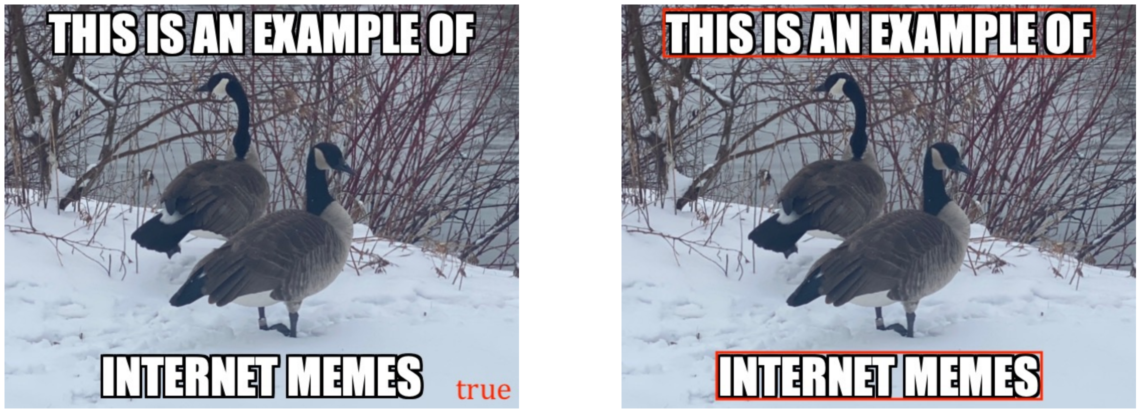
Internet memes can be 'political', 'branded', or just a bit of fun, giving us a laugh [86]. Nevertheless, Internet memes can also function as the medium of propaganda for far-right

Figure 1.1: An example of Internet memes in the form of a still image.

organizations to engage with a wider audience, 'adopting humour, irony and ambiguity' to inscribe 'popular culture iconography' with hateful messaging [3]. Thus, assessing the offensiveness level of the sentiments of user-uploaded content has great significance for maintaining a friendly online community. Due to the large volume of images and texts users upload, it is inefficient to review those content manually. Consequently, an automated solution is required to alleviate the extensiveness of online hate speech for reasons including productivity, cost, latency and scalability.

Assessing the offensiveness level of images automatically is a multimodal (vision and language) task in the sub-field of Visual Question Answering (VQA). Recent years have witnessed tremendous progress and explosion of interest in the VQA task by the machine intelligence community [78]. For instance, Facebook AI launched a competition, named Hateful Meme Challenge [38], in 2020 to detect hateful memes consisting of both images and text with early fusion multimodal frameworks, where the top five submissions achieved Area Under the Receiver Operating Characteristic curve (AUROC) scores of 0.79–0.84 on the unseen test set, significantly above the baselines [37].

In recent research, the multimodal framework is the mainstream solution to detect hateful memes, which can be summarized as the following three steps [93, 49, 74, 59, 83]: (a) Extract the texts and detect the entities in the images; (b) Combine the two modalities (vision and language); (c) Assess the sentiment of the memes comprehensively. Due to the fact that a significant part of user-uploaded images contains no text, it would waste a considerable amount of time and computation resources to assess the sentiment of the images with the whole multimodal framework. Thus, it is necessary to filter out non-text images before the framework for improving the efficiency of automatic hateful memes



(a) The classification result of Figure 1.1. (b) The localization result of Figure 1.1.

Figure 1.2: Difference in the definition of text detection.

detection.

1.2 The Problem Statement

As the primary step for numerous applications, text detection has witnessed significant progress in the recent years [53, 67, 94, 89, 12]. However, most of the recent text detection research defines text detection as a task to localize the texts in the images by outputting bounding boxes as shown in Figure 1.2b. Thus, almost all state-of-art text detection models are trained and tested on those datasets without any non-text samples, such as COCO-Text [82], ICDAR2015 [35], MLT-2017 [62], SynthText [25] and MSRA-TD500 [88]. Therefore, the performance of those models for binary classification task is not reported. Thus, to explore the impact of negative samples, this thesis aims at evaluating the performance of existing text detection algorithms in the scenario of filtering non-text images out of user-uploaded images, i.e., to generate binary outputs as Figure 1.2a shows.

1.3 Significance of the Study

Hate speech, defined as “any communication that disparages a target group of people based on some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic” by Nockleby [63], is harm-producing and amounts to a speech-act constituting harm itself [8]. However, with the development of social media, hate speech can be expressed and spread easily in the form of Internet memes, since users can save, post, and repost images. For instance, Facebook, a mainstream social networking site, took an action on 9.6 million pieces of content for violating their hate speech policies in the first quarter of 2020 [66]. At the sheer scale of the internet and the huge amount of content, it is impossible for human administrators to inspect every data point. Consequently, machine learning and artificial intelligence play a never more important role in mitigating important societal problems by detecting malicious content [37]. Considering the time and computation cost of automatic hate speech detection systems, binary text detection module, sifting Internet memes from all the user-uploaded content for further sentiment assessing, contributes to maintaining a clean online environment efficiently.

1.4 Limitations and Delimitations of the Study

Although Internet memes could combine vision and language in various forms, such as GIF and videos, this thesis only studies on still images. In addition, blurred (Figure 1.3c) or distorted images (Figure 1.3d) are not included in the dataset.

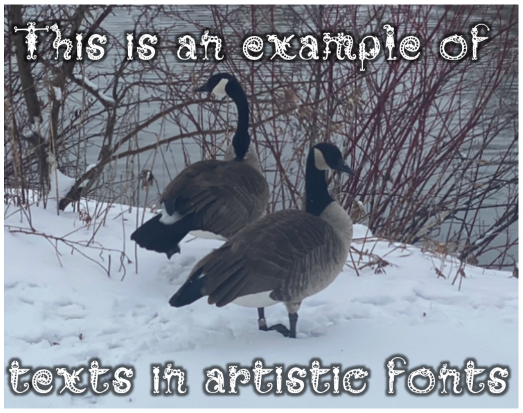
Additionally, for the texts in images, this thesis only considered texts in printed English. Handwritings (Figure 1.3b) and texts in artistic fonts (Figure 1.3a) were not studied in this study. Also, texts in languages other than English were included in this study but not regarded as a major concern.

Moreover, this thesis only reviewed open-source object detection and text detection algorithms published with a paper before 2020.

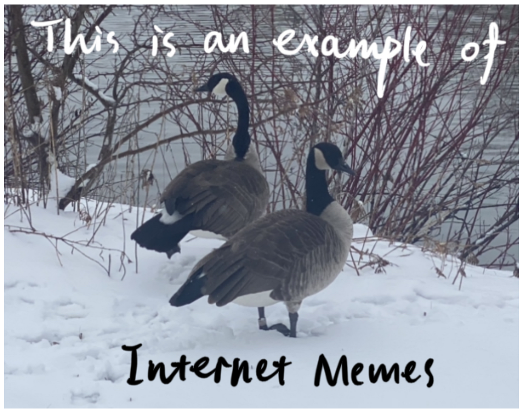
1.5 Definition of Terms

This thesis notates images without any texts as ‘non-text images’, and images with texts embedded as ‘text images’. For the purpose of filtering out non-text images, the term ‘text detection‘ is defined as a binary classification task, which takes an image as input and outputs binary results. The result is ‘true’ when the input image contains texts; the result is ‘false’ when the input image contains no text. Consequently, the term ‘negative samples’ in this thesis refers to non-text images.

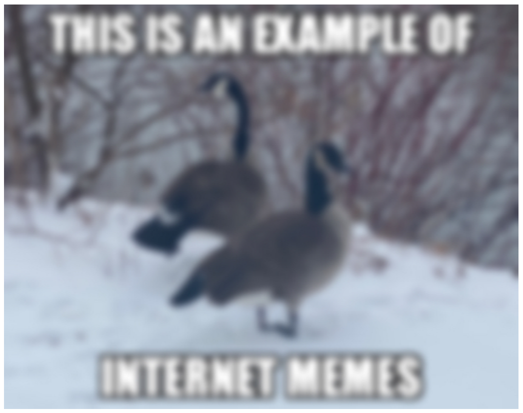
For an object-detection system, ‘bounding box’ indicates the rectangular box generated to present the location of the object, and ‘objectness’ stands for the confidence of the presence of object in the bounding box.



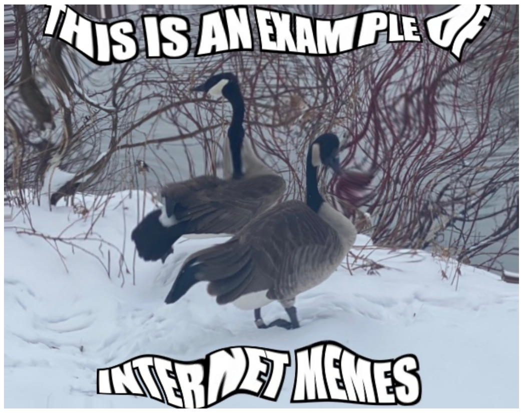
(a) An example of images embedded with texts in artistic fonts.



(b) An example of images embedded with handwriting.



(c) An example of distorted images embedded with printed texts.



(d) An example of distorted images embedded with printed texts.

Figure 1.3: Different types of text images not in the scope of this study.

Chapter 2

Literature Review

This chapter is a comprehensive review of recent major works relating to object detection and its sub-field, text detection, including their tendencies, aims, and methodologies. This chapter also reasons why prior works in text detection usually considered only positive samples and points out the potential influence of excluding negative samples.

As discussed in the previous chapter, this study is under the background of filtering out non-text samples efficiently. Thus, efficiency, including model size and training/predicting time, is the main concern when reviewing prior works. Additionally, recall is also a key parameter. Since the ultimate goal is to maximize the use of computation resources and time, this study tries to sift out as many negative samples as possible, while keeping all positive samples for more processes.

2.1 Object Detection

Object detection is a computer vision task that locates visual object instances of certain predefined categories in digital images and videos, combining image classification with precise object localization to provide a comprehensive understanding of the image [4]. As one of the most fundamental and challenging problems in computer vision, object detection has received great attention and achieved remarkable breakthroughs with the rapid development of deep learning techniques [41] in recent years. Thus, it is widely accepted that the progress of object detection has generally gone through two historical periods: 'traditional object detection period' and 'deep learning based detection period' in the past two decades[95]. Compared to traditional object detection methods relying one man-crafted feature extraction by experts [64], such as VJ detector [84], SIFT [54], HOG [14], DPM [21], deep learning techniques, especially Convolutional Neural Networks (CNN) [40], allow computational models to learn significantly complex, subtle, and abstract representations [50]. In the deep learning era, object detection can be categorized into two genres: 'two-stage detection', which frames the detection as a 'coarse-to-fine' process, and 'one-stage detection', which frames the detection as a "complete in one step" process [95].

