



NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

**SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

MSc THESIS

**Multilingual Text Detection on Scene Images using MASK
RCNN Method**

Nikolaos V. Naoum

Supervisor: Vasilios Katsouros, Research Director

ATHENS

MARCH 2021



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Multilingual Text Detection on Scene Images using MASK
RCNN Method**

Νικόλαος Β. Ναούμ

Επιβλέπων: Βασίλειος Κατσούρος, Ερευνητής Α'

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2021

MSc THESIS

Multilingual Text Detection on Scene Images using MASK RCNN Method

Nikolaos V. Naoum

S.N.: DS1180015

SUPERVISOR: **Vasilios Katsouros**, Research Director

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Multilingual Text Detection on Scene Images using MASK RCNN Method

Νικόλαος Β. Ναούμ

A.M.: DS1180015

ΕΠΙΒΛΕΠΩΝ: **Βασίλειος Κατσούρος, Ερευνητής Α'**

ABSTRACT

In the new era of technology, where innovations come day by day the new ideas, methods, procedures in the field of Computer sciences getting more advance. One of them research becomes most active topic these days is 'text detection and recognition' presents in images or videos. The accurate information present in the text is very useful for a wide range of real-life applications. However, it is a very complicated assignment to localize and read texts from natural scene images. Scene text detection and recognition application, due to majority and variety of these applications present in the market, it seeks the attention of community to the computer technology more and become curios. There are some problems which are unsolved till know in the world of text detection which are languages, colors, orientation, fonts, style that need to be resolved. The recent advancements in deep learning have increased the attention of potential researchers towards scene text detection. CNN is designed in the way that it automatically adapts spatial hierarchies of Text detection Processing features through using multiple building blocks - layers. First, it collects the text each word separately and after detecting the different parts of text it recollects the whole image and then present an output. In this report it is analyzed the techniques like LOMO and PMTD etc. for text detection. Our proposed method is using MASK RCNN technique and it is implemented and tested in order to offer a framework that has a powerful baseline and offers so many advantages such as flexibility, robustness, fast time of training and inference. All the methods help us to achieve two different activities which are instance segmentation and text detection on scene images. The methods that are proposed in the research have already offers some application that include the Multilingual text detection on scene images, but this is not meant that these applications are ideal to use or perfect. There is a lag and missing features in these applications and have a room for improvement which can help in achieving better results in terms of boundary detection. The world of technology improving day by day and being update with the technology and contribute in the technology is the best way to express greeting or thanks to the technology.

SUBJECT AREA: Multilingual Text Detection

KEYWORDS: text detection, CNN, detector, ICDAR, competition, Deep Learning, Computer Vision, bounding box, datasets, RCNN, algorithm, recognition, MASK R-CNN, hyper parameters

ΠΕΡΙΛΗΨΗ

Στη νέα εποχή της τεχνολογίας, όπου οι καινοτομίες αυξάνονται μέρα με τη μέρα, οι νέες ιδέες, μέθοδοι, διαδικασίες στον τομέα της επιστήμης υπολογιστών απαιτούνται όλο και περισσότερο. Ένα από τα πιο ενεργά θέματα αυτές τις μέρες είναι η «ανίχνευση και αναγνώριση κειμένου» σε εικόνες ή βίντεο. Οι ακριβείς πληροφορίες που υπάρχουν στην εικόνα είναι πολύ χρήσιμες για ένα ευρύ φάσμα εφαρμογών στην πραγματική ζωή. Ωστόσο, είναι μια πολύ περίπλοκη διαδικασία να εντοπιστεί και να αναγνωριστεί κείμενο σε εικόνες σκηνής. Η ανίχνευση και αναγνώριση κειμένου σε εικόνες σκηνής, λόγω της ποικιλίας των εφαρμογών που υπάρχουν στην αγορά, αναζητά την προσοχή της κοινότητας της τεχνολογίας των υπολογιστών όλο και περισσότερο. Υπάρχουν ορισμένα προβλήματα σε αυτόν τον τομέα που δεν έχουν επιλυθεί ακόμα όπως είναι πολυγλωσσία, χρώματα, προσανατολισμοί, γραμματοσειρές, στυλ. Οι πρόσφατες εξελίξεις στη βαθιά μάθηση έχουν αυξήσει την προσοχή δυνητικών ερευνητών στην ανίχνευση κειμένου. Η αποτελεσματικότητα των Convolutional Neural Networks βασίζεται σε μεγάλο βαθμό στην απόδοση του αλγορίθμου που υιοθετείται για την ανίχνευση αντικειμένου. Υπάρχουν πολλές δυσκολίες που πρέπει να αντιμετωπιστούν οι οποίες σχετίζονται με την ανίχνευση κειμένου σκηνής. Το μεγαλύτερο πρόβλημα είναι ότι οι περισσότερες από τις μεθόδους που χρησιμοποιούνται για την ανίχνευση κειμένου δείχνουν καλύτερη απόδοση όταν οι συνθήκες είναι υπό έλεγχο, όταν οι περιπτώσεις στις οποίες το κείμενο έχει κανονικό σχήμα και κανονική αναλογία. Λόγω περιορισμένων μορφών αναπαράστασης κειμένου και περιορισμένου δεκτικού μεγέθους CNN, οι μέθοδοι αυτοί δεν εντοπίζουν τις πολύπλοκες σκηνές, όπως κείμενα που έχουν αυθαίρετο σχήμα ή είναι μακρά κείμενα. Η προτεινόμενη μέθοδος που εφαρμόζεται και δοκιμάζεται είναι η Mask RCNN για να προσφέρει ένα πλαίσιο που έχει μια ισχυρή βάση και προσφέρει πολλά πλεονεκτήματα θεσμικής σαφήνειας στην έννοια, την ευελιξία, την ευρωστία και τον γρήγορο χρόνο εκμάθησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανίχνευση πολύγλωσσου κειμένου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ανίχνευση κειμένου, βαθιά δίκτυα, Mask RCNN, υπολογιστική όραση, αλγόριθμος, αναγνώριση κειμένου

Στην σύντροφο της ζωής μου Ιουλία και τα παιδιά μου Βασίλη και Όλγα.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα, Ερευνητή Α' Βασίλειο Κατσούρο, για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωση της.

CONTENTS

1. INTRODUCTION	15
1.1 History/Background	15
1.2 Transfer learning	17
1.3 Key components of Deep Learning Model	18
1.1.1 Convolutional Neural Network - CNN	18
1.1.2 Components of Convolution Neural Network	19
2. PROBLEM DEFINITION	20
3. COMPETITION DESCRIPTION	21
3.1 RCC-MLT-2017	21
3.2 RCC MLT-2019	21
3.3 Tasks and evaluation	22
3.3.1 Task I	22
3.3.2 Result format	22
3.3.3 Evaluation	22
3.3.4 Task II	23
3.3.5 Format of the results	23
3.3.6 Evaluation	23
4. RELATED WORK	24
4.1 Multi Stage Text Detector - LOMO	25
4.1.1 Introduction	25
4.1.2 Problems which are meant to be resolved	25
4.1.3 Architecture of LOMO	26
4.1.4 Direct Regressor	26
4.1.5 Iterative Refinement Module	27
4.1.6 Shape Expression Module	29
4.2 Text Polygon Generator	30
4.2.1 Interface	31
4.2.2 Inference Phase	32
4.2.3 Results achieved from the experiment	32

4.2.4	Analysis of the features	33
4.2.5	Structural method	33
4.2.6	Applications which this model is offering.....	34
5.	PMTD – PYRAMID MASK TEXT DETECTOR	35
5.1	Introduction	35
5.2	Architecture	35
5.2.1	Pyramid label.....	35
5.2.2	Plane clustering.....	38
5.3	Datasets.....	40
6.	IMPLEMENTED MODEL - MASK RCNN	41
6.1	Problems which need to be resolved	41
6.2	Additions made by MASK R-CNN method	41
6.3	Architecture of MASK R-CNN model	42
6.4	FASTER R-CNN	42
6.5	MASK R-CNN	42
6.6	Mask representation	44
6.7	RoIAlign.....	44
6.8	Architecture of the MASK R-CNN network	46
6.9	Architecture of the head.....	47
6.10	MASK R-CNN model architecture convolutional network based on masked region 47	
7.	IMPLEMENTATION	48
7.1	MASK R-CNN model implementation	48
7.1.1	Engineering activities	50
7.1.2	Tested hyper parameters	52
7.1.3	Prediction/ object detection and segmentation	54
7.1.4	Head architecture	58
7.1.5	Data set	58
7.1.6	Training.....	58
8.	EVALUATION	62

8.1	Ablation experiment.....	64
8.1.1	Multinomial vs independent mask	65
8.1.2	Class specific vs class-agnostic masks	65
8.2	Mask branch-segmentation	66
8.3	Bounding box detection results.....	67
8.3.1	Evaluation of MASK R-CNN for human pose estimation	69
8.3.2	Over-simplified supervision	69
8.3.3	Imprecise segmentation labels.....	70
8.3.4	Error propagation.....	70
9.	RESULTS.....	71
10.	FUTURE WORK.....	75
11.	ANNEX I.....	77
12.	REFERENCES	82

LIST OF FIGURES

Figure 1.1	20
Figure 1.2.....	21
Figure 1.3.....	21
Figure 4.1	28
Figure 4.2.....	30
Figure 4.3.....	31
Figure 5.1	37
Figure 5.2.....	38
Figure 5.3.....	39
Figure 5.4.....	42
Figure 6.1	45
Figure 6.2.....	46
Figure 6.3.....	48
Figure 7.1	51
Figure 7.2.....	52
Figure 7.3.....	53
Figure 7.4.....	56
Figure 7.5.....	57
Figure 7.6.....	58
Figure 7.7.....	59
Figure 7.8.....	59
Figure 7.9.....	61
Figure 7.10.....	62

Figure 8.1	65
Figure 8.2	66
Figure 8.3	72
Figure 9.1	74
Figure 9.2	74
Figure 9.3	75
Figure 9.4	75
Figure 9.5	76
Figure 9.6	76
Figure 11.1	84
Figure 11.2	84
Figure 11.3	85

LIST OF TABLES

Table 4.1	34
Table 6.1	47
Table 7.1	50
Table 8.1	64
Table 8.2	66
Table 8.3	67
Table 8.4	68
Table 8.5	68
Table 8.6	68
Table 8.7	69
Table 8.8	70
Table 9.1	73
Table 9.2	73

1. INTRODUCTION

This section will provide a detailed discussion on the history or background of scene text and its recognition. This study focuses on a competition in the field of text recognition with computer vision. The survey of related work, along with the problem definition, is also part of this discussion. Delimitations associated with the proposed study are also presented. In the end, gender aspects and ethical considerations are provided. Further things are given in detail below.

1.1 History/Background

In natural scenes, the text carries high-level semantics as a product of human manipulation. This is one of the most key properties of text that makes it an important source of information both in images and videos. The accurate information present in the text is very useful for a wide range of real-life applications. The applications include Human-Computer Interaction (HCI) [1], target geolocation [2], industrial automation [3], navigation of robots [4], image search, and many more [5]. Research scientists have declared automatic text detection with recognition one of the most active and trending topics these days as they offer a source to use textual information in images or videos. However, it is a very complicated assignment to localize and read texts from natural scenes [6].