As Figure 2.1 shows, both one-stage and two-stage object detection frameworks evolved to achieve higher speed and accuracy in the deep learning era, which inspired the innovation of text detection. Overall, the one-stage detection strategy has real-time speed but lower accuracy, especially for small objects, while the two-stage detection strategy provides more precise detection, but consumes more time. However, both two types of frameworks are adopting advantages of each other, which blurs the boundary of one-stage and two-stage detection.

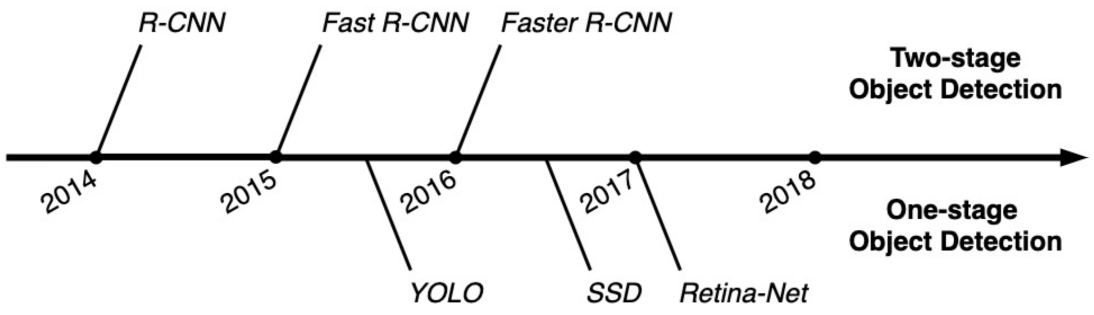


Figure 2.1: A timeline of object detection frameworks development, including the milestones of two-stage object detectors (R-CNN [24], Fast R-CNN [23], Faster R-CNN [71]) and one-stage object detectors (YOLO [68], SSD [51], Retina-Net[47]).

2.1.1 Two-stage Object Detection Frameworks

The two-stage detection methods, also called region-based methods, consist of two phases: (1) extracting region proposals, and (2) classifying regions based on computed CNN features. This section introduces the Region-based Convolutional Neural Networks (R-CNNs) series of work as a representative of two-stage detection systems.

R-CNN

R-CNN [24] is the first paper using CNN for object detection, which has a profound influence on almost all the other two-stage detection methods. R-CNN detector consists of two modules: region proposal module and feature extraction module. The region proposal module extracts a set of category-independent object proposals by selective search [80]; The feature extraction module feeds rescaled proposals into a CNN model to extract features. It then classifies objects within each region with linear Support Vector Machine (SVM) classifiers. It also regresses the bounding-boxes for precise prediction. Compared to traditional object detection methods, R-CNN gives a 30% relative improvement over the best previous results on the dataset PASCAL VOC 2012 [20, 1].

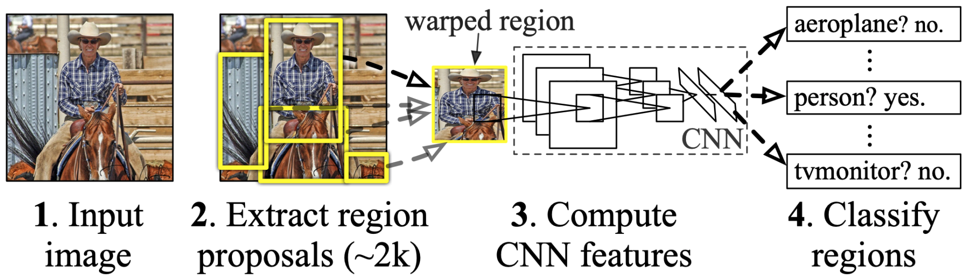


Figure 2.2: The framework of R-CNN [24].

Fast R-CNN

Since the publication of R-CNN[24], numerous models have been proposed to improve the performance of R-CNNs on object detection tasks. Fast R-CNN [23] integrates the advantages of R-CNN and SPPNet[28] and accelerates the computation by sharing computation. Instead of performing a CNN forward pass for each region proposal, Fast R-CNN produces a convolutional feature map and then extracts a fixed-length feature vector from the feature map with a Region of Interest (RoI) pooling layer for each object proposal. Additionally, Fast R-CNN improves the object detection task by jointly training the classifier and the bounding box regressor by using a multi-task loss on each labelled RoI.

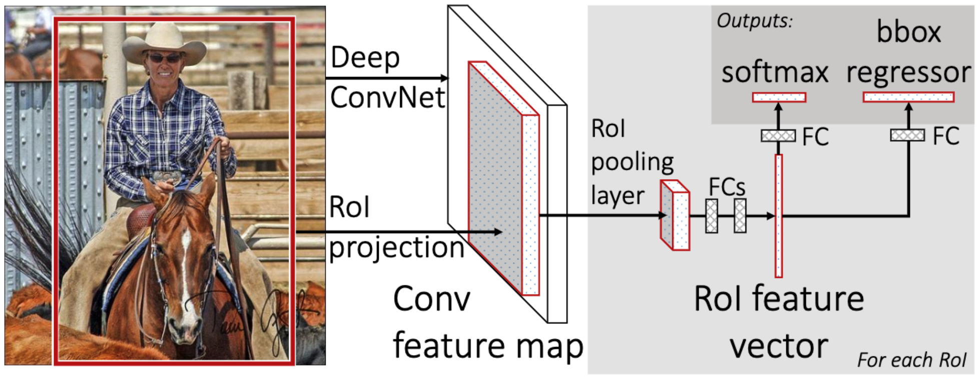


Figure 2.3: The framework of Fast R-CNN [23].

Faster R-CNN

Although Fast R-CNN [23] can train VGG16 [76], a very deep detection network, 9 times faster than R-CNN [24] and 3 times faster than SPPnet [28], its detection speed is still limited due to the proposal detection. To breakthrough this bottleneck, Faster R-CNN[71] was proposed shortly after the Fast R-CNN, introducing RPN to achieve nearly cost-free region proposals and propose a training scheme to combine the Fast R-CNN and the RPN into a unified system.

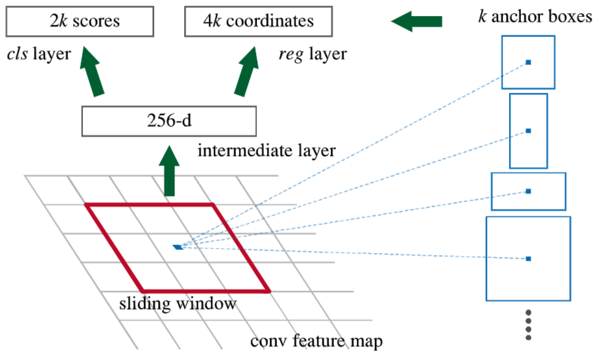


Figure 2.4: The structure or RPN [50].

In contrast to using Selective Search [80] to propose RoI, RPN introduces novel "anchor" boxes. As Figure 2.4 shows, anchor boxes are a series of multiple scales and aspect ratios bounding boxes, generated with a sliding fixed-size window on the shared feature maps.

For each anchor boxes, the training scheme calculates binary classification labels according to a certain threshold of Intersection over Union (IoU) and the ground truths. Taking those binary classification labels and the vertices of anchor boxes as the inputs, RPN combines the loss of binary classification and region regression for training. Thus, the Faster R-CNN waives nearly all computational burdens of Selective Search at test-time and shares convolutional features with the downstream detection network to simultaneously regress region bounds and object categories scores at each location on a regular grid, which leads to a significant improvement of efficiency and overall object detection accuracy.

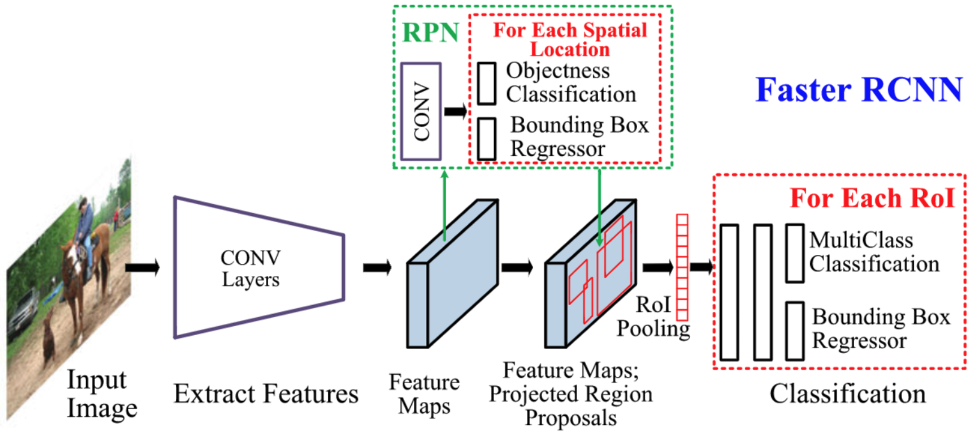


Figure 2.5: The framework of Faster R-CNN [50].

2.1.2 One-stage Object Detection Frameworks

In contrast to two-stage detection systems, which apply image classifiers on hypothesized bounding boxes to perform detection, one-stage detection systems adopt global regression or classification strategies without region proposal networks. Therefore, one-stage detectors are also called region-free detectors.

YOLO (You Only Look Once)

YOLO [68] was proposed as the first one-stage object detector in 2015, which converts the object detection task into a regression problem with only a single neural network to perform unified detection. Inspired by the GoogLeNet [79], an image classification model, the single neural network in YOLO has 24 convolutional layers followed by 2 fully connected layers, which takes entire images as inputs and requires only one network evaluation. The prediction pipeline pre-processes input images by dividing them into cells with a grid, predicts bounding boxes for each grid cell and confidence scores for each bounding box, and post-process the outputs with Non-Maximum Suppression (NMS) to eliminate multiple detections.

Due to the novel framework, YOLO can be optimized end-to-end directly on detection performance with real-time speeds and high average precision. Compared to region-based methods, such as R-CNNs, YOLO has better performance on encoding contextual information and distinguishing objects from backgrounds. However, the low-resolution grid limits the performance of YOLO, especially under several scenarios, including small objects, adjacent objects close to each other.

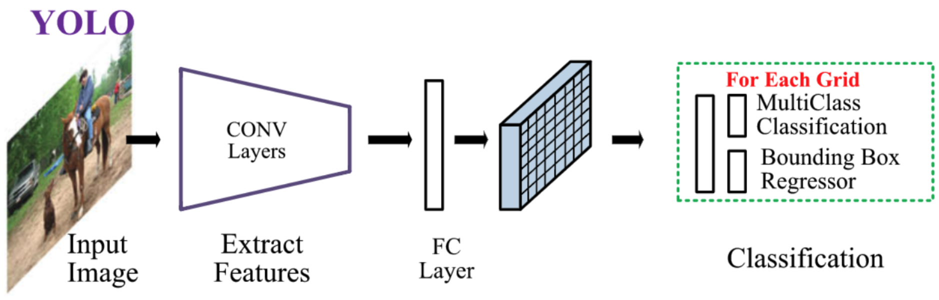


Figure 2.6: The framework of YOLO [50].

YOLO evolves. YOLOv2 [69] proposes Darknet-19, a backbone network for classification, and adopts a series of design decisions to achieve higher detecting precision, including batch normalization, anchor boxes, dimension clusters, fine-grained features and multi-scale training. Inspired by ResNet [29], YOLOv3 [70] proposed Darknet-53, a deeper and robust feature extractor, and adopts independent logistic classifiers, performing better on complex datasets with overlapping objects. Furthermore, YOLOv4 [10] achieves even higher detection accuracy by adopting a series of "bag of freebies" and "bag of specials". The former concept indicates the methods changing the training strategy or increasing the training cost only; the later concept represents the plugin modules and post-processing methods with a small amount of inference cost yet significant improvement in the accuracy.

SSD (Single Shot MultiBox Detector)

SSD [51] is the first object detector without pixels or features resampling for bounding box hypotheses. In contrast to YOLO [68], SSD adopts the VGG16 backbone network [76] and a set of default anchor boxes to detect objects of various scale and aspect ratios on multiple feature map layers to improve the detection accuracy on small objects. Additionally, SSD predicts object categories and offsets in bounding box locations and their corresponding confidence scores with a small convolutional filter, instead of detecting the presence of object categories in the default grid as YOLO.

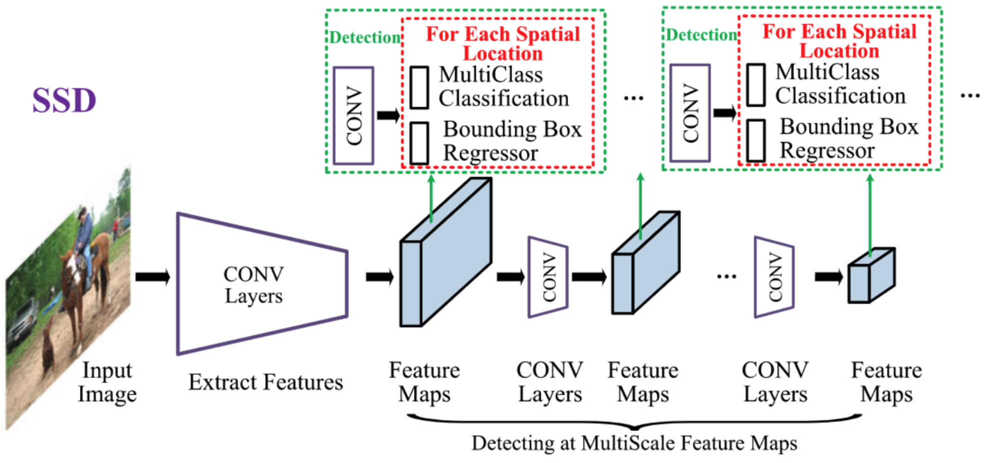


Figure 2.7: The framework of SSD [50].

With the framework predicting on multiple feature maps, SSD increases detection speed further while achieving high accuracy with relatively low-resolution input. Compared to the state-of-art two-stage object detector in 2016, Faster R-CNN [71], SSD is three times faster and achieves higher accuracy on the object detection datasets PASCAL VOC [20] and COCO [48]. Thus, most of the subsequent one-stage object detection systems were developed based on SSD. Furthermore, DSSD (Deconvolutional Single Shot MultiBox Detector) [22] is proposed based on SSD by adding the deconvolutional module and the prediction module to strengthen features by increasing the resolution of the feature maps.

RetinaNet

RetinaNet [47] is a one-stage object detector using ResNet-FPN [29, 46] as the backbone network. The main contribution of RetinaNet is introducing the Focal Loss for classification to solve the problem of foreground-background class imbalance. Unlike two-stage detectors, such as R-CNNs, filtering out most background locations in the first stage, i.e., region proposal stage, most of the one-stage detectors generate a dense set of bounding boxes for backgrounds, which leads to extreme imbalance between positive locations and negative locations. Focal loss helps the system to focus on the hard training examples and avoid the vast number of easy negative examples overwhelming the detector during training by down-weighting the loss assigned to well-classified or easy examples. Combining the advantage of ResNet, FPN and Focal Loss, RetinaNet outperforms DSSD [22] in terms of detection precision, especially on small and medium objects.

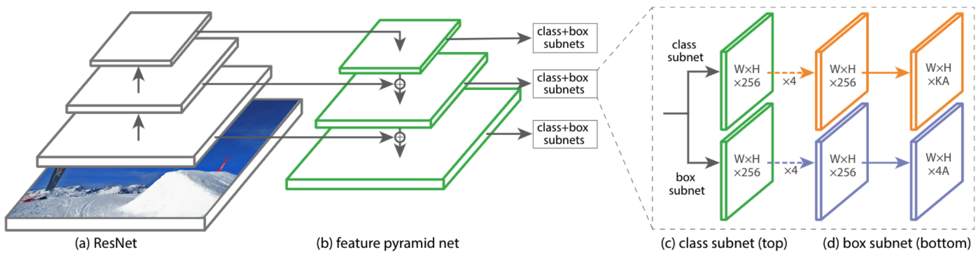


Figure 2.8: The architecture of RetinaNet [47].

2.2 Text Detection

Text detection, as a subfield of object detection, also exploits the witnessed advancement in deep learning and object detection to adapt to complicated scenarios. In the classical machine learning era, the majority of text detection methods relied on Sliding Window [39, 11, 26] and Connected Components Analysis [56, 19]. Compared to the classical machine learning based methods, deep learning brings significant improvements in efficiency, robustness and precision to text detection frameworks. Thus, this section focuses on only deep learning based text detection systems.

Different from object detection, the ultimate purpose of text detection is not just to locate the texts in images precisely, but to serve text recognition and even further applications, such as VQA [9, 57]. Therefore, recent researches on text detection, including algorithms and datasets, are oriented by challenges in text recognition, especially the Robust Reading competition held by International Conference on Document Analysis and Recognition (ICDAR). Furthermore, the trend of text detection and recognition is not limited to pursuing extreme performance but solving more complicated and comprehensive problems, such as multi-language texts[61, 60], distorted images [32, 34], and texts aligned in arbitrary shapes and orients [13]. Thus, this section aims to review text detection methods according to their performance under the scenario of classifying text and non-text images, instead of their robustness under complicated scenarios.

Influenced by the categorization of text recognition, most text detection surveys classify text detection frameworks according to the methods used in the model. Raisi et al. categorized text detection methods into bounding-box regression-based, segmentation-based, and hybrid approaches [67]. Long et al. classified text detection methods into pixel-level, component-level, and character-level methods [53]. Liu et al. used region proposal-based, segmentation-based, and hybrid methods to distinguish text detection methods [50]. Lin et al. grouped text detectors as semantic segmentation based methods, general object detection based methods, and hybrid methods [45]. Considering the study is under the scenario of distinguishing text/non-text images efficiently, this section categorizes text detection frameworks into one-stage and two-stage text detection.

2.2.1 Two-stage Text Detection Frameworks

The criteria to distinguish two-stage text detection algorithms are different from object detection algorithms. The pipeline of a text detection algorithm may consist of more than two stages, such as feature extraction, text-line generation, word partition, proposal filtering, etc. Unlike two-stage object detectors, the first stage of a two-stage text detector is not necessarily to be the region proposal. Thus, this study classifies a one-stage text detector from a two-stage one by whether the detector has a text instance segmentation phase.

EAST (Efficient and Accurate Scene Text detector)

As Figure 2.9 shows, EAST [92] consists of two stages: (a) Fully Convolutional Network (FCN), the feature extraction stage, which produces text regions predictions directly without word partition; (b) NMS, the candidate aggregation stage, which is used as the post-processing applying thresholding on the scores of text regions to yield word or line-level detection results.

EAST provides two geometry shapes for text regions, rotated box and quadrangle, and corresponding loss functions to detect multi-oriented and multi-quadrilateral shapes text.

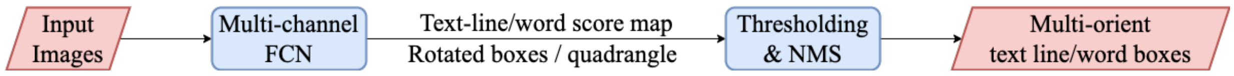


Figure 2.9: The pipeline of EAST.

SAST(Single-Shot Arbitrarily-Shaped Text detector)

Though names as a single shot text detector, SAST [85] consists of two stages, the feature extraction stage and the text instance segmentation stage, as Figure 2.10 shows. The feature extraction stage uses context attention blocks

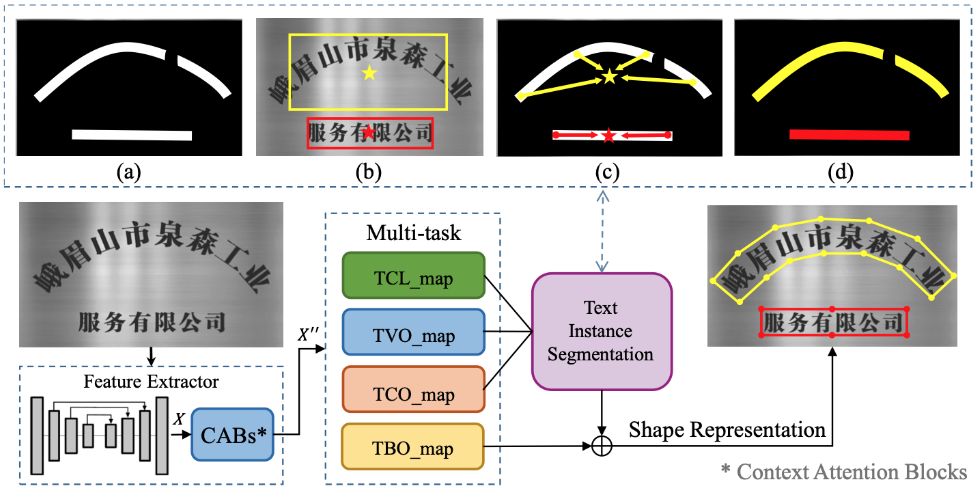


Figure 2.10: The pipeline of SAST [85].

2.2.2 One-stage Text Detection Frameworks

MSP-Net (Multi-scale Spatial Partition Network)

MSP-Net [5] is proposed to distinguish images that contain text from a large volume of natural images, which is highly similar to this study. As Figure 2.11 shows, MSP-Net divides images into multiple-scale squares via spatial partitions and distinguishess text images from non-text images by simultaneously predicting the presence of texts in each squares of images in a single forward propagation.