Due to a variety of applications, scene text detection and recognition have drawn computer vision's community attention. A couple of unsolved problems still exist, including variations in fonts, colors, languages, orientations, and many other challenges that still need to be resolved [7,8]. The recent advancements in deep learning have increased the attention of potential researchers towards scene text detection. Many segmentation frameworks and effective Convolutional Neural Networks (CNN) have been developed to deal with various challenges associated with text detection. They include Fully Convolutional Network (FCN), Region-based Convolutional Neural Networks (R-CNN), and Single Shot multi-box Detector (SSD) [9,10,11]. A few approaches that detect texts tackle the problem through semantic segmentation which is a pixel-level classification of the image. As a result, a

saliency map gets generated, and it can only detect coarse text blocks. Thus, semantic segmentation by itself cannot distinguish between two objects in the same class which are close to each other. Therefore, complex steps are required in post-processing to extract the precise boundaries of texts.

Unlike the FCN (semantic segmentation) technique, more methods take text as a particular object or other leverages frameworks for object detection. They may include You Only Look Once (YOLO), SSD, Dense Box, and R-CNN [12]. These methods detect text lines or words as instances. A major limitation of these techniques is they are unable to deal with curved text detection. Some recent approaches, e.g., Fused Text Segmentation Network (FTSN), IncepText, and PixelLink, are developed to detect both curved as well as straight texts in a unified manner. The mentioned techniques consider text detection as instance segmentation issue [13,14,15]. IncepText and FTSN borrow Fully Convolutional Instance aware semantic Segmentation (FCIS) framework to resolve text detection issue whereas, the text is detected by linking pixels within the similar text instances in PixelLink technique [16].

The major challenges in the field of text detection can be divided into three different types as given below [17,18].

- Interference factors
- Diversity of scene text
- Complexity of background

Many interference factors are involved in creating problems in the detection and recognition of scene text, including distortion, low resolution, noise, partial occlusion, blur, illumination (non-uniformly distributed).

The characters normally consist of the same size, uniform management, and single font in most of the documents that are comparatively easy to detect, but all these parameters are different in the case of natural scenes, and hence it creates lots of problems while detecting and recognizing text in natural scenes.

The backgrounds are usually more complex in natural images or videos. Different elements, e.g., grass, fence, signs, cannot be distinguished from the true text. Therefore, it leads to many errors and confusion in precise text detection.

To tackle the above-mentioned challenges, many research studies have been introduced, and significant progress has been made to achieve the desired results, as discussed earlier. Despite achieving promising results, used instance segmentation techniques have been replaced by the state of the art and the latest algorithms. Mask R-CNN is one of the latest and effective approaches to detect and recognize every type of text in natural scenes. This study implements this model to get the desired results. Mask R-CNN model is helpful in the detection and recognition of curved as well as multi-oriented texts in a unified manner from different images and videos. Mask R-CNN utilizes both semantic segmentation in combination with object detection.

1.2 Transfer learning

Massive changes have been observed in the vision of computers due to progress in the sector of deep learning from the past few years. In computer vision, there are specific patterns that machine learning models consider in an image that is generated by the programmer, this is called feature engineering. Feature engineering in computer vision is the task of manually generating filters which the machine learning model uses when looking for patterns in the input image. However, deep learning offers the capability for the model itself to deduce the patterns of data from the bigger data sets and train itself as per the requirement. Deep learning models have surpassed human's capabilities in regards to speed and accuracy.

In deep learning the neural network is trained on larger data sets and capable of establishing rules which are required to explain the pattern which is very much unlike the traditional computer vision techniques. Traditionally when the software is developed the programmer is the one who establishes the rules which are then followed by the computer. In supervised machine learning the combination of training data and ground truth labels are supplied to the model. The prediction values which are achieved by the ground truths are compared by the deep learning model to be able to update the trainable model parameters.

1.3 Key components of Deep Learning Model

Neurons are the basic units of neural network weights and biases are the trainable parameters. The output which is achieved from the previous neuron is multiplied by the weight and a bias value is added. All the signals from the upcoming neurons are added up by the transfer function and passed on to the activation function. The activation function checks if the input is over a specific threshold. As per the strength of the input the activation function gets excited or not. The illustration below is showing an artificial neural network that has inputs from X_1 to X_n .

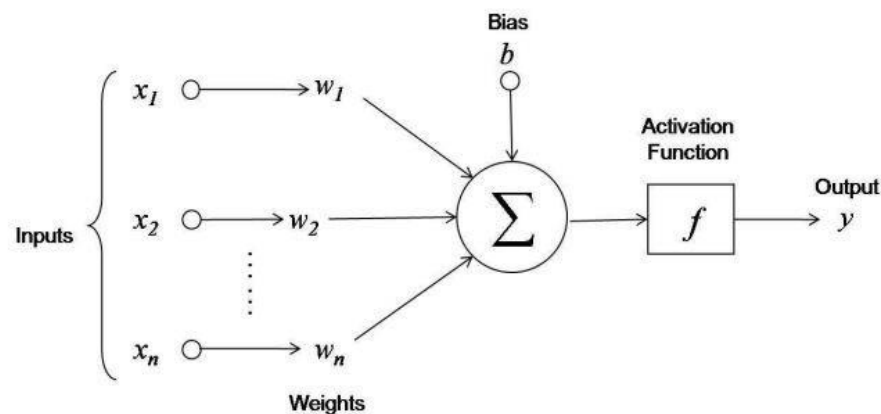


Figure 1.1 depicts the structure of an artificial neuron of which a deep neural network is built upon.

1.1.1 Convolutional Neural Network - CNN

A convolutional neural network is a subclass of deep learning and it is used widely in tasks in the field of computer vision. The architecture which is involved in the convolutional neural network facilitates picking up low-level features in the early layers of the network which are then combined with features that are more complex and deeper in the network. Figure 1.2 is showing the neural network which helps in detecting features that are the low key which is then combined in the later layers which are more complex.

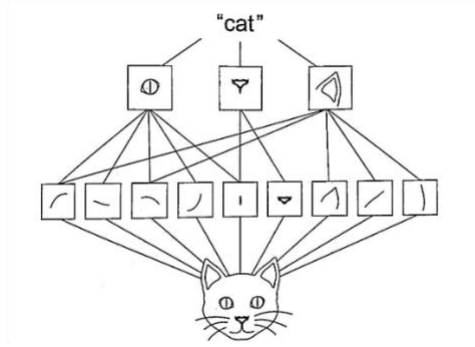


Figure 1.2 is a simplification of the structure of a neural network, with the early layers recognizing the simple features and the later layers combining the simple features into more complex shapes.

1.1.2 Components of Convolution Neural Network

The convolution neural network has four components which include neurons, layers, activation function, and loss function. The data is perceived by the CNN through convolutional filters which strides along with the image which is the input image. A convolution filter is a window that has width and dimension. The window travels along with the image. The filter calculates the dot product between the input image and filter. Images which are RGB the dimension of the window is 3, one for every color, which can be seen in Figure 1.3

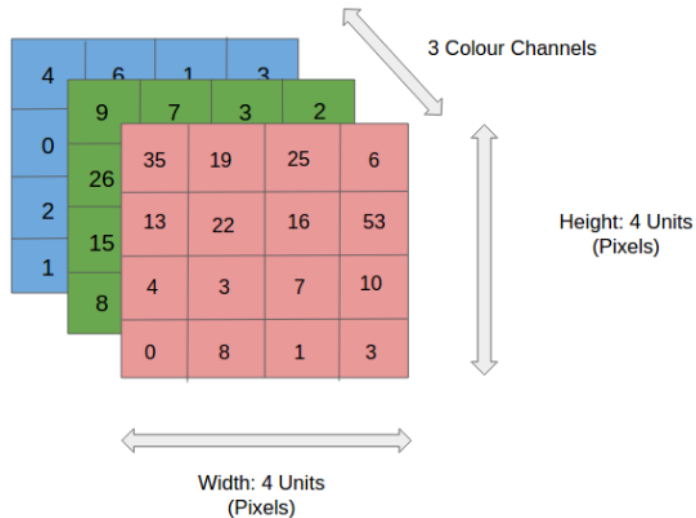


Figure 1.3 depicts the three dimensions of a filter of 4x4 size with three color channels/dimensions.

2. PROBLEM DEFINITION

Due to the increase in cosmopolitan activities in modern cities, there is a need for a detection technique and a recognition system that is immense and is a robust Multi-lingual scene text. The goal is to benchmark which is systematic and capable of providing a push to the state of art techniques. The competition was a step forward and built on RCC-MLT - 2017. Improvement and additions were made in the form of end-to-end task, in the image data set additional languages were utilized, a synthetic data set which was large scale and multilingual was utilized for assisting the training and an end-to-end recognition method was utilized as a baseline. The challenge was aimed to address four aspects concerning multilingual scene text.

- Text detection
- Classification of a script that has cropped words
- Joint detection and classification of the script
- Detection and recognition end-to-end

3. COMPETITION DESCRIPTION

Strong competition in the field of text detection and recognition exists. Some of the major competitions are provided below.

- ICDAR - RCC MLT-2017
- ICDAR - RCC MLT-2019

3.1 RCC-MLT-2017

The key component is reading the scene text. The reading of the scene text may be incorporated in a lot of applications which range from helping the visuals that are impaired. The information gained from the images is utilized in mapping services and systems which are used to provide geographical information. Larger integration systems like self-driving cars are also utilize recognition and scene text detection techniques. The challenge was related to the detection of scene text which was multilingual and the identification of scripts that were different. The research which was made for the detection of scene text and recognition was made considering the text which was in English. The English text had a wide range of data sets and benchmarks were well defined. The other lingual data sets focused on Arabic of French. The available data set contained only two scripts. The provided dataset contained a small number of images. The goal of this competition was to locate and classify the scripts in the images.

3.2 RCC MLT-2019

The given dataset in this competition was 20,000 natural scene images. The text instances from the following languages which include Arabic, Bangla, Chinese, Devanagari, English, French, German, Italian, Japanese, and Korean. Multi lingual Text detection

The main objective of this task is the localization of the text rightly which is multi-lingual in an image at the word level. There are the total number of scene images which are 10,000 and each image has a correspondence to the file containing the image ground truth (GT). The GT file has a coordinate list of bounding boxes for each text word that is present in the image. Four corner points that are in the clockwise direction represent the bounding boxes.

The test set contains images that are 10,000 in number. Each image that is participating is expected to produce bounding boxes that have four corners for every single word which is detected in the image. H-mean which is also known as f-measure is used as a ranking metric. Both recall and precision are utilized in the standard f-measure for the detection of word bounding boxes. Detection is counted correct if there is an overlap of 50% on the bounding boxes.

3.3 Tasks and evaluation

To be able to participate in the RCC-MLT-2019 challenge there is a requirement of participation in at least one task. The first three tasks in the MLT-2019 were similar to MLT-RCC-2017. However, data with new languages and the quality of the data set increased.

3.3.1 Task I

Task 1 was about the text detection of multiple texts. The method used for the detection of the multi-text script should be capable of generalizing the detecting text of different scripts. The input of this task is images with embedded text of various languages. The GT which is provided in this task contains information that is more than the requirement. As the GT is shared with Task3 and Task 4.

3.3.2 Result format

The required result format is an txt file for each image which the model has predicted on, each txt file should contain one row for every script detected on the image. The result for each detected word/script in an image should be saved as the coordinates surrounding the script X1, Y1, X2, Y2, X3, Y3, X4, Y4, and the valid scripts are Arabic, Latin, Chinese, Japanese, Korean, Bangla, Hindi, Symbols, Mixed, None.

3.3.3 Evaluation

For the evaluation of the method which is utilized, a metric is used for the ranking which is known as human or f-measure. The standard f-measure relies on the recall and the precision of the word bounding boxes that are detected. If overlap happens to be more than 50% the detection was considered as positive.

3.3.4 Task II

Task 2 is about the identification of a script that has cropped words. The text which is presented in the data set images is in 10 languages. Some images which are presented in the data set are sharing a similar script. Some words are assigned to a script which is known as Symbol. This Symbol script is special because punctuations and math symbols appear as an individual word. A total number of 8 scripts is available and the number is excluding the mixed and do not care script. To perform the identification, task all the cropped words which are presented in the data set as the separate image file. Ground truth script and transcription was provided along with the image file. Transcription file can be ignored.

3.3.5 Format of the results

Original information about the image is provided and the word images are extracted from the original image. In the ground truth file, each line has the following format

[word image name], x1, y1, x2, y2, x3, y3, x4, y4, [original image name]

The result should provide the scripts of each of the images. Whereas each image which is an input image is a cutout block from the scene image. A request for the script name for every single image is made.

[word image name], script

3.3.6 Evaluation

The results which are gained from the ground truth table are computed in the following way. A script ID is provided to each word image by the participants. In the case of a correct result, an increment is observed in the count of the correct result. The accuracy of prediction is something that determines the final evaluation of the method. It is summarized as follows

Let $G = \{g_1, g_2, \dots, g_i, \dots, g_m\}$ it represents the set of the script which are true.

$T = \{t_1, t_2, \dots, t_i, \dots, t_m\}$ represents the set of script classes that are returned by the given method.

4. RELATED WORK

The R-CNN is utilized the number of candidates in the region of object by bounding the box object detection and for the evaluation of convolutional network. For the attending of Rols on feature map with the utilization of RoIPool which helped in fast speed and better accuracy. Region Proposal Network RPN helped in achieving attention mechanism by learning to the faster R-CNN. The faster R-CNN is robust and flexible to the improvements. Faster R-CNN is helping and serving as an infrastructure to the many several researches. Due to effectiveness of R-CNN, several approaches of instance segmentation rely largely on segment proposal. Deep mask and other works are made learn to propose candidates that are presented in the segment. Then fast R-CNN classifies them. These methods are slow and less accurate because in this methods segmentation proceeds to the recognition. A complex multi stage cascade that is capable of predicting segment proposals with the utilization of bounding box proposal that was proposed by Dai et al. The implemented method relies largely on parallel prediction of masks and class labels, which is flexible and simpler as compared to the other. Instance segmentation is a combination of semantic segmentation + object detection. This combination was made to be able to predict a set that is sensitive to position and the output that is channeled is fully convolutional. These output channels make the system fast by simultaneously addressing the classes of the object, boxes and masks. However, on the instances like overlapping FCIS displays errors and shows edges that are spurious. Another solution was proposed in regards to the segmentation of instances. This solution was associated with the success of semantic segmentation. In this method, the pixels are cut which lie in the same category but in different instances. The cutting starts from the results of classification which is made as per the pixel.

4.1 Multi Stage Text Detector - LOMO

4.1.1 Introduction

The term LOMO is an abbreviation of Look More than Once. For a text which is arbitrary in shape there is a requirement of an accurate detector to take advantages of methods which are employed for detection and segmentation. Direct Regressor (DR) branch generated a quadrangle proposal of a text at first place. A feature block was extracted by refinement which was iterative on the proposal, then a long text was perceived by Iterative Refinement module (IRM). All the geometrical properties like region of the text, center line of the text and offsets of the borders were considered for the reconstruction of the irregular text by Shape expression module (SEM).

4.1.2 Problems which are meant to be resolved

Scene text detection

The effectiveness of CNN relies largely on the performance of algorithm which is adopted for the detection of object. There are multiple difficulties which needs to be address which are associated with the detection of scene text. The biggest problem is that most of the methods which are utilized for the detection of text shows better performance when conditions are under control. Controlled conditions means that the text is in regular shape and regular ratio. Due to limited representation forms of text and limited receptive size of CNN the methods fail to detect the complex scenes such as texts which have arbitrary shape and they are long texts.

Representation Problem

The text detection methods which are largely used apply representations which are simpler like rectangles which are aligned to the axis, rectangles which are rotated, or quadrangles etc. In cases when it has complex texts like curved or wavy texts the detection methods are likely to fail in such scenarios. The method LOMO is trying to fix all the issues which are related to the detection of the text.

4.1.3 Architecture of LOMO

The network architecture which was adopted for the LOMO relies on these three modules which are:

- Direct regressor DR
- Iterative Refinement module IRM
- Shape expression module SEM

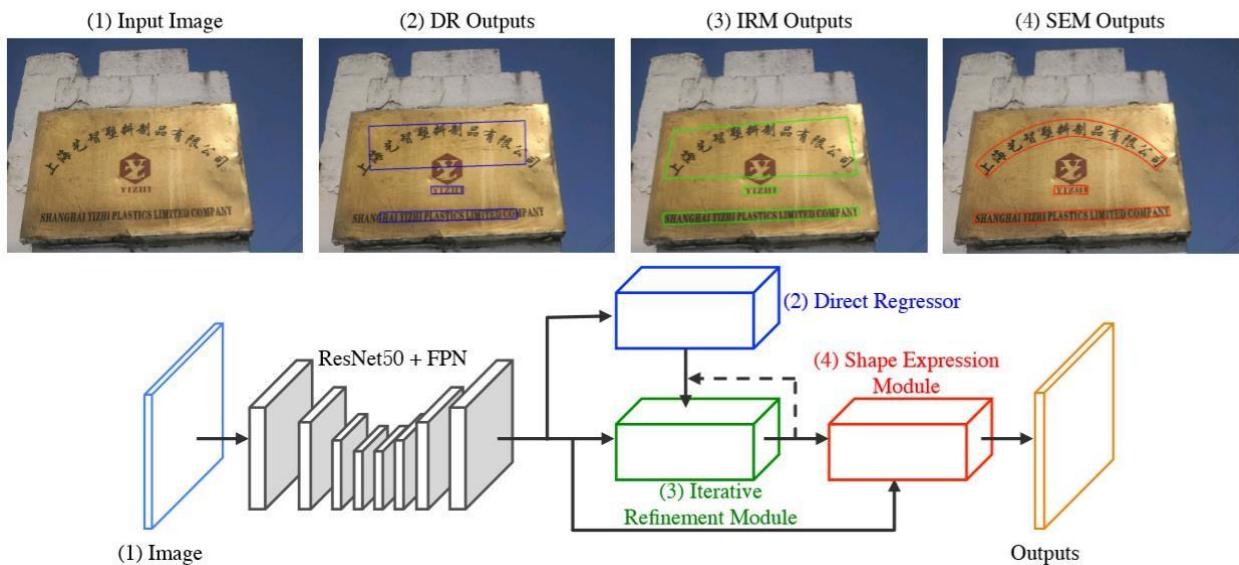


Figure 4.1 illustrate the architecture of LOMO.

As it can be seen in Figure 4.1 the architecture is divided into four parts. At first place it may have extracted the shared map of the feature for three branches which includes DR, IRM and SEM. The feature is shared by feeding the input image to the backbone of the network. The backbone of the network in this case is ResNet 50 along with FPN. In ResNet 50 the feature map of stages 2,3,4,5 is merged. The channel number is 128 and the size of map which is of shared image is $\frac{1}{4}$ of the size of the input image. Then it is sent to direct regressor for the prediction of the text.

4.1.4 Direct Regressor

For the direct regression of the text a sub network which is fully convolutional is adopted. To indicate the pixel wise confidence of the text a dense prediction channel of text is

calculated which is based on the maps of shared features. The pixels of the regions of original text which is a shrunk version is considered to be positive. The quadrangle which is containing the pixel an offset value is predicted to the four corners of the quadrangle. The channels which predict offset are 8 channels which are from the positive sample. Location regression term and text/non-text classification term are two branches on which loss function of DR consists of, text/non-text classification are regarded as binary segmentation task on the down sampled score map which is $\frac{1}{4}$. To improve the generalization of scale of DR for the detection of text instances under the field size which is receptive it proposed a version which is scale-invariant instead of directly utilizing the dice coefficient loss. The dice coefficient function which is in variant defined as following:

$$L_{cls} = 1 - \frac{2 * \text{sum}(y \cdot \hat{y} \cdot w)}{\text{sum}(y \cdot w) + \text{sum}(\hat{y} \cdot w)} \quad (1)$$

Where y is the 0/1 label map

\hat{y} is the predicted scope map

Sum is a cumulative function which is in 2D space

w is the weight map which is 2D

The normalized constant λ is divided by the shorter side of quadrangles from which they belong to get the value of positive positions. The value of negative positions is set to be 1.0. to optimize the term which is L_{loc} known as location regression smooth L1 is adopted.

$$L_{dr} = \lambda L_{cls} + L_{loc} \quad (2)$$

4.1.5 Iterative Refinement Module

The Region based object detector with the bounding box regression task only is inherited by the design of IRM. To extract the feature block of the input text which is quadrangle it utilizes RoI transform layer instead of RoI pooling layer or RoI align layer. Without even changing the aspect ratio the feature block of quadrangle proposal can be extracted by the former one in comparison to the latter two ones. In the same receptive field, the location which is close to the corner point can perceive accurate information regarding the

boundary. To regress the coordinate offsets of each corner a mechanism was introduced which is known as corner attention.