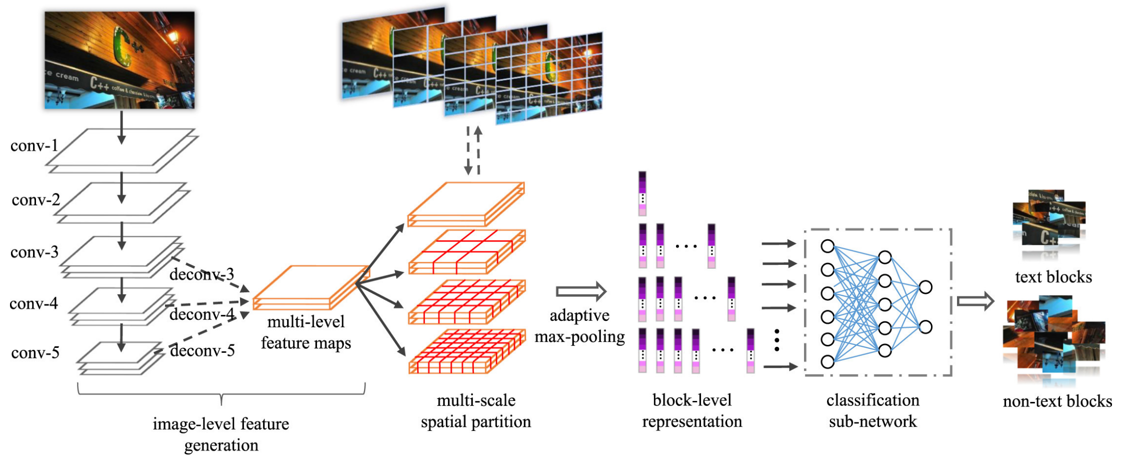


Figure 2.11: The architecture of MSP-Net [5].

Due to the extreme lightweight architecture, this framework has high throughput but limited performance, especially with those texts of extreme aspect ratios, which is hard to capture via spatial partitions.

TextBoxes

Inspired by SSD [51], TextBoxes [43] is an end-to-end trainable one-stage text detector, which proposes the text-box layers to solve this problem that SSD fails on text detection due to the extreme aspect ratios of texts. The architecture is as Figure 2.12 shows.

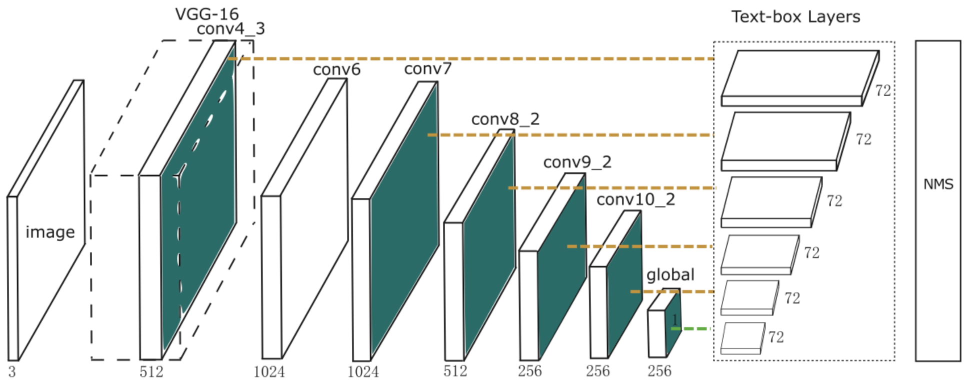


Figure 2.12: The architecture of TextBoxes [43].

DB (Real-time scene text detection with Differentiable Binarization)

The main contribution of DB-Net [44] is introducing a novel differentiable binarization module (DB module) to perform an end-to-end trainable binarization process in a segmentation network. The traditional segmentation-based network consists of two stages, pixel-level prediction and post-processing algorithms, to get the bounding boxes. DB uses two heads to produce probability map and threshold map from a FPN backbone, and calculates the binary map from the former two maps through the DB module to directly outputs predictions. Based on a simple segmentation network, DB achieves state-of-the-art results in terms of both detection accuracy and speed on the different benchmarks.

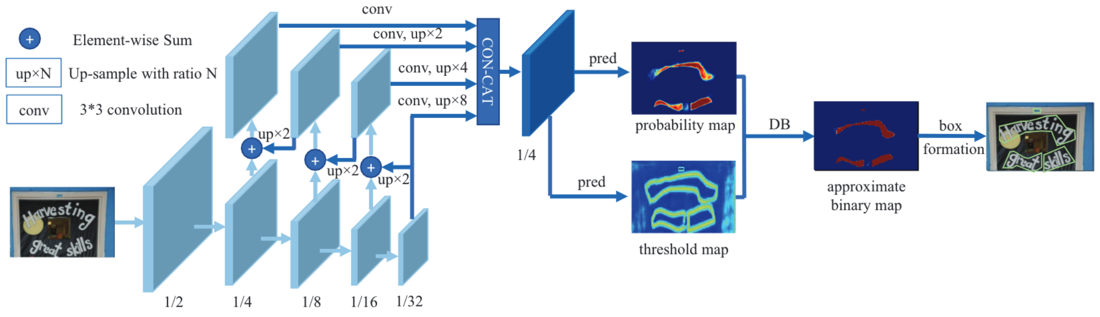


Figure 2.13: The architecture of DB-Net [44].

2.3 Other Technical Innovations

Deep-learning-based architectures, especially object detection architectures, can be divided into "head", "neck" and "backbone" networks, where the backbone network is a term of feature extractors, the head network indicates the method predicting and outputting according to extracted features, and the neck networks stands for the sub-network between the head and the backbone for better feature extraction. In this section, ResNet, a backbone network, and FPN, a neck network, are introduced used, which are used to build object detectors and text detectors in this study.

2.3.1 ResNet (Residual Network)

With the development of CNN, deeper neural networks are proposed to better extract the feature and further represent certain function classes [77] than shallower neural networks.

However, it is difficult to train neural networks with increasing depth due to the degradation problem, which is the phenomenon triggered by higher training error that the accuracy gets saturated and then degrades rapidly with the network depth increasing. ResNet [29] introduces the residual learning module to address the problem, which creates a shortcut connection between the input and output to a CNN block to imply an identity mapping, so that the following layers can learn the feature from the input easier from the residual.

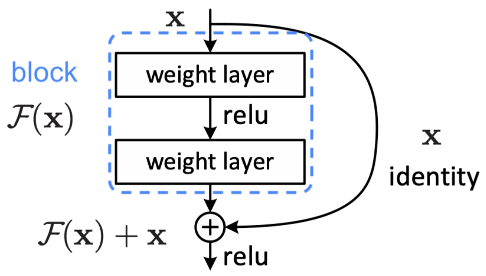


Figure 2.14: A block using residual learning [29].

As Figure 2.14 shows, notate x as the input of the block, and $\mathcal{F}(x)$ as mapping of this block. By the shortcut connection, The original mapping is recast into the residual mapping, $\mathcal{F}(x) + x$, for easier optimization, which can be realized by feed-forward neural networks. With residual learning, ResNet into very deep neural networks by stacking blocks. Compared to VGG [76] which can be up to 19 weight layers, ResNet can be up to 152 weight layers. Thus, ResNet is used in various frameworks, such as R-CNNs, for better feature extraction. In this study, I used ResNet50, the 50-layer residual network, as the backbone network of object detectors.

2.3.2 FPN (Feature Pyramid Network)

Pyramids structures, a series of images or feature maps in a hierarchy of high to low resolution, are considered as effective methods to representing image information since the traditional image processing era [2]. However, it is compute and memory intensive to build feature pyramids upon image pyramids, especially with increasingly deeper neural networks in the deep learning era. To alleviate this problem, FPN, a top-down feature pyramid architecture with lateral connections, is proposed to extract high-level semantic features and detect objects at all scales with marginal extra cost [46].

2.3.3 ResNet + FPN

ResNet is a good candidate as the bottom-up pathway of FPN. According to the experiments of running ResNets with different numbers of layers on the CIFAR-10 test set[29], the classification error decreases as the number of layers increases until it reaches 112 layers. Considering the affordable computation resources and training time, this study uses the ResNet50, the 50-layer ResNet, to build the backbone. 7

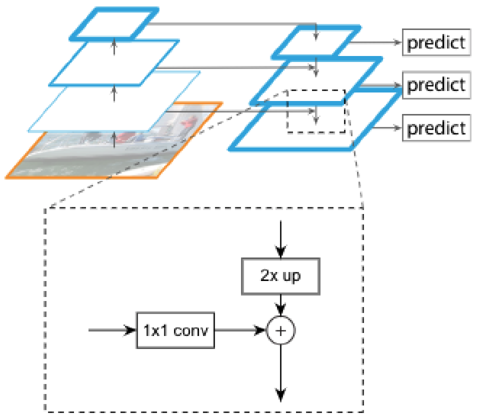


Figure 2.15: A block merging the lateral connection and the top-down pathway by addition[46].

As Figure 2.15 shows, FPN consists of a bottom-up pathway, a top-down pathway and lateral connections. The bottom-up pathway conduct feed-forward computation with ResNet [29] to downsample and create "stages" of decreasing spatial resolution. And the top-down pathway reuses the last stage with the nearest neighbour upsampling spatially coarser to reconstruct the stages of higher resolution. Additionally, lateral connections merge two feature maps of each stage to combine the higher-level semantics from the top-down pathway and more precise locations of features from the bottom-up pathway. Furthermore, FPN leverages the architecture as a feature pyramid and enables prediction independently on each stage.

Chapter 3

Methodology

This chapter explains the methods used in this quantitative study, which evaluates the performance of object and text detectors on classifying text and non-text images. Two datasets were generated and used as benchmarks for the binary classification task, whose data were collected from two text detection datasets. The experimental data for performance comparison were gathered by controlling the structure of the backbone network and training parameters.

3.1 Dataset

Scene text and born-digital text are considered as two basic classes of texts in images, where the former refers to text naturally present on a scene image, such as advertising boards and signage boards; the latter refers to text that is digitally placed in a digitally created image [89]. In this study, object detectors and text detectors were evaluated with both two types of texts.

As discussed in Section 1.2, most text detection datasets include only positive samples. This section introduces the source of negative samples and the method of creating new datasets for binary text detection. To prevent the possibility that the model is learning the difference between two datasets instead of the difference between text and non-text, the negative samples and positive samples for experiments were selected from the same dataset.

Since the text detection and object detection frameworks require bounding boxes as their ground truth format, the ground truth in the training process is presented by minimum bounding box — for each text instance in a sample image, its minimum bounding box is a rectangle of the smallest area but covering every pixel of the text as Figure 3.1b and 3.4a shows.

3.1.1 Scene Text Dataset

The dataset COCO is proposed for advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding, which contains 2.5 million labelled instances of easily recognizable 91 objects types in 328k images [48]. Based COCO, COCO-Text[1] annotates the text instances in a part of samples in the dataset `train2014`[2] (the 2014 version training dataset of COCO).

Note that COCO-Text annotates only partial positive samples in `train2014`, it is incorrect to flag all the other samples in `train2014` excluded from COCO-Text as negative samples. In other words, notating the whole `train2014` as the universe, the set of all positive samples is the absolute complement of the set of all negative samples, while the samples annotated by COCO-Text is just a subset of all positive samples, which indicates the absolute complement of COCO-Text contains both positive and negative samples. Thus, we cannot get a reliable source of negative samples directly by conducting set subtraction between `train2014` and COCO-Text. Therefore, we created a new dataset for binary text detection, named as `coco1000`[3], by selecting 500 non-text images from `train2014` manually and combining those negative samples with 500 positive samples from COCO-Text.