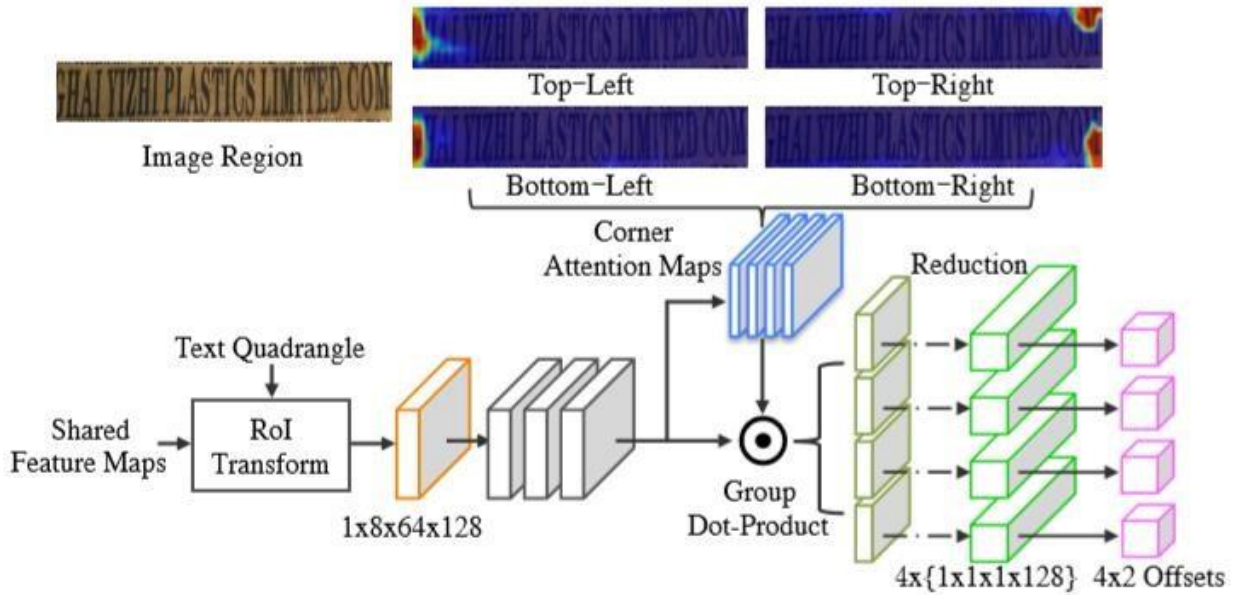


Figure 4.2 Iterative Refinement Module

Shared feature map to RoI transform layer is fed in case of one text quadrangle and then a feature block is obtained which is $1 \times 8 \times 64 \times 128$. To extract rich component which is named as f_r convolutional layer which is 3×3 are followed. To learn the attention maps known as m_a which have four corners a sigmoid layer and a convolutional layer which 1×1 is utilized. In order to support the offset regression of the respective corners the values which are presented on each attention corner map are denoted to the contribution weights. By sum reduction operation and group dot production regression features can be extracted which are 4 corners along with f_r and m_a .

$$f_c^i = reduce_sum (f_r \cdot m_a^i, axis = [1,2]) \quad | i = 1,$$

f_c^i denotes the i -th corner regression feature which has a shape $1 \times 1 \times 1 \times 128$

m_a^i is the i -th learned corner attention map

To predict the offset of 4 corners which is between the input quadrangle and the GT text book which is based on f_c which is a regression feature on the corner four headers are

applied. Each header consists of two convolutional layers which are 1×1 . In the training phase K is a detected quadrangle which is kept preliminary the corner regression loss can be represented as

$$L_{irm} = \frac{1}{K \times 8} \sum_{i=0}^n \sum_{j=1}^8 \text{smoothL1}(ckj - \hat{ckj})$$

Where ckj is the j -th coordinate offset between the k -th pair of quadrangles which are detected.

\hat{C}^kj is the corresponding predicted value.

In case of the corner regression which is respective the high support is represented when the response is strong on the four corners on the attention map. Refinement can be performed more than once while testing by the IRM.

4.1.6 Shape Expression Module

Traditionally when the text instances are of irregular shape like they are curved or wavy the text expression of quadrangle most of the time fails to describe. It proposed a module which is inspired by R-CNN to solve this problem which is shape expression module. A fully convolutional network which is followed by RoI transform layer is what it called SEM. A region where foreground pixels are marked as 1 and background pixels are marked as 0 is known as a text region which is a binary mask. A binary mask which is based on side shrunk version of annotation which is text polygon is known as text center line. The area which has valid values in case of a positive response on the locations which are corresponding to the text line map are known as 4 channel maps which have border offsets.

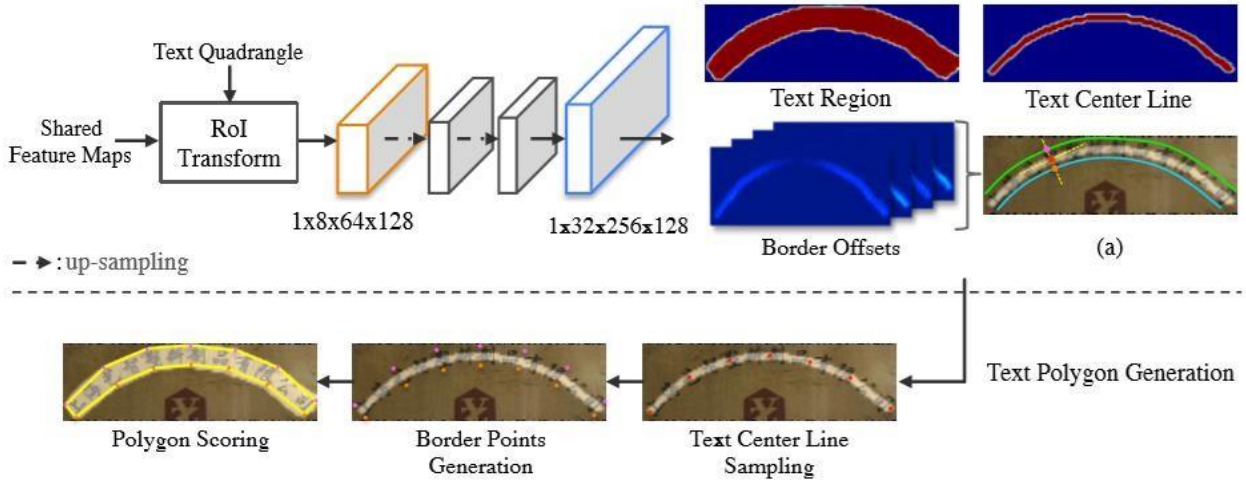


Figure 4.3 Shape Expression Module

The above illustration is showing the structure of SEM. The objective function of SEM is defined as follows

$$L_{sem} = \frac{1}{K} \sum (\lambda_1 L_{tr} + \lambda_2 L_{tcl} + \lambda_3 L_{border})$$

Where

K represents the number of text quadrangles which are kept from IRM

L_{tr} is known as dice coefficient loss for the text region

L_{tcl} is known as dice coefficient of text center line

L_{border} are calculations made by smooth L1 loss

0.01, 0.01 and 0.1 are the values which are assigned to the weights which are λ_1 , λ_2 and λ_3 .

4.2 Text Polygon Generator

The text instances which are of arbitrary shape are reconstructed with the utilization of a strategy which is flexible, so it utilized or considered text polygon generation strategy. Following are the steps on which text polygon strategy consists of:

- Text center line sampling
- Border point generation

- Polygon scoring

N-points are sampled at an interval which is equidistant left to right on the line map which is text centered and predicted all this process happens in text center line sampling. In case of text detection which is curved the value of n is set to 7, in case of quadrangle annotations the value of n is set to 2 as per the label definition in the SCUT-CTW1500. 4 border offset map which are present in the same location provide information on which corresponding border points are determined which are based on center line points which are sampled. The polygon representation of the text is obtained when all the border points are linked clockwise. As per the text region response which is present within the polygon mean value is computed which is considered as a new confidence score.

4.2.1 Interface

Following is the loss function which is utilized to train the proposed network in an end-to – end manner.

$$L = \gamma_1 L_{dr} + \gamma_2 L_{irm} + \gamma_3 L_{sem}$$

Whereas

- L_{dr} is the loss of DR
- L_{irm} is IRM
- L_{sem} is the SEM

The weights are set to 1.0 and they are trade off which are γ_1 , γ_2 and γ_3 .

It is divided training into two stages which are

- Warming up
- Fine tuning

DR branch is trained for 10 epochs only with the utilization of data set which is synthetic. This helps DR in generating proposals which are high recall and helps in covering text instances in real data. All this process happens in the stage which is known as warming up. In the stage which is called as fine tuning it trains data sets which are available from the following competitions: ICDAR2015, ICDAR2017-RCTW, SCUT-CTW1500, Total Text,

and ICDAR2017-MLT. The proposal which is generated by DR branch is utilized by both SEM and IRM branches. To keep top K proposals, it utilized NMS which is an abbreviation of Non-Maximum Suppression. In practical scenarios 50% of the top K proposals are replaced with GT text quadrangles which are distributed randomly, since DR shows poor performance at first which affects the convergence of branches which are IRM and SEM in this case. IRM is responsible of performing refinement only once while training.

4.2.2 Inference Phase

Score map and geometry maps of quadrangle are generated by DR at first place. Preliminary proposals are generated by NMS. To perform multiple refinement shared feature maps and both proposals are feed to IRM. To be able to generate the precise text which is polygon and confidence score the quadrangles which are refined and shared feature maps are fed into the SEM. To remove the polygon which are low-confidence a threshold S is utilized, which is then set to 0.1.

4.2.3 Results achieved from the experiment

LOMO is evaluated on ICDAR2017-MLT for verify the generalized ability of LOMO for the detection of text which is a multilingual scene. On the basis of Synth model which is pre-trained fine tuning of detector is performed for 10 epochs. The longer side is set to 1536 for testing which is single-scale and the longer side scale which includes 512, 768, 1536 and 2048 are set to testing which is multi-scale. LOMO leads in performance in single scale testing as compared to other existing models. However, the borders and AFN-RPN did not indicate the type of testing scales which are utilized which can be seen in the Table 4.1;

Table 4.1 LOMO evaluation on ICDAR2017-MLT

Method	Recall	Precision	Hmean
E2E-MLT [18]	53.8	64.6	58.7
He et al. [19]	57.9	76.7	66.0

Lyu et al. [20]	56.6	83.8	66.8
	57.5	81.0	67.3
FOTS [21]	62.1	77.7	69.0
Border [22]	66.0	75.0	70.0
AF-RPN [23]	62.3	81.9	70.8
FOTS MS [21]	70.6	74.3	72.4
Lyu et al. MS [20]			
LOMO	60.6	78.8	68.5
LOMO MS	67.2	80.2	73.1

4.2.4 Analysis of the features

The model is aiming to provide solution to the problem which is there in regards to the detection of the image in the existing methods with the utilization of detection and segmentation-based methods. This experiment can help in achieving high performance on the standard bench mark and can help in maximizing the annotation which is word –level with the utilization of methods that rely on detection-based techniques. However, the method which is proposed fails to handle texts efficiently which are long. The reason of this failure is the huge variance in the text aspect ratios. This model is workable for the detection of texts which are complex and are of arbitrary shape.

4.2.5 Structural method

If we look into the structure of this method this method is introducing two modules which includes:

- IRM which is an abbreviation of iterative refinement module

- SEM which is an abbreviation of shape expression module

These two modules rely largely on DR which is a shot text detector direct regressor which has improved by adopting direct regression manner. In the case when words instances are text, a text proposal is generated by DR at the very first place and then by regressing the offset of the coordinates once or more than once to refine the quadrangle proposals close to ground truth by IRM. When it gets an input, which has a text which is irregular the geometrical attributes of the text instances are regressed by SEM and a polygon expression is reconstructed which is not just more precise but it can fit the text of arbitrary shape very well and effectively.

4.2.6 Applications which this model is offering

Following are the four applications which this model is offering which includes:

- Detection of text which is long
- Detection of text which is curved
- Detection of text which is oriented
- Detection of text which is multi-lingual

5. PMTD – PYRAMID MASK TEXT DETECTOR

5.1 Introduction

Pyramid Mask Text Detector or PMTD, a detector that encodes the information of the shape and location of each text instance in a soft text mask. For this purpose, a Plane-clustering algorithm is introduced to do this procedure. This algorithm finds the most fitting pyramid mask for the text mask. Experiments on the PMTD and its algorithm gets very effective results which are described in the evaluation section.

5.2 Architecture

The architecture for the PMTD is shown in the figure 12.

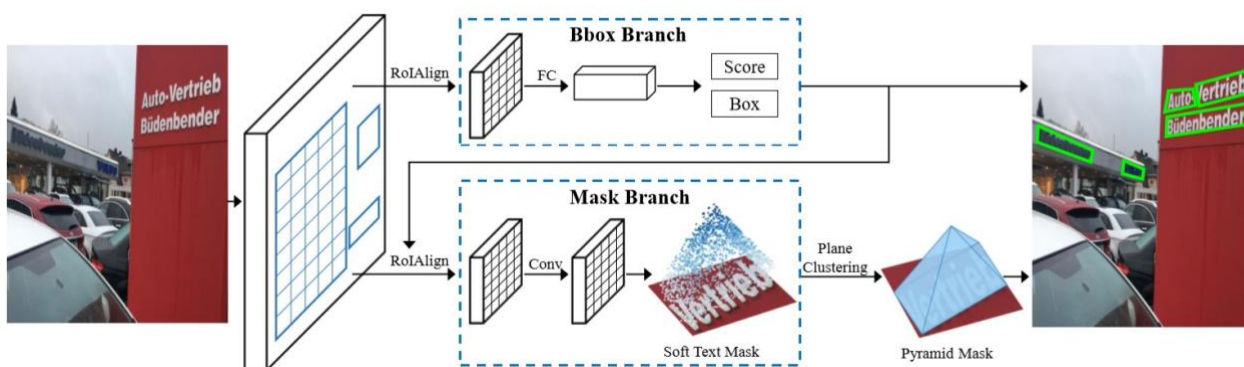


Figure 5.1. Illustration of the Pyramid Mask Text Detector architecture

5.2.1 Pyramid label

In this the mask's hard label of the class $\in \{0,1\}$ refines by it to the score $\in [0, 1]$ which is soft pyramid label this all done. PMTD can capture the given data location and the shape. Specifically, the center of text area is assigned as the apex of pyramid with the score = 1. The boundary of the text area is assigned as the bottom edge of the pyramid. This whole process uses linear interpolation to fill each side of the pyramid in the (see Figure 5.2).

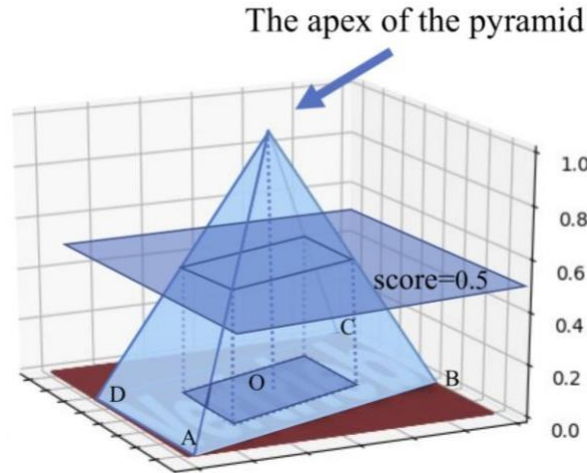


Figure 5.2. Soft pyramid generation

Formally, given the four corner points $A(x_a, y_a), B(x_b, y_b), C(x_c, y_c), D(x_d, y_d)$ of a quadrilateral, the value $score_p$ for the point p can be calculated as follows.

First, the center of text region $O(x_o, y_o)$ can be obtained by:

$$x_o = (x_a + x_b + x_c + x_d)/4 \quad (4-1)$$

$$y_o = (y_a + y_b + y_c + y_d)/4 \quad (4-2)$$

For every region R_{OMN} (region between two rays OM and ON) from $R_{OAB}, R_{OBC}, R_{OCD}, R_{ODA}$, the \vec{OP} can be decomposed uniquely:

$$\vec{OP} = \alpha \vec{OM} + \beta \vec{ON} \quad (4-3)$$

$$\begin{bmatrix} x_p - x_o \\ y_p - y_o \end{bmatrix} = \begin{bmatrix} x_m - x_o & x_n - x_o \\ y_m - y_o & y_n - y_o \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad (4-4)$$

Then, α and β can be obtained by

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} x_n - x_o & -1 \\ y_n - y_o & -1 \end{bmatrix}^{-1} \begin{bmatrix} x_p - x_o \\ y_p - y_o \end{bmatrix} \quad (4-5)$$

The region R which P belongs to needs to satisfy the following condition:

$$\alpha \geq 0 \text{ and } \beta \geq 0 \quad (4-6)$$

Then the $score_p$ can be calculated by:

$$score_p = \max(1 - (\alpha + \beta), 0) \quad (4-7)$$

When the process is in the training stage. There is one pixel presents which appears near the center of the instance, its amenable field instantly filled with another positive pixels and have a higher score accordingly. For the pixel, which appears near the center of the instance have low score near to the 0 and also contain the background context. From this, to attain more precise results in PMTD, there would be a larger receptive field is needed. After getting this information. Now, the first four convolution layers replaced with the stride 2 in the mask head for enlargement of the receptive field.

The deconvolution is harmful for the pixel wise regression because it may cause the check board patterns mentioned in [51].

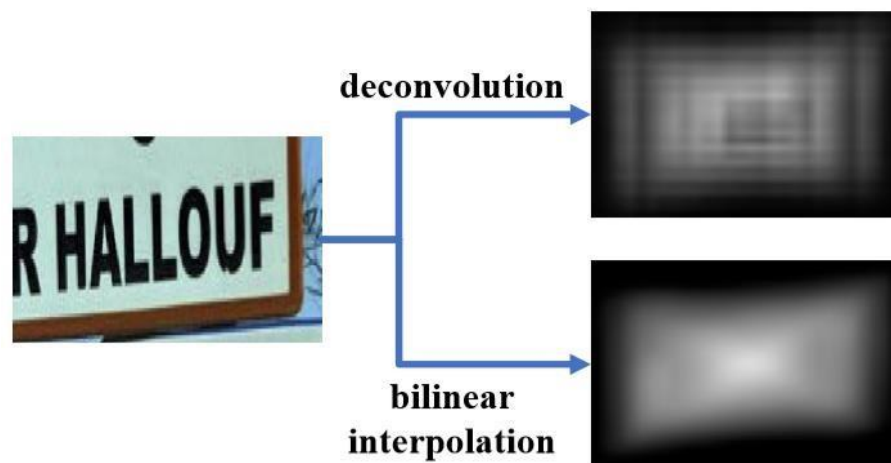


Figure 5.3. Deconvolution Causes

The check board pattern problem can be mitigated by replacing the deconvolution with bilinear interpolation (seen in Figure 5.3).

Let employs pixelwise L_1 loss to optimize the predicted text mask. Following the design in Mask R-CNN, the loss function of the whole network is as follows:

$$L = L_{rpn} + \lambda_1 L_{cls} + \lambda_2 L_{box} + \lambda_3 L_{pyramid_mask}$$

(4-8) λ_1 , λ_2 and λ_3 are set to 1, 1, 5 respectively in its experiments.

The new style label improves the pixel mislabeling problem in the training mode. For trying this, you should take the background pixel near to the boundary and see the results.

5.2.2 Plane clustering

As discussed in the Introduction of the PMTD, a plane cluster algorithm is generated for running the whole process. The plane clustering algorithm iteratively updated the fitting text box from the predicted soft text mask.

There will be reverse engineering of the pyramid label from the text area, in this reverse process, the pyramid constructs first from the text mask, and then the bottom edge of the pyramid is taking as the Output text box. In End, the most critical or sensitive part of this process is rebuilt and parameterized the location of the pyramid.

As told previously, the pyramid has four supporting planes from which one is the base plane. Let's get into the context of pyramid mask, the predicted soft mask can be converted into the point set of (x, y, z) in which the x and y denotes their location and remaining z stands for predicted score of the pixel-set. The formula for the base plan is $z=0$, and other supporting plan are equally determined by an equation $Ax + By + Cz + D = 0$, $C = 1$. Mainly, the task for plan clustering algorithm is to reduce to find the optimal parameter which are $\{A, B, D\}$ for each supporting plane.

Here is the Plane Clustering Algorithm:

Input:

$$point: location = (x, y), predicted_{score} = z$$

$$points = [point_num = H * W, (x, y, z)]$$

Output:

$$plane: Ax + By + Cz + D = 0, C = 1$$

$$planes = [plane_num = 4, (A, B, D)]$$

Function Plane Clustering(points)

$$P \leftarrow SELECTPOSITIVE(points)$$

$$apex. (x, y) \leftarrow MEAN(P). (x, y)$$

$$apex. z \leftarrow 1$$

$$planes \leftarrow INITPLANES(apex)$$

While $iter < \max_iter$ and Reject ($residuals$) do

$$G \leftarrow \emptyset \times plane_num$$

For $p \in P$ do

$$plane_i \leftarrow NearestPlane(p, planes)$$

End for

$$planes, residuals \leftarrow RLS(G)$$

End while

Return planes

End function

From starting, the set P (positive points) is build by the condition $z > 0.1$. After the starting stage, the apex of the initial pyramid is assigned as the center of P, having an ideal score $z = 1$. The predicted text bounding box, shown in the left image are the four different vertexes of the pyramid facing in the bottom which initialized as the four corner points

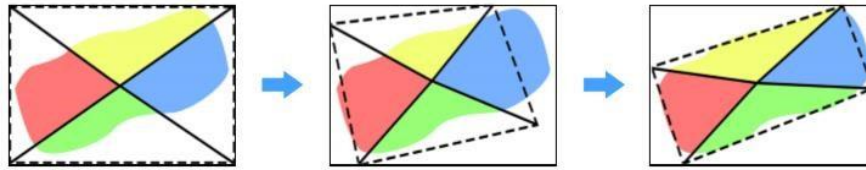


Figure 5.4. Illustration of the Plane Clustering Algorithm.

When the pyramid is initialized, the iterative updating scheme for clustering point is implemented as shown in the figure 5.4. For the assignment step, each point goes to the nearest plane and this is a repeated step. The robust least square algorithm is employed to lapse the four supporting planes from the clustered plane algorithm points respectively, which leads to the noise in the predicted text mask.

The Final square angle pyramid is obtained after the iteration reaches to the maximum iteration or when the RLS is small enough by returned through regression residual. After the final square obtained the text box can be easily calculated out by the 4 supporting planes and the plane $z=0$. The maximum iteration and the residual threshold is assigned to 10 and $1e-4$ respectively in this experiment. The plane clustering algorithm takes the advantage for the whole soft mask's information for most fitting pyramid.

5.3 Datasets

ICDAR 2017 MLT It is a multi-oriented, multi-scripting, and multi-lingual scene text dataset. It consists of 7200 training images, 1800 validation images, and 9000 test images. Four vertices of the quadrilateral annotate the text regions in this. This dataset is one of the difficult and larger dataset for image detection.

ICDAR 2015 is another multi-oriented text detection dataset only for English, which includes 1000 training images and 500 testing images. Similar to ICDAR 2017 MLT.

6. IMPLEMENTED MODEL - MASK RCNN

The core purpose of this method which is Mask RCNN is to offer a framework which can enable instance segmentation for the systems which have powerful baseline which includes Fast R-CNN and FCN (which is a fully convolutional network). The systems which have powerful baseline offer so many advantages which includes intuitive clarity to the concept, flexibility, robustness, fast time of training and inference.