3.1.2 Born-digital Text Dataset

SynthText [25] is a project that provides source code[4] and a dataset of background images, `bg_img`[5], to generate born-digital text datasets synthetically. Though the `bg_img` is

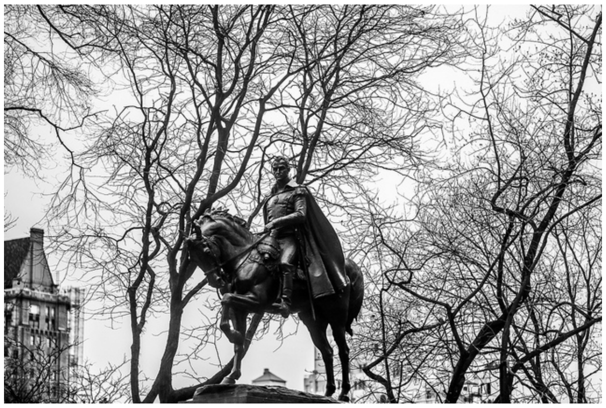
[1]Download: https://github.com/bgshih/cocotext/releases/download/dl/cocotext.v2.zip

[2]Download: http://images.cocodataset.org/zips/train2014.zip

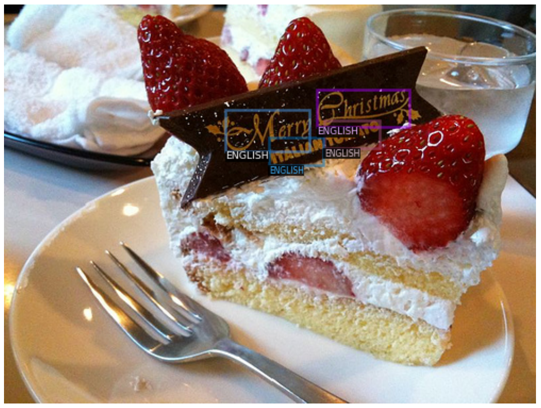
[3]Download: https://drive.google.com/file/d/163U8722lrJwKF1GermdLOXG7wrhHso2n/view?usp=sharing

[4]Source code: https://github.com/ankush-me/SynthText

[5]Download: https://thor.robots.ox.ac.uk/~vgg/data/scenetext/preproc/bg_img.tar.gz



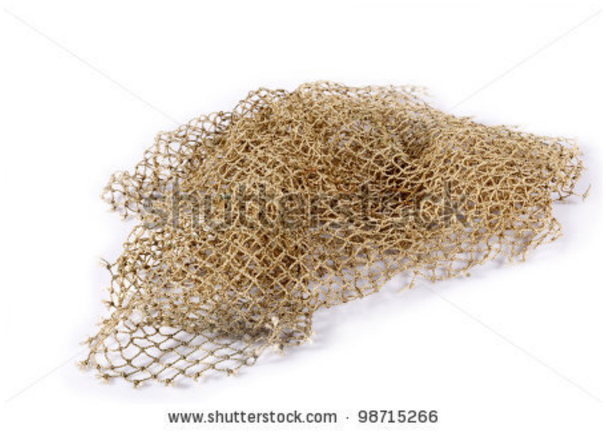
(a) A negative sample.



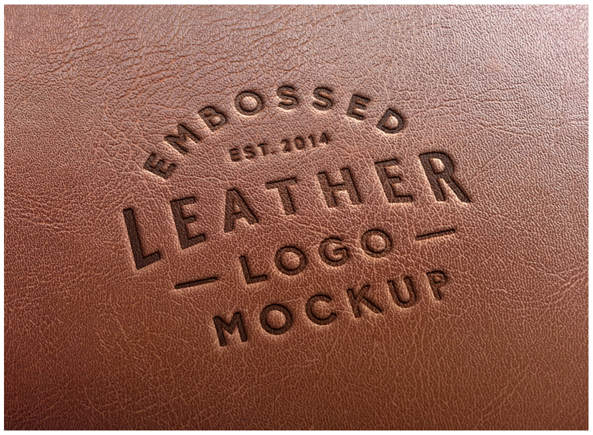
(b) A positive sample with its annotations.

Figure 3.1: Samples from `coco1000`.

supposed to be a reliable source of non-text images, it contains some images with born-digital texts (Figure 3.2a) and scene texts (Figure 3.2b).



(a) `net_19.jpg` contains born-digital text.



(b) `leather_44.jpg` contains scene text.

Figure 3.2: Examples of text images from `bg_img`.

Based on the fact that `bg_img` contains positive samples, we created a dataset for born-digital text detection, named as `synthtext498`[6], combining manually selected non-text background images from `bg_img` and text images from the pre-generated SynthText

[6]Download: https://drive.google.com/file/d/1tFQpn4KC8eSDDT8qLCcWUQivLut4M_sT/view?usp=sharing

dataset[7]. Observe that the bounding boxes in the ground truth of the pre-generated SynthText dataset were presented via convex quadrilaterals instead of rectangles. For unifying the shape of bounding boxes, the vertices of a bounding box in the `synthtext498` were generated by calculating the minimum and the maximum of the abscissa and ordinate as Figure 3.3 shows.

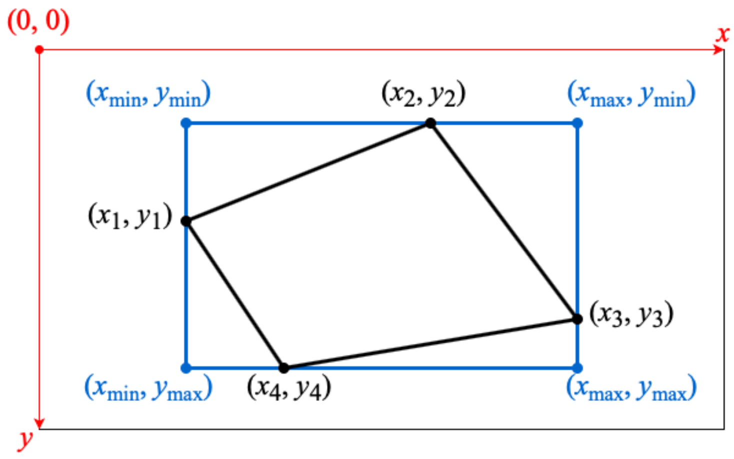
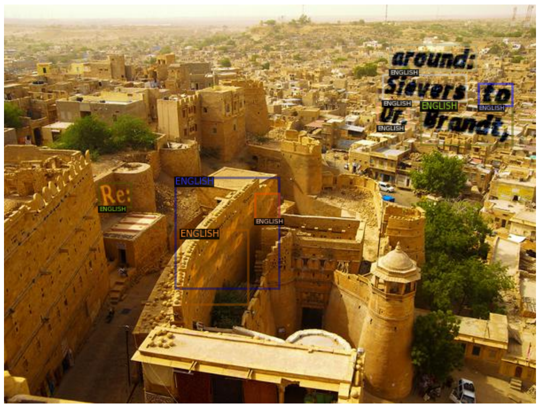


Figure 3.3: Converting a convex quadrilateral bounding box into a rectangle bounding box.

(a) A negative sample.



(b) A positive sample with its annotations.

Figure 3.4: Samples from `synthtext498`.

[7]Download: `https://thor.robots.ox.ac.uk/~vgg/data/scenetext/SynthText.zip`

3.2 Frameworks

As discussed in the previous chapter, this study chose Faster R-CNN [71] and RetinaNet [47] as the milestone of two-stage and one-stage object detector to be implemented and evaluated under the binary text detection scenario. As for text detectors, DB-Net [44] and SAST [85] were selected as representative one-stage and two-stage frameworks, and their pre-trained models were evaluated in this study. Additionally, both two selected object detection frameworks were established with the same backbone network to extract the features of input images. And the selected pre-trained models of DB-Net and SAST also adopted the backbone network with the identical structure of the network used in object detectors.

3.2.1 Backbone Network

As discussed in Section 2.3.3, the backbone network was built according to the FPN structure [46], where the ResNet50 [29] functioned as the bottom-up pathway.

The Structure of ResNet + FPN The structure of the backbone network is shown as Figure 3.5. Based on the spatial pyramid hierarchy of FPN, the ResNet could be divided into five stages. Between each stage, a shortcut connection layer matched the dimension for residual learning. The first level of the ResNet50 was similar to the stem block in GoogLeNet [79]. The rest of the stages were labelled as Res2 to Res4 sequentially. Each stage halved the size of the output feature maps while maintaining the time complexity by stacking an increasing number of 3-layer convolution blocks.

The Implementation of ResNet50 + FPN In this study, the 50-layer of ResNet was implemented. Notating the width and height of an input image as H and W, the stem compressed the input image to $^1/_4H \times {}^1/_4W$ with a 7×7 convolution, followed by Rectified Linear Unit (ReLU) and max pooling layer (kernel size $= 3 \times 3$, stride $= 2 \times 2$, padding $= 1 \times 1$). The rest of the stages were labelled as Res2 to Res4 sequentially. Each stage halved the size of the output feature maps while maintaining the time complexity by stacking an increasing number of 3-layer convolution blocks. In every convolution block, the first 1×1 convolution reduced the dimensions, so the 3×3 convolution reduced dimensions to extract features with smaller dimensions for less computation, and the second 1×1 convolution restored the dimensions. After each convolutional layer of ResNet50 listed in Table 3.1, batch normalization was adopted to accelerate the convergence, followed by

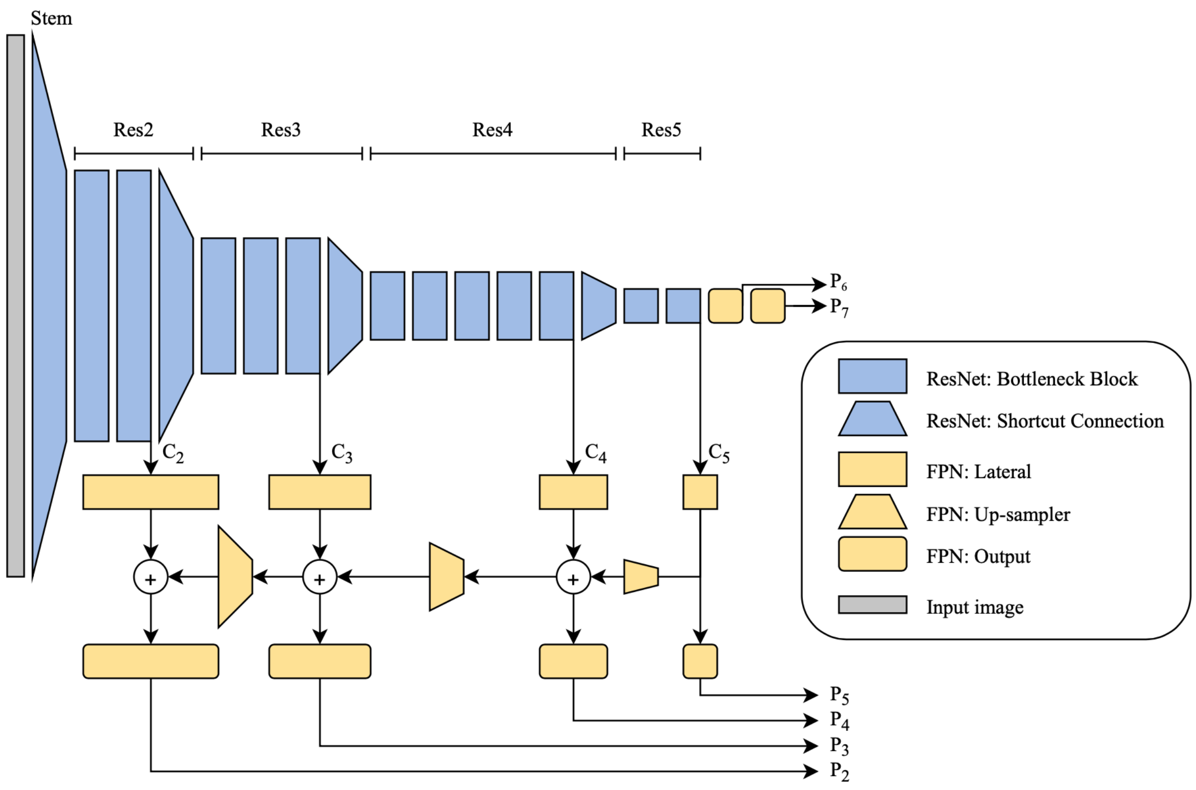


Figure 3.5: The structure of the backbone network combining ResNet50 and FPN.

ReLU activation to alleviate the vanishing gradient problem further. Notate the level of the stage as l, then the size of the feature map at level l is ${}^{1}/_{2^l}H \times {}^{1}/_{2^l}W$.

Due to the large memory footprint yet limited semantic features of the first stage (stem), the FPN structure utilized the feature activations of the last residual block from Res2 to Res5, notated as C_2 to C_5 sequentially for prediction. The lateral connections, 1×1 convolutional layers, unified the channel dimensions to 256 from C_2 to C_5. As Figure 3.5 shows, at the top of the backbone network, the output of lateral connection on $C5$, notated as C_5', was used to generate a coarser feature map of the same size with C_4 via nearest neighbour up-sampling by a factor of 2. The up-sampled C_5', merged with the lateral connection output of C_4 via element-wise addition, continued to undergo up-sampling along the top-down pathway to reconstruct finer resolution maps. This process would keep iterating until it reached the C_2 level. Then the merged maps at the second, third and fourth level output the final feature maps, notated as P_2 to P_6 correspondingly, through 3×3 convolutional layers to reduce the aliasing effect of up-sampling. Additionally, P_5 was generated through a 3×3 convolution directly from C_5', while P_6 and P_7 were generated via convolutions from C_5. The parameters of the covolutional layers in FPN were listed in Table 3.2.

3.2.2 Object Detection Frameworks

The main divergence between two-stage and one-stage detection frameworks is of the detection head structure, which localizes and classifies the objects based on the feature maps extracted by the backbone network. This subsection illustrates the structure of two object detectors, emphasizing how these detection heads regress the classify.

The Two-stage Object Detection Framework: Faster R-CNN

The Structure of the Faster R-CNN Head The head of the selected two-stage object detector, Faster RCNN, consisted of two parts, the RPN and the RoI head. In RPN, the anchor generator first placed anchor boxes of r different aspect ratios and a specific size on the feature map at each level. Then 4 layers of feature maps, P_2 to P_6, were sent to RPN one by one. Next, the input feature map underwent a convolution, and its outputs were used to output objectness logits and anchor deltas through another two convolutional layers. The objectness logits were represented by r channels, indicating the logits of the anchors of different aspect ratios on each grid cell. And the anchor deltas used $4r$ channels to represent the 4 vertices of each r anchors. At each level, RPN adopted NMS algorithm

Names	Layers	#Input	#Output	Kernel	Stride	Padding
Stem	2-D Convolutional	3	64	7×7	2×2	3×3
Res2	Shortcut Connection	64	256	1×1	1×1	
	2-D Convolution	64	64	1×1	1×1	
	2-D Convolution	64	64	3×3	1×1	1×1
	2-D Convolution	64	256	1×1	1×1	
	Bottleneck Block $\times 2$	256	64	1×1	1×1	
		64	64	3×3	1×1	1×1
		64	256	1×1	1×1	
Res3	Shortcut Connection	256	512	1×1	2×2	
	2-D Convolution	256	128	1×1	2×2	
	2-D Convolution	128	128	3×3	1×1	1×1
	2-D Convolution	128	512	1×1	1×1	
	Bottleneck Block $\times 3$	512	128	1×1	1×1	
		128	128	3×3	1×1	1×1
		128	512	1×1	1×1	
Res4	Shortcut Connection	512	1024	1×1	2×2	
	2-D Convolution	512	256	1×1	2×2	
	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	1024	1×1	1×1	
	Bottleneck Block $\times 5$	1024	256	1×1	1×1	
		256	256	3×3	1×1	1×1
		256	1024	1×1	1×1	
Res5	Shortcut Connection	1024	2048	1×1	2×2	
	2-D Convolution	1024	512	1×1	2×2	
	2-D Convolution	512	512	3×3	1×1	1×1
	2-D Convolution	512	2048	1×1	1×1	
	Bottleneck Block $\times 2$	2048	512	1×1	1×1	
		512	512	3×3	1×1	1×1
		512	2048	1×1	1×1	

Table 3.1: The parameters of the bottom-up pathway, ResNet50, in FPN. #Input and #Output stands for the channel dimension of the inputs and outputs. Convolutional blocks (as Figure 2.15 shows) are represented in merged rows with the numbers of blocks stacked.

Names	Layers	#Input	#Output	Kernel	Stride	Padding
Lateral C_2	2-D Convolution	256	256	1×1	1×1	
Output P_2	2-D Convolution	256	256	3×3	1×1	1×1
Lateral C_3	2-D Convolution	512	256	1×1	1×1	
Output P_3	2-D Convolution	256	256	3×3	1×1	1×1
Lateral C_4	2-D Convolution	1024	256	1×1	1×1	
Output P_4	2-D Convolution	256	256	3×3	1×1	1×1
Lateral C_5	2-D Convolution	2048	256	1×1	1×1	
Output P_5	2-D Convolution	256	256	3×3	1×1	1×1
Output P_6	2-D Convolution	2048	256	3×3	2×2	1×1
Output P_7	2-D Convolution	256	256	3×3	2×2	1×1

Table 3.2: The parameters of the top-down pathway and lateral connections in FPN. #Input and #Output stands for the channel dimension of the inputs and outputs.

to rank the top-1000 regions and passed 1000 proposal boxes and objectness logits to RoI Head. The RoI Head conducted RoI pooling with 4 layers of RoI Align to extract a small feature map from each RoI [27], flatten sequentially and generate classification scores and bounding box predictions through two fully connected layers. Finally, the Faster RCNN Head output top-100 proposals through NMS and sift out proposals with objectness lower than a certain threshold.

Implementation of the Faster R-CNN Head In this study, the anchor generator was implemented to place anchor boxes of 3 different aspect ratios $\{0.5, 1, 2\}$ and a specific size on the feature map at each level ($\{32, 64, 128, 512\}$ corresponding to feature maps $\{P_2, P_3, P_4, P_5\}$). Thus, the objectness logits were represented by 3 channels, indicating the logits of 3 anchors of different aspect ratios on each grid cell in RPN,. And the anchor deltas used 3×4 channels to represent the 4 vertices of each 3 anchors. The parameters of the convolutions in RPN is listed in Table 3.3.

The One-stage Object Detection Framework: RetinaNet

Structure of the RetinaNet Head Different with the Faster RCNN Head, The selected one-stage object detector, RetinaNet, took the outputs $\{P_3, P_4, P_5, P_6, P_7\}$ from the backbone instead of $\{P_2, P_3, P_4, P_5, P_6\}$. P_2 was discarded for its intensive computation demand, while P_7 was added for better detection of big objects. In RetinaNet Head, the

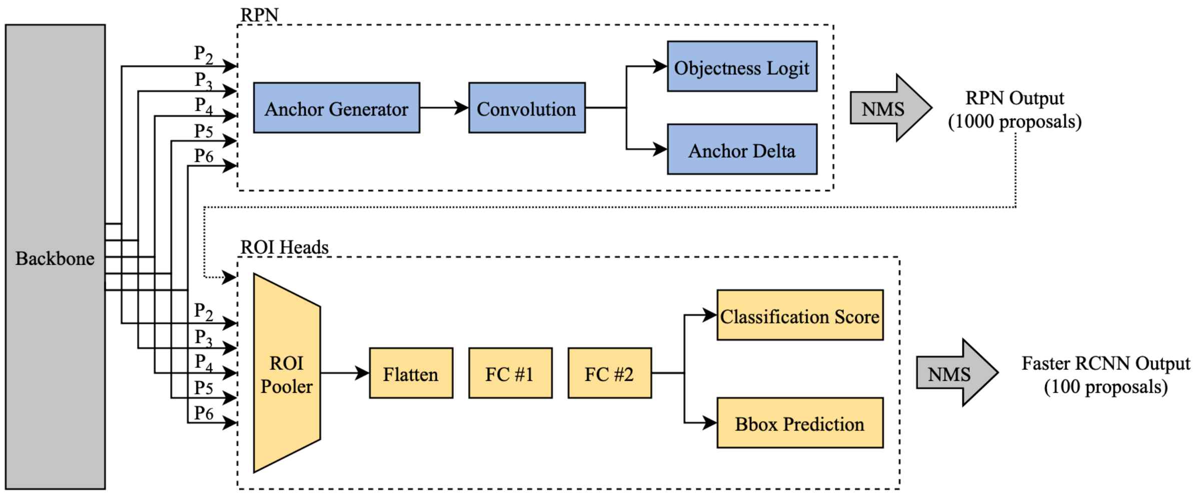


Figure 3.6: The structure of Faster RCNN Head.

Names	Layers	#Input	#Output	Kernel	Stride	Padding
Convolution	2-D Convolution	256	256	3×3	1×1	1×1
Objectness Logit	2-D Convolution	256	3	1×1	1×1	
Anchor Delta	2-D Convolution	256	12	1×1	1×1	

Table 3.3: The parameters of the convolutions in the RPN head.

feature map from each pyramid level was processed in two sub-networks, the `class` subnet for object classification and the `box` subnet for bounding box regression.