6.1 Problems which need to be resolved

Significant tasks which are related to computer vision are as follows;

- Classification
- Detection of object
- Segmentation of image

The predecessors to the Mask-RCNN model, systems like FCN and FCNN were capable of detecting object. The model of Mask RCNN is designed to perform an important task which is segmentation of an image. For systems including Faster RCNN, mask branch, FPN and RoI align the Mask RCNN is approximately same no visible difference is seen. When a regional proposal network is added to fast R CNN faster R-CNN forms. Faster RCNN= RPN + Fast- RCNN. The mentioned models' architectures are discussed in more detail below.

6.2 Additions made by MASK R-CNN method

Following are the additions which are made by Mask R-CNN.

- In this method which is Mask RCNN to the classification and location branch of fast R-CNN a new mask branch is added.
- FPN which is an abbreviation of feature pyramid network is added before RPN.
- For the masking of image segmentation RoI pooling is replaced by RoI Align.

6.3 Architecture of MASK R-CNN model

The conceptual architecture of Mask R-CNN is simple and mainly consists of;

- Faster R-CNN which has two outputs for each object which is candidate
- Class label
- Bounding-box offset
- An addition of a third branch that outputs the mask of the object

Mask RCNN is not just an intuitive but also a natural idea. Spatial layout of an object is required for the additional third branch mask, which is very much different from the box output and class. The important missing part of Faster R-CNN is pixel-to-pixel alignment that is achieved by this model that is Mask R-CNN. This is accomplished by adding an additional neural network capable of semantic segmentation which is classification of each individual pixel in the inputted image.

6.4 FASTER R-CNN

In Mask R-CNN model, firstly detector of Faster R-CNN is reviewed. The faster R-CNN constitutes of two stages majorly which are:

- The first stage of the architecture consists of a Region Proposal Network. The object bounding boxes are proposed to candidates by the RPN. Which constructs a number of proposed regions of interest (RoI) on the inputted image. The RoI network is trained to place boxes with higher frequency around detected objects in the image.
- The classification of performance, regression of bounding box and extraction of features out of every candidate box with the utilization of Rolpoll are performed in the second stage which is an essence of fast R-CNN.

For faster inference both the features which are performed in the above mentioned two stages can be shared. A detailed comparison between other frameworks and faster R-CNN is offered here.

6.5 MASK R-CNN

The Mask R-CNN has adopted a procedure which is majorly based on two stages;

- The first stage is RPN
- It not just predicts the class and box offset it also displays a binary mask for each RoI as an output.

Mask R-CNN model is very much unlike to other systems. In Mask R-CNN the classification majorly depends upon the predictions of the mask of the objects. In this approach the Fast R-CNN architecture is followed for the classification of bounding-box and for the performance of regression which simplifies the pipeline of R-CNN which is multistage.

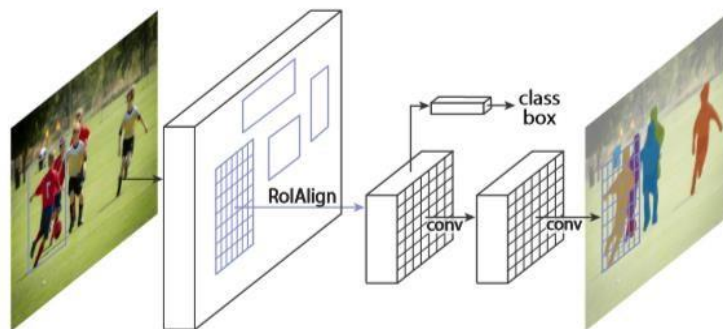


Figure 6.1 The illustration is showing the framework which is achieved in the Mask R-CNN model for instance segmentation

On each sampled RoI the multi task loss is defined as follows during training;

$$L = L_{cls} + L_{box} + L_{mask}$$

The K binary masks which have resolution of $m \times m$ for each class which are included in K are encoded by the mask branch which has a Km^2 dimension of the output for each RoI. To the encoded mask branch a sigmoid is applied per pixel and L mask is defined as the average cross entropy loss which is binary. The RoI which is associated with K which is a Ground truth class, Lmask is defined on the k-th mask only. As per the definition of Lmask it allows network for the generation of mask for each and every class without initiating any competition within the class. Despite, it depends largely on the classification class which is dedicated and it predicts the labels of class which are then used for the selection of output mask, further which helps in decoupling the mask and prediction of class. Conventionally, per-pixel softmax and a cross-entropy loss which is multinomial are used when FCN's are

applied to semantic segmentation, the result is the completion of masks across the classes. In this model per-pixel sigmoid and binary loss are utilized which do not result in completion of masks across the classes.

6.6 Mask representation

Fully connected layers result in the collapse of box offset in the form of vectors that are short outputs. The spatial structure that is extracted by the mask can be addressed with the utilization of correspondence that is pixel-to-pixel (offered by convolution) naturally. For each RoI the mask of $m \times m$ order is predicted with the utilization of FCN. This prediction allows each layer that is present in the mask branch for the maintenance of layout of an object that is spatial and explicit. Thus, preventing it from collapsing into vector representation that lacks in the dimensions that are spatial. The representation in this model is fully convolutional which is more accurate in comparison to the previous models. In addition, it requires a few parameters. In order to preserve correspondence that is per-pixel spatial and explicit there is a requirement of alignment between the pixel-to-pixel behavior and RoI features. The main motivation behind this was the development of RoI-align layer that plays a significant role in the mask prediction.

6.7 RoIAlign

From each RoI the small feature maps are extracted with the utilization of standard operation that is RoIPool. A floating number RoI was quantized to the granularity of the feature map that is discrete. The subdivision of quantized RoI was done into the bins that were spatial which were quantized themselves and then finally the feature's values that were covered by each bin were aggregated.

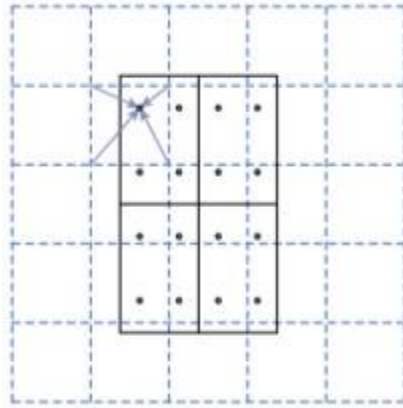


Figure 6.2 RoIAlign method

By computing of the continuous coordinate which is x quantization is achieved whereas feature map stride is 16. A disturbance in the alignment observed between the RoI and extracted features when quantization was achieved. The disturbance of alignment is seen to be leaving negative impacts on the predictions which were made regarding the pixel-accurate masks. However, no impacts were observed on the classification since it is robust for small translation. For proper alignment of the features which are extracted with the input and for the removal of harsh quantization of RoIPool, RoI align layer is proposed in this model. This proposed solution works by simply avoiding the quantization of RoI boundaries. For the computation of exact values of the input features at the sampled locations which are four and regular bilinear interpolation is utilized. As soon as no quantization is performed the results which are achieved are not sensitive to the location. Large improvements are made in the RoIAlign. By overlooking the issues which might come across regarding the alignment RoI Wrap was utilized and implemented. The table is showing the role of alignment which is crucial.

Table 6.1 The role of alignment

	Align	Bilinear	agg	AP	AP50	AP75
RoIPool [12]			Max max	26.9	48.8	26.4

RoIWarp [10]		✓	Max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	Max	30.2	51.0	31.8
	✓	✓	ave	30.3	51.2	31.5

6.8 Architecture of the MASK R-CNN network

Following approaches are utilized and numerous architectures were instantiated with Mask R-CNN.

- For the extraction of features over the whole image convolutional backbone architecture was utilized
- For classification, regression and prediction of the mask individually for each RoI network head is utilized. With the utilization of network-depth feature which is nomenclature this architecture is playing the role of a backbone.

ResNet [33] and ResNeXT [34] networks with a depth of 50 or 101 layers are evaluated. In table 6.1 the fourth stage of Convolutional layer are utilized for the extraction of features and implementation of Faster R-CNN and ResNet is done. FPN which is a feature pyramid network also proposed as a backbone network. for the construction of feature pyramid within a network with the utilization of an input which is single-scale top-down architecture with a lateral connection is utilized by FPN. RoI Feature from different pyramid as per the scale are extracted by faster R-CNN with FPN. Rest of the approach is very much similar to the vanilla ResNet. Mask R-CNN with the utilization of ResNet FPN is showing efficiency in terms accuracy and speed.

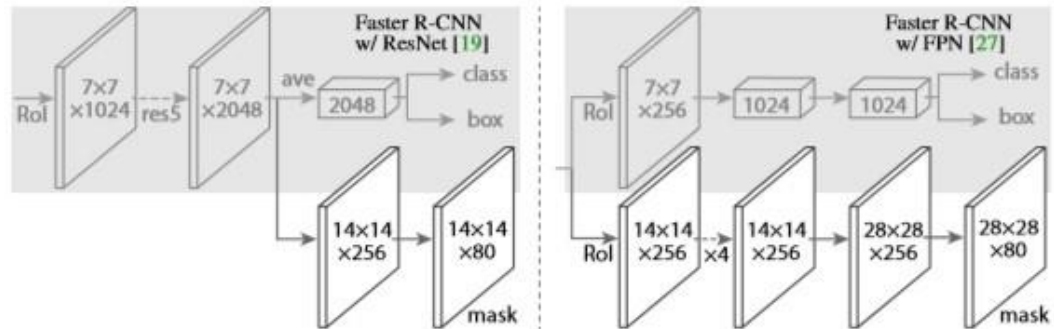


Figure 6.3 Mask R-CNN with the utilization of ResNet FPN

The 5-th stage of ResNet with intensive computing is utilized on the head of ResNet C4. The backbone in FPN utilized res5 which further allowed the head which are efficient and utilized limited filters. In this architecture the mask branches own structure which is straight forward. There is a room of future improvement if utilize design which is complex.

6.9 Architecture of the head

The heads of faster R-CNN which were existent were extended. The head of ResNet and FPN backbones can be seen on the Right/Left panels. Spatial resolution and channels are denoted by numbers. Conv is denoted by arrows.

6.10 MASK R-CNN model architecture convolutional network based on masked region

The model is classified into the following segments;

- Instance segmentation
- Object detection
- Semantic segmentation

The Mask RCNN inbuilt on two parts majorly which include object detection and semantic segmentation. The object detection relies on Fast R-CNN and semantic segmentation relies largely on fully convolutional neural network. Mask R-CNN utilizes an architecture which is very much similar to Faster R-CNN. There are two phases in the faster R-CNN the first phase out of two phases is known as RPN which is an abbreviation of region proposal network. It is an extension of Faster R-CNN.

7. IMPLEMENTATION

In the context of MLT 2019 competition there are implemented two tasks out of four. The tasks which are implemented in this thesis include;

- Text detection
- Script identification

7.1 MASK R-CNN model implementation

It is utilized convolutional Neural Network Architecture Mask R-CNN for the detection of the object and segmentation of scripts which are present in the images. It is also utilized a modified version of open source implementation of Mask R-CNN.. The test training data set is there to figure out the format of the input of training data so that it can be used to train on external datasets.

The following model sample was used/modified for the script model.

Mask _ RCNN/ sample/balloon/ballon.py.