In the `class` and box subnet, the anchor generator generated multiple anchors of different aspect ratios and sizes at each spatial position. In contrast to the RPN used in Faster R-CNN, the classification subnet in RetinaNet was deeper and did not share parameters with the box regression subnet. The `class` subnet and `box` subnet are both consist of 4 convolutional layers, and each convolutional layer is followed by a ReLU activation. The `class` subnet predicted the probability of object presence in every anchor at each spatial position, whereas the `box` subnet regressed the offsets of the generated anchors to any nearby ground-truth object in parallel. The design of the `box` subnet was identical to the classification subnet except that it output 4 channels, standing for 4 vertices of the bounding box, per spatial location. Then, for each pyramid level, the NMS algorithm was applied to rank and output 1000 top-scoring bounding boxes. Next, the RetinaNet Head filtered out all the prediction results with a certain threshold.

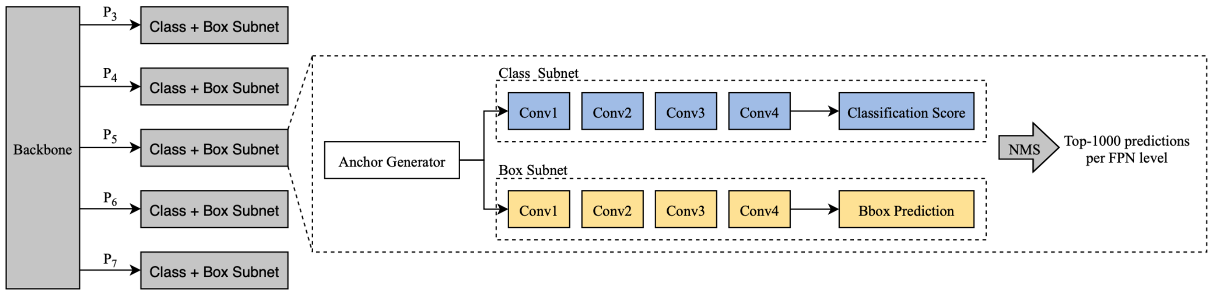


Figure 3.7: The structure of RetinaNet Head.

Implementation of the RetinaNet Head In this study, the anchor generator at each level was implemented to place 9 anchors at each spatial position, which were with 3 aspect ratios {1:2, 1:1, 2:1} and sizes $\{2^0, 2^{1/3}, 2^{2/3}\}$of the original set of 3 aspect ratios. The parameters of convolutional layers in RetinaNet Head are listed as Table 3.4.

Verification of the Implemented Object Detectors

The accuracy of the implemented object detectors was evaluated with the AP (Average Precision) metrics under the object detection scenario to verify the correctness of our implementation. However, the performance of the Faster RCNN framework was not evaluated

Names	Layers	#Input	#Output	Kernel	Stride	Padding
Class Subnet	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
Box Subnet	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
	2-D Convolution	256	256	3×3	1×1	1×1
Class Score	2-D Convolution	256	720	3×3	1×1	1×1
Box Prediction	2-D Convolution	256	36	3×3	1×1	1×1

Table 3.4: The parameters of RetinaNet Head. #Input and #Output stand for the channel dimension of the inputs and outputs.

with a ResNet-FPN backbone in its paper [71]. In addition, the performance of the RetinaNet framework was only tested on the COCO dataset by its authors Lin et al. [47]. And Lin et al. compared the accuracy of the Faster RCNN and the RetinaNet with a 101-layer version of the ResNet-FPN backbone. Thus, the implemented Faster RCNN and RetinaNet adopting ResNet-50-FPN backbone were verified on the COCO dataset with the metrics listed in Table 3.5. The results compared with the data of with ResNet-101-FPN from Lin et al. were listed in the Table 3.6.

Metric	Meaning
AP	AP at IoU $= .50 : .05 : .95$
AP_{50}	AP at IoU $= .50$
AP_{75}	AP at IoU $= .75$
AP_s	AP for small objects: area < 322
AP_m	AP for medium objects: $322 <$ area < 962
AP_l	AP for large objects: area > 962

Table 3.5: The metrics used for evaluating the accuracy (Average Precision) of object detectors.

According to the data in Table 3.6, with both ResNet-50-FPN and ResNet-101-FPN, the RetinaNet framework had a higher overall accuracy than the Faster RCNN framework,

Backbone	Framework	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-101-FPN (Lin et al.)	Faster RCNN	36.2	59.1	39.0	18.2	39.0	48.2
	RetinaNet	**39.1**	59.1	**42.3**	**21.8**	**42.7**	**50.2**
ResNet-50-FPN (mine)	Faster RCNN	32.4	**51.8**	35.3	14.9	**33.9**	41.5
	RetinaNet	**33.3**	51.4	**35.8**	**15.0**	32.9	**45.8**

Table 3.6: Object detection results (bounding box AP), vs. the results in the paper [47] on the COCO dataset.

while the former one didn't have a higher AP_{50}. Due to the lower error rate of 101-layer ResNet-FPN [29], the implemented object detectors had slightly lower AP than the 101-layer version implemented by Lin et al. Additionally, Lin et al. [47] recorded the AP of the RetinaNet framework with the 50-layer version of the ResNet-FPN backbone, varying from 30.5 to 35.7. And the AP of the RetinaNet detector implemented in this study (33.3) was in that range. Thus, the implemented object detectors were verified.

3.2.3 Text Detection Frameworks

This study selected the pre-trained text detector provided by PaddleOCR[8][18, 17], which is a practical ultra lightweight OCR system, providing both libraries for text detection and text recognition. Both two selected text detectors, SAST[85] and DB-Net[44], were adopted ResNet50 as the backbone and trained on the dataset ICDAR2015 [35] for a fair evaluation.

3.3 Experimental Setup

Experiments were all run on one Tesla K80 GPU, with 11441.0 MB memory, on Google Colab[9], a cloud platform of computation, with two logical CPUs, 13 GB RAM.

Additionally, the object detectors were trained on both `COCO1000` and `SynthText498` from the pre-trained model provided by `detectron2` for 200 iterations with a learning rate of 0.0125, and the threshold for predictionwas set as 0.7.

[8]Source code: https://github.com/PaddlePaddle/PaddleOCR
[9]https://colab.research.google.com/?utm_source=scs-index

Chapter 4

Result Analysis

4.1 Experiment Results

This section lists the results of running object detectors and text detectors on two datasets. Additionally, the reasons for the results are also explained in this section.

4.1.1 Object Detectors

Metrics	Faster RCNN	RetinaNet
AUROC	0.85	0.93
Precision	0.85	0.93
Specificity	0.86	0.96
Accuracy	0.82	0.77
Recall	0.78	0.57
F1 score	0.81	0.71
Throughput	2.885	2.491
Training time	542 s	529 s

(a) The results on `COCO1000`.

Metrics	Faster RCNN	RetinaNet
AUROC	0.96	0.97
Precision	0.73	0.92
Specificity	0.63	0.92
Accuracy	0.81	0.92
Recall	0.98	0.91
F1 score	0.84	0.91
Throughput	2.806	2.489
Training time	571 s	556 s

(b) The results on `SynthText498`.

Table 4.1: The results of object detectors on two datasets. The unit of throughput is image per second.

As the test results listed in Table 4.1, the Faster RCNN had an overall excellent performance on both scene text images and born-digital images, whereas the RetinaNet performed significantly better on the born-digital dataset yet achieved a barely satisfactory recall and accuracy on the scene text dataset. However, although the RetinaNet had a lower training time than the Faster RCNN as expected, the RetinaNet had a lower throughput than the Faster RCNN, which contrasts the conclusion drawn in the literature review.

Explanation

This section provides a potential reason of why the RetinaNet had a lower throughput than the Faster RCNN in this study, which is contrary to our expectation. As discussed in Section 3.2.1, each stage of the backbone network halved the size of the output feature maps. Notating the width and height of an input image as H and W, and the level of the stage as l, then the size of the feature map at level l is $^1/_{2^l}H \times {}^1/_{2^l}W$. The sizes of ResNet50 outputs were listed in Table 4.2a, and the size of FPN outputs in each level was listed in Table 4.2b.

Names	Height	Width	Channel
Input	H	W	3
C_2	$^1/_4H$	$^1/_4W$	256
C_3	$^1/_8H$	$^1/_8W$	512
C_4	$^1/_{16}H$	$^1/_{16}W$	1024
C_5	$^1/_{32}H$	$^1/_{32}W$	2048

(a) The size of ResNet50 outputs.

Names	Height	Width	Channel
P_2	$^1/_4H$	$^1/_4W$	256
P_3	$^1/_8H$	$^1/_8W$	256
P_4	$^1/_{16}H$	$^1/_{16}W$	256
P_5	$^1/_{32}H$	$^1/_{32}W$	256
P_6	$^1/_{64}H$	$^1/_{64}W$	256
P_7	$^1/_{128}H$	$^1/_{128}W$	256

(b) The size of FPN outputs.

Table 4.2: The size of the outputs of ResNet50 and FPN.

In this study, the Faster RCNN detector was implemented to generate 3 anchors at each spatial position on every level of the feature map. Thus, the Faster RCNN detector generated $^{1023}/_{4096}HW$ anchors for an input image, which was calculated as Equation 4.1.

$$N_{\text{anchors}} = \sum_{l=2}^{6} \frac{1}{2^l}H \times \frac{1}{2^l}W \times 3 = \frac{1023}{4096}HW \tag{4.1}$$

And as Equation 4.2 shows, the implemented RetinaNet detector generated 9 anchors at each spatial position and ${}^{3069}/{}_{16384}HW$ anchors in total for an input image.

$$N'_{\text{anchors}} = \sum_{l=3}^{7} \frac{1}{2^l} H \times \frac{1}{2^l} W \times 9 = \frac{3069}{16384} HW \tag{4.2}$$

For instance, if the size of input image is 800×1280, then the size of P_2 to P_7 would be $\{200 \times 320, 100 \times 160, 50 \times 80, 25 \times 40, 13 \times 20, 7 \times 10\}$, and the anchor generator of the Faster RCNN should generate 255780 anchor boxes, while the anchor generator of RetinaNet should generate 194970 anchor boxes. As discussed in the Section 3.2.2, though the RetinaNet generated fewer anchors than the Faster RCNN, the RetinaNet calculated the offsets and the objectness on each anchor box before ranking the top-scoring regions with NMS at each level, while the Faster RCNN calculated the offsets and objectness after the first NMS sifting the top-1000 proposals. Thus, Faster RCNN calculated the objectness and offset of fewer anchors could be the potential reason for the observation that Faster had a higher throughput than RetinaNet.

4.1.2 Text Detectors

Metrics	DB	SAST
Precision	0.50	0.87
Specificity	0.24	0.88
Accuracy	0.51	0.84
Recall	0.79	0.80
F1 score	0.61	0.83
Throughput	8.997	2.530

(a) The results on `COCO1000`.

Metrics	DB	SAST
Precision	0.56	0.83
Specificity	0.20	0.80
Accuracy	0.60	0.88
Recall	0.99	0.95
F1 score	0.72	0.89
Throughput	7.544	1.828

(b) The results on `SynthText498`.

Table 4.3: The results of text detectors on two datasets. The unit of throughput is image per second.