It is committed the structure of the given JSON-file containing the annotation boundaries for the training data, to be able to reconstruct the given txt-label files into a similar structure to be able to train the Mask R-CNN model. The structure was reconstructed and saved as each boundary box x- and y-points for each script into a CSV-file. The file which generates the CSV-file from the txt-files is Jupyternotebook_MaskRCNN-fix_labels.ipynb.

Table 7.1 shows the structure of the CSV-file containing the annotation boxes for the scripts.

filename	class name	data
tr_img_00	2	{'name': 'polyline', 'all_points_x': [33, 106, 91, 19], 'all_points_y': [1097, 1083, 1144, 1142]}
tr_img_00	1	{'name': 'polyline', 'all_points_x': [743, 1082, 1078, 737], 'all_points_y': [894, 850, 896, 932]}
tr_img_00	2	{'name': 'polyline', 'all_points_x': [252, 470, 466, 257], 'all_points_y': [579, 545, 586, 620]}
tr_img_00	2	{'name': 'polyline', 'all_points_x': [495, 705, 701, 487], 'all_points_y': [537, 506, 539, 576]}
tr_img_00	2	{'name': 'polyline', 'all_points_x': [408, 773, 763, 393], 'all_points_y': [419, 380, 449, 515]}

The structure as can be seen in Table 7.1 was derived by investigating the JSON-format from the demo training data files.

The file `balloon.py` is modified in a way so that it can train 10 classes with the utilization of label data which is being read from the CSV-file.

```
class BalloonDataset(utils.Dataset):

    def load_balloon(self, dataset_dir, subset):
        self.filenamees=[]
        """Load a subset of the Balloon dataset.
        dataset_dir: Root directory of the dataset.
        subset: Subset to load: train or val
        """

        # Add classes. We have only one class to add.
        self.add_class("root", 1, "1")
        self.add_class("root", 2, "2")
        self.add_class("root", 3, "3")
        self.add_class("root", 4, "4")
        self.add_class("root", 5, "5")
        self.add_class("root", 6, "6")
        self.add_class("root", 7, "7")
        self.add_class("root", 8, "8")
        self.add_class("root", 9, "9")
        self.add_class("root", 10, "10")
```

Figure 7.1 illustrates the part of the code where all of the script classes are added.

Classes from 1 to 10 are added which is corresponding to the to a specific script which is present in the training data. The ID number 1-10 is later translated back into the specific script name (e.g Latin, Chinese, Arabic) in the code responsible for predicting on future images.

```

# We mostly care about the x and y coordinates of each region
if subset=="train":
    #List of train-filenames with annotations
    train_annotations=['data_train.csv']
    for file_name in train_annotations:
        self.annotations = pd.read_csv(os.path.join(dataset_dir, file_name))
        filenames=self.annotations.filename.unique()
        #self.annotations = list(annotations.values())
        for file_x in filenames:
            self.classID=[]
            self.polygons=[]
            self.filenames.append(file_x[:-4]+".jpg")
            list_x=self.annotations.loc[self.annotations['filename']==file_x]
            for index, row in list_x.iterrows():
                #self.polygons.append(row['data'])
                self.classID.append(str(row['class name']))
                dict1= eval(row['data'])
                self.polygons.append(dict1)
            image_path = os.path.join(dataset_dir, (file_x[:-4]+".jpg"))
            image = skimage.io.imread(image_path)
            height, width = image.shape[:2]
            self.add_image(
                "root",
                image_id=(file_x[:-4]+".jpg"), # use file name as a unique image id
                path=image_path,
                width=width, height=height,
                polygons=self.polygons,
                # I have added the classID variable
                #> Have Problem here <
                classID=self.classID
            )

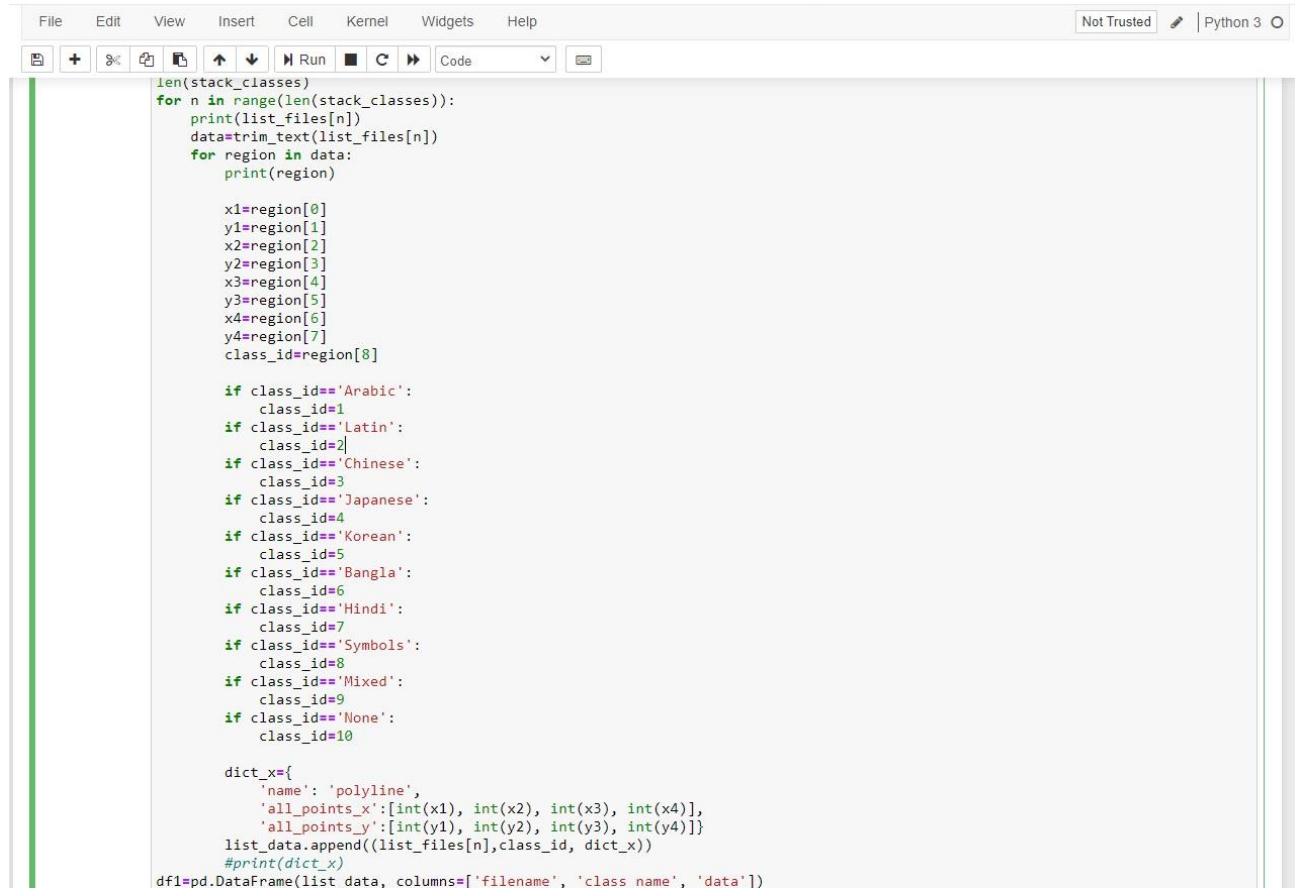
```

Figure 7.2 illustrates the part of the code which had to be modified to be able to take CSV-files as input instead of the original JSON-file.

The file which is responsible for loading the images and the labels of the training and validation data had to be modified to be able to input the CSV-file with the annotation of the script boundaries.

7.1.1 Engineering activities

First of all, the dataset was split into a training set and a validation set. 70% of the dataset was placed into the training set, and the rest 30% was placed on the validation set.



```

len(stack_classes)
for n in range(len(stack_classes)):
    print(list_files[n])
    data=trim_text(list_files[n])
    for region in data:
        print(region)

        x1=region[0]
        y1=region[1]
        x2=region[2]
        y2=region[3]
        x3=region[4]
        y3=region[5]
        x4=region[6]
        y4=region[7]
        class_id=region[8]

        if class_id=='Arabic':
            class_id=1
        if class_id=='Latin':
            class_id=2
        if class_id=='Chinese':
            class_id=3
        if class_id=='Japanese':
            class_id=4
        if class_id=='Korean':
            class_id=5
        if class_id=='Bangla':
            class_id=6
        if class_id=='Hindi':
            class_id=7
        if class_id=='Symbols':
            class_id=8
        if class_id=='Mixed':
            class_id=9
        if class_id=='None':
            class_id=10

        dict_x={
            'name': 'polyline',
            'all_points_x':[int(x1), int(x2), int(x3), int(x4)],
            'all_points_y':[int(y1), int(y2), int(y3), int(y4)]
        }
        list_data.append((list_files[n],class_id, dict_x))
        #print(dict_x)
df1=pd.DataFrame(list_data, columns=['filename', 'class name', 'data'])

```

Figure 7.3 Illustrates the how the txt-labels are transformed into the correct structure.

The output from the file seen in Figure 7.3 can be seen in Table X above. The output CSV-file is made to match the previously mentioned JSON-file containing the demo training dataset available at Github with the original model code.

Modified the rootX1.py file to be able to take in the CSV-label file to train the model. It was changed NUM_CLASSES to 11 (all the classes plus 1). This image shows how the model takes in CSV-file and stores it for the model to train.

The modified code goes through every image which is present in the training data set. Also, it thereon generates a bitmap for every instance of scripts. A bitmap is the annotation form which the model uses to train for semantic segmentation. The function which loads the data generates images where pixels are white (pixel value=255) for the background and black (pixel value=0) for the specific class. For every matching file name, a bounding

box is added around the word as a label in every row. The file containing modified training model along with other files which are from the GitHub model are uploaded to the google drive. A connection was built between Google Calobratory and Google Drive to make sure the files are received from the Google Drive. The creators of the Github Mask-RCNN code added argpraiser-python commands to be able to run the file from the terminal window. To run the code in Google Colaboratory it is needed to add an "!" in front of the code block to emulate a terminal. Then it is needed to add the address to the folders containing the pretrained weights, training and validation data, and where to save the logs/new weights.

! python root X1.py train --dataset = ../train images/ --weights = / weights / ... --logs = /folder'. This line is a command line which is mentioned in the argparser. In the python agrparser makes the running of the python file along with inputs easy in the terminals. Deep learning model are trained for 24 hours until the environment closes by Google Colaboratory along with an integrated GPU. Upon that the google colab had to be restarted and continued from the old weights.

'--dataset' path is taken to the training data as an input.

'--- weights' path is taken to the pre trained weights as an input.

Weight trained on the COCO dataset were used as starting weights when training the Mask-RCNN model. 330 thousand images along with labels around the object are presented in the COCO data set. The pretrained weights are used to decrease the amount of data which was required to train the important images which needs to be learn from an image utilized COCO data set. The later layers in the network are trained and this technique is known as transfer learning.

To get optimal results for the given dataset it is tested 60 hyperparameter combinations in order to yield the lowest validation loss possible. But given more time more hyperparameter combinations can be tested to ensure the best results.

7.1.2 Tested hyper parameters

Following are the hyper parameters which are tested to get the optimal(lowest) loss value.

RPN_NMS_THRESHOLD: This variable stands for Non Maximum Suppression for the region proposal network, this value is used to pick the optimal region proposal rectangles which overlaps the objects in the best way. The NRM threshold takes in a list of proposal rectangles and outputs a lower amount of proposal rectangles after the threshold.