The test results of text detectors on two types of datasets are listed in Table 4.3. Compared to the results of object detectors, the performance of both two text detectors appeared no significant distinction between two different types of datasets. And the SAST performed better under the evaluation of all correctness metrics on both datasets. However, due to the light-weighted detection head, the DB had 3 to 4 times higher throughput than the SAST.

4.2 Discussion

The results indicate no strong correlation between the number of stages and the performance for the object and text detectors. The lower number of stages does not imply the performance of the detector, such as higher throughputs or lower accuracy. For instance, the one-stage object detector has a slightly slower inference speed than the two-stage object detector, which is contrary to the claims of Lin et al. [47] and explained with a potential reason in Section 4.1.1.

As for practical application on building a multimodal hateful speech detection system, the observation with the limited dataset implies that both text detection frameworks were good candidates to build the text or non-text classifier. Considering that most hateful memes are categorized as born-digital images, the Faster RCNN would be an economical choice to sift out memes from an extremely high volume of images due to the high AUROC it obtained on the `SynthText` dataset. Additionally, the high accuracy of the SAST and RetinaNet suggested that they could be a good choice for the task requiring high precision on classifying a small number of images, such as the images uploaded by the tagged unfriendly users.

However, the reliability of this data is impacted by the size of the datasets. And the generalizability of the results is limited since the dataset was designed based on the text detection datasets instead of the real user-uploaded images. In addition, the results cannot compare the performance between a text detector and an object detector, constrained by the methodological choices. The future studies that should be taken into account are listed in the next chapter.

Chapter 5

Future Work

Future works are required to obtain a more conclusive result by enriching the benchmark datasets. Firstly, the limited size of the benchmark should be addressed by future work. As for implementing a better classifier to distinguish text and non-text images, the deep learning methods need large datasets to minimize both bias and variance. As for evaluating the existing methods, testing detectors with larger datasets provide more solid results, since the higher volume of data helps to reduce the impact of outliers. Secondly, the real-world datasets, which are supposed to be collected from social media, would be a more valuable benchmark to evaluate existing detectors in the scenario of classifying positive and negative text images. User-uploaded images on the Internet are way more chaotic and complicated than just scene text images plus born-digital images.

In addition, future works are required to draw a more informative conclusion by performing a cross-sectional comparison. In other words, the results comparing text and object detectors would have a more practical application on choosing a proper framework for a text/non-text image classifier. It suggests further experiments, which evaluate those detectors without implementation bias, could be done.

Furthermore, a more comprehensive review on categorizing text detection frameworks according to the number of stages would be interesting to see. The mainstream text detection categorization classifies text detection architectures based on the methods. Thus, it is more challenging to define "stage" fairly and classify text detectors according to it. This study classifies a one-stage text detector from a two-stage one simply by whether they need text instance segmentation phases. However, this classification principle cannot be applied to all the text detectors, since some text detectors state their frameworks including no less than two stages yet no instance segmentation phases. For example, the Text-

Block FCN [91] contains five phrases without any segmentation procedure: (1) salient map generation; (2) text block generation; (3) candidate character component extraction; (4) orientation estimation; (5) text line candidates extraction.

References

[1] Visual object classes challenge 2012 (voc2012).

[2] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[3] Tina Askanius. On frogs, monkeys, and execution memes: Exploring the humor-hate nexus at the intersection of neo-nazi and alt-right movements in Sweden. *Television & New Media*, 22(2):147–165, 2021.

[4] Lubna Aziz, Md. Sah Bin Haji Salam, Usman Ullah Sheikh, and Sara Ayub. Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. *IEEE Access*, 8:170461–170495, 2020.

[5] Xiang Bai, Baoguang Shi, Chengquan Zhang, Xuan Cai, and Li Qi. Text/non-text image classification in the wild with convolutional neural networks. *Pattern Recognition*, 66:437–446, 2017.

[6] Xiang Bai, Baoguang Shi, Chengquan Zhang, Xuan Cai, and Li Qi. Text/non-text image classification in the wild with convolutional neural networks. *Pattern Recognition*, 66:437–446, 2017.

[7] Xiang Bai, Baoguang Shi, Chengquan Zhang, Xuan Cai, and Li Qi. Text/non-text image classification in the wild with convolutional neural networks. *Pattern Recognition*, 66:437–446, 2017.

[8] Eric Barendt. What is the harm of hate speech? *Ethical Theory and Moral Practice*, 22(3):539–553, 2019.

[9] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Minesh Mathew, CV Jawahar, Ernest Valveny, and Dimosthenis Karatzas. Icdar 2019 com-

petition on scene text visual question answering. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1563–1570. IEEE, 2019.

[10] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[11] Xiangrong Chen and Alan L Yuille. Detecting and reading text in natural scenes. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[12] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021.

[13] Chee Kheng Chng, Yuliang Liu, Yipeng Sun, Chun Chet Ng, Canjie Luo, Zihan Ni, ChuanMing Fang, Shuaitao Zhang, Junyu Han, Errui Ding, et al. Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1571–1576. IEEE, 2019.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[15] Richard Dawkins and Nicola Davis. *The Selfish Gene.* Macat Library, 2017.

[16] Gabriele de Seta. Memes in digital culture. *New Media & Society*, 17(3):476–478, 2015.

[17] Yuning Du, Chenxia Li, Ruoyu Guo, Cheng Cui, Weiwei Liu, Jun Zhou, Bin Lu, Yehua Yang, Qiwen Liu, Xiaoguang Hu, et al. Pp-ocrv2: Bag of tricks for ultra lightweight ocr system. *arXiv preprint arXiv:2109.03144*, 2021.

[18] Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, et al. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*, 2020.

[19] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970. IEEE, 2010.

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

[21] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. Ieee, 2008.

[22] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[23] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[24] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[25] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] Shehzad Muhammad Hanif and Lionel Prevost. Text detection and localization in complex scene images using constrained adaboost algorithm. In *2009 10th international conference on document analysis and recognition*, pages 1–5. IEEE, 2009.

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Hiroto Honda. Digging into detectron 2, Jul 2020.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[32] Masakazu Iwamura, Takahiro Matsuda, Naoyuki Morimoto, Hitomi Sato, Yuki Ikeda, and Koichi Kise. Downtown osaka scene text dataset. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 440–455, Cham, 2016. Springer International Publishing.

[33] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

[34] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

[35] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. ICDAR 2015 competition on Robust Reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[36] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013.

[37] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Casey A Fitzpatrick, Peter Bull, Greg Lipstein, Tony Nelli, Ron Zhu, et al. The Hateful Memes Challenge: Competition report. In *NeurIPS 2020 Competition and Demonstration Track*, pages 344–360. PMLR, 2021.

[38] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The Hateful Memes Challenge: Detecting hate speech in multimodal memes. *arXiv preprint arXiv:2005.04790*, 2020.

[39] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. 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, 2003.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[42] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE transactions on image processing*, 27(8):3676–3690, 2018.

[43] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[44] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11474–11481, 2020.

[45] Han Lin, Peng Yang, and Fanlong Zhang. Review of scene text detection and recognition. *Archives of computational methods in engineering*, 27(2):433–454, 2020.

[46] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[47] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[48] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[49] Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, and Helen Yannakoudakis. A multimodal framework for the detection of hateful memes. *arXiv preprint arXiv:2012.12871*, 2020.

[50] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[51] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[52] Xiyan Liu, Gaofeng Meng, and Chunhong Pan. Scene text detection and recognition with advances in deep learning: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(2):143–162, 2019.

[53] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision*, 129(1):161–184, 2021.

[54] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[55] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.

[56] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[57] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209, 2021.

[58] Ryan M. Milner. Media lingua franca: Fixity, novelty, and vernacular creativity in Internet memes. *AoIR Selected Papers of Internet Research*, 3, Oct. 2013.

[59] Niklas Muennighoff. Vilio: State-of-the-art visio-linguistic models applied to hateful memes. *arXiv preprint arXiv:2012.07788*, 2020.

[60] Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, et al. Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition—rrc-mlt-2019. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1582–1587. IEEE, 2019.

[61] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1454–1459. IEEE, 2017.

[62] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, Wafa Khlif, Muhammad Muzzamil Luqman, Jean-Christophe Burie, Cheng-lin Liu, and Jean-Marc Ogier. ICDAR2017 Robust Reading Challenge on multi-lingual scene text detection and script identification - RRC-MLT. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1454–1459, 2017.

[63] John T Nockleby. Hate speech. *Encyclopedia of the American constitution*, 3(2):1277–1279, 2000.

[64] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[66] Tekla S. Perry. Q&A: Facebook's CTO is at war with bad content, and AI is his best weapon, Jul 2021.

[67] Zobeir Raisi, Mohamed A Naiel, Paul Fieguth, Steven Wardell, and John Zelek. Text detection and recognition in the wild: A review. *arXiv preprint arXiv:2006.04305*, 2020.

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

[69] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[70] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[71] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[73] Benet Oriol Sabat, Cristian Canton Ferrer, and Xavier Giro-i Nieto. Hate speech in pixels: Detection of offensive memes towards automatic moderation. *arXiv preprint arXiv:1910.02334*, 2019.

[74] Vlad Sandulescu. Detecting hateful memes using a multimodal deep ensemble. *arXiv preprint arXiv:2012.13235*, 2020.

[75] Asif Shahab, Faisal Shafait, and Andreas Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *2011 international conference on document analysis and recognition*, pages 1491–1496. IEEE, 2011.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[77] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *arXiv preprint arXiv:1507.06228*, 2015.

[78] Yash Srivastava, Vaishnav Murali, Shiv Ram Dubey, and Snehasis Mukherjee. Visual question answering using deep learning: A survey and performance analysis. In *International Conference on Computer Vision and Image Processing*, pages 75–86. Springer, 2020.

[79] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[80] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 international conference on computer vision*, pages 1879–1886. IEEE, 2011.

[81] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[82] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge J. Belongie. COCO-Text: Dataset and benchmark for text detection and recognition in natural images. *CoRR*, abs/1601.07140, 2016.

[83] Riza Velioglu and Jewgeni Rose. Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge. *arXiv preprint arXiv:2012.12975*, 2020.

[84] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[85] Pengfei Wang, Chengquan Zhang, Fei Qi, Zuming Huang, Mengyi En, Junyu Han, Jingtuo Liu, Errui Ding, and Guangming Shi. A single-shot arbitrarily-shaped text detector based on context attended multi-task learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1277–1285, 2019.

[86] Lyndon CS Way. The importance of memes. *European Journal of Communication*, 34(5):552–556, 2019.

[87] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[88] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090, 2012.

[89] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1480–1500, 2014.

[90] Weibo Zhang, Guihua Liu, Zhuohua Li, and Fuqing Zhu. Hateful memes detection via complementary visual and linguistic networks. *arXiv preprint arXiv:2012.04977*, 2020.

[91] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4159–4167, 2016.

[92] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[93] Ron Zhu. Enhance multimodal transformer with external label and in-domain pre-train: Hateful meme challenge winning solution. *arXiv preprint arXiv:2012.08290*, 2020.

[94] Yingying Zhu, Cong Yao, and Xiang Bai. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, 2016.

[95] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.