RPN_TRAIN_ANCHORS_PER_IMAGE: This variable determines how many anchors that should be present per image. Each anchor is a specific rectangle of different aspect ratio, these are then later used when generating the regions of interest RoI.

DETECTION_MIN_CONFIDENCE: This value determines the threshold of the predictions which counts as true or not. A low value indicates that more predictions will be counted as true and a high value indicates that the predictions have to have high accuracy in order to classify an object as a specific class.

TRAIN_ROIS_PER_IMAGE_NMS_THRESHOLD: This value determines how many regions of interest should be generated per image,

DETECTION_NMS_THRESHOLD: This variable stands for the Non-maximum Suppression of the final object detection stage of the Mask RCNN network. The NRM threshold takes in a list of proposal rectangles and outputs a lower amount of proposal rectangles after the threshold.

MAX_GT_INSTANCES: This value determines the threshold for the maximum number of objects that can be found in an image. If the value is too high it might induce false positive objects detected-

ROI_POSITIVE_RATIO: This value determines how much overlap in the IoU metrics which is calculated as a positive/negative classification. If ROI_Positive_Ratio is 0.5 then all ROI

LEARNING_RATE: Determines how big changes/steps the neural network makes to the weights in order to get towards the optimal loss value which yield the best prediction accuracy.

LEARNING_MOMENTUM: Determines how fast the learning rate momentum is accumulated. Higher momentum means that the models learning rate momentum increases faster.

WEIGHT_DECAY: Is a regularization technique which adds a small loss to the weights

```
#Make grid of hyperparameters to test
mask_rcnn_hyperparameters2={ 'RPN_NMS_THRESHOLD': [[0.7, 0.8]],
                              'RPN_TRAIN_ANCHORS_PER_IMAGE': [256, 512],
                              'DETECTION_MIN_CONFIDENCE': [0.6,0.7,0.8],
                              'TRAIN_ROIS_PER_IMAGE': [200, 250],
                              'DETECTION_NMS_THRESHOLD': [0.3,0.4],
                              'MAX_GT_INSTANCES': [100, 150],
                              'ROI_POSITIVE_RATIO': [0.25, 0.33, 0.45],
                              'LEARNING_RATE': [0.001,0.005,0.01,0.02],
                              'LEARNING_MOMENTUM': [0.8,0.9],
                              'WEIGHT_DECAY': [0.0001,0.0005]}
```

Figure 7.4 displaying the hyperparameter combinations which were generated.

The hyperparameter interval of values were chosen around the standard hyperparameter values which were included in the code from Github. So, for example if the standard value for RPN-NMS-Threshold was at 0.75, it was chosen the interval to be from 0.7 to 0.8. All of these values determine how well the model will be able to detect the objects which it is trained on.

Out of 4608 combinations the testing of model is done with randomly picked hyper parameters, based on the range of hyperparameters seen in the table seen in Figure 7.4. Out of the 60 tested hyperparameters

7.1.3 Prediction/ object detection and segmentation

Figure 7.5 is showing the code from the file which is the Google Colab notebook detection.ipynb. The Figure 7.5 is the class which is generated separately in Google Colaboratory which was dedicated to make predictions on the images. It makes predictions on the input images by loading the previously saved weights from the training. The visualization from the predictions is printed out by the model along with boundary box, segmentation mask, prediction accuracy and class.

```

# Visualize results
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                           self.class_names, r['scores'])

#To export the data of every detected text
n=0
self.ID=[]
self.lengths=[]
self.list_of_regions=[]
self.accuracy=[]

for ID in (r['class_ids']):
    (y1,x1,y2,x2)=(r['rois'][n])
    accuracy1=r['scores'][n]
    self.accuracy.append([accuracy1])
    length2=(y2-y1)**2+(x2-x1)**2
    length=math.sqrt(length2)
    length=int(length)
    self.ID.append(ID)
    self.lengths.append(length)
    self.list_of_regions.append((x1,y1,x2,y2))

    n=n+1
self.all_accuracy.append(self.accuracy)
self.detected_region.append(self.list_of_regions)
self.filename.append(name)
self.detected_class.append(self.ID)

```

Figure 7.5 Displaying the code section which generates the wanted result format for the competition.

```

▶ #Test model on images in IMGAGE_DIR
test_dir='/content/gdrive/My Drive/mask_test'
os.chdir(test_dir)
lista_im=os.listdir(test_dir)
print(len(lista_im))
c1=detect(lista_im)
c1.image()

```

Processing 1 images					
image	shape: (1280, 1706, 3)	min: 0.00000	max: 255.00000	uint8	
molded_images	shape: (1, 2048, 2048, 3)	min: -123.70000	max: 151.10000	float64	
image metas	shape: (1, 23)	min: 0.00000	max: 2048.00000	int64	
anchors	shape: (1, 1047552, 4)	min: -0.17686	max: 1.14560	float32	
ts_img_09079.jpg					
Processing 1 images					
image	shape: (787, 1024, 3)	min: 0.00000	max: 255.00000	uint8	
molded_images	shape: (1, 2048, 2048, 3)	min: -123.70000	max: 150.10000	float64	
image metas	shape: (1, 23)	min: 0.00000	max: 2048.00000	float64	
anchors	shape: (1, 1047552, 4)	min: -0.17686	max: 1.14560	float32	
ts_img_09084.jpg					
Processing 1 images					
image	shape: (414, 870, 3)	min: 0.00000	max: 255.00000	uint8	
molded_images	shape: (1, 2048, 2048, 3)	min: -123.70000	max: 151.10000	float64	
image metas	shape: (1, 23)	min: 0.00000	max: 2048.00000	float64	
anchors	shape: (1, 1047552, 4)	min: -0.17686	max: 1.14560	float32	
ts_img_09072.jpg					

Figure 7.6 shows the function which takes as input the folder with the images to predict.

In Figure 7.6 the input folder to the function detect contains 10000 images which are from the competition and are present in the google drive folder model prediction was ran. In the above picture the model was ran to save the results from the competition predictions are not visualized, as that slows down the prediction process of the 10000 images. This function does not save the prediction into the correct result structure but passes it on to the function 'save_txt' seen in Figure 7.7


```

def save_txt(filenamees, points, accuracy):
    list_strings=[]
    for n in range(len(filenamees)):
        string=''
        acc=accuracy[n]

        for m in range(len(points[n])):
            x1=points[n][m][0]
            y1=points[n][m][1]
            x2=points[n][m][2]
            y2=points[n][m][1]
            x3=points[n][m][0]
            y3=points[n][m][3]
            x4=points[n][m][2]
            y4=points[n][m][3]
            acc_x=round(acc[m],3)

            temp_string=f'{{x1}},{{y1}},{{x2}},{{y2}},{{x3}},{{y3}},{{x4}},{{y4}},{{acc_x}}\n'
            string +=temp_string
        list_strings.append(string)
    with open(f'/content/gdrive/My Drive/mask_results/res_{{filenamees[n][:-4]}}.txt', "w") as text_file:
        print(string, file=text_file)
    return list_strings
an=save_txt([c1.filename,c1.detected_region,c1.all_accuracy])

```

Figure 7.7 illustrates the function which modifies the result data into the correct one for the competition.

The function 'save_txt' takes as input the filename of the image, the points and the accuracy which is yielded from the function seen in Figure 7.5 and Figure 7.6 (calling on function).

As the predictions were made on all the images by the model the prediction data regarding every image was saved in the model which was required in the competition. The structure of the result file was supposed to look in the following way: X1, y1, x2, y2, x3, y3, x4, y4, confidence

```

res_ts_img_07991 - Notepad
File Edit Format View Help
1861,877,2045,877,1861,923,2045,923,0.9409999847412109
631,498,750,498,631,1097,750,1097,0.8980000019073486
1928,1272,2014,1272,1928,1354,2014,1354,0.8029999732971191
529,324,829,324,529,393,829,393,0.7820000052452087
1925,824,2035,824,1925,1444,2035,1444,0.7770000100135803

```

Figure 7.8 illustrates the format in which the competition required the results to be in.

7.1.4 Head architecture

The heads of faster RCNN are extended. The heads of ResNet C4 and FPN backbones are shown on the left/right panel. Arrows are there to show if convolutional, deconvolutional or fully connected layers are inferred from the context or not. The convolutions in this model are of order 3×3 except the output one which is 1×1 . The deconvolutions are of order 2×2 with a stride which is 2. ReLU [40] is utilized in the hidden layer. The fifth stage of ResNet is denoted by Left: 'res5'. To operate the first conv at Rol of 7×7 with the stride 1 the ResNet is altered. A stack of four convolutions which are consecutive this term is used Right: '*4'.

7.1.5 Data set

Two data sets are utilized which include:

COCO data set

ICDAR – MLT – 2019 dataset

Fine annotation is done to train images which are 2975, 500 val and 1525 test images. A number of 20k coarse images are present for training without instance annotation. The pixel of all images is 2048×1024 .

7.1.6 Training

Training Loss

The different colors which can be seen in the picture below it is due to Google Colabortory Pro which runs approximately to 24 hours until the environmental closes, the different colors represent different runs. I had to utilize weights that were saved previously, and continue the next training from there. The training was restarted multiple times. As I did not choose the last weight overlapping happened which can be seen in the graph. The weights which have low validation loss were chosen and then I continued to train them. To be able to observe the general trends I smoothed the chart in the Tensorflow Graph package.

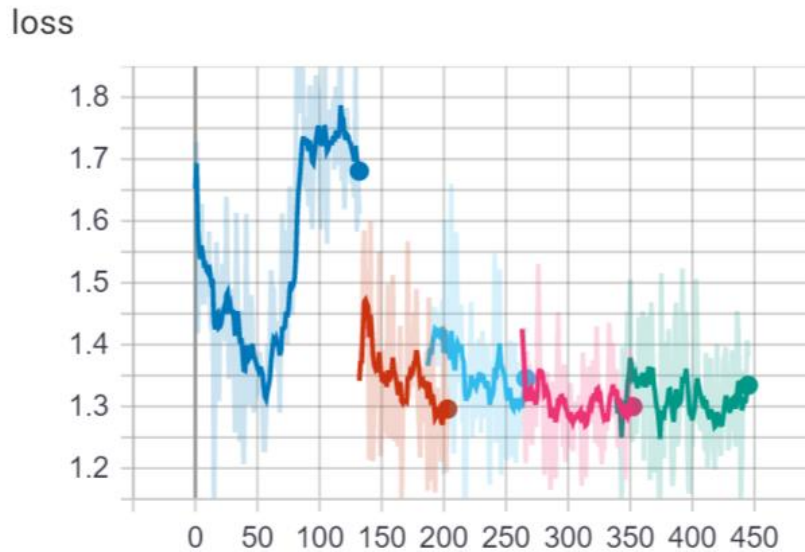


Figure 7.9 illustrating the training data loss. The darker colored lines represent the smooth/average loss and the lighter colored lines represent the actual value at each epoch.

In regards to the validation loss the general trend which is observed is that around epoch 350 the results got better. After epoch 350 the smoothed loss function seemed to be increased. If we see at the real value, the validation loss seemed to be as low at 350 epochs as the validation loss is at 300 epochs.

val_loss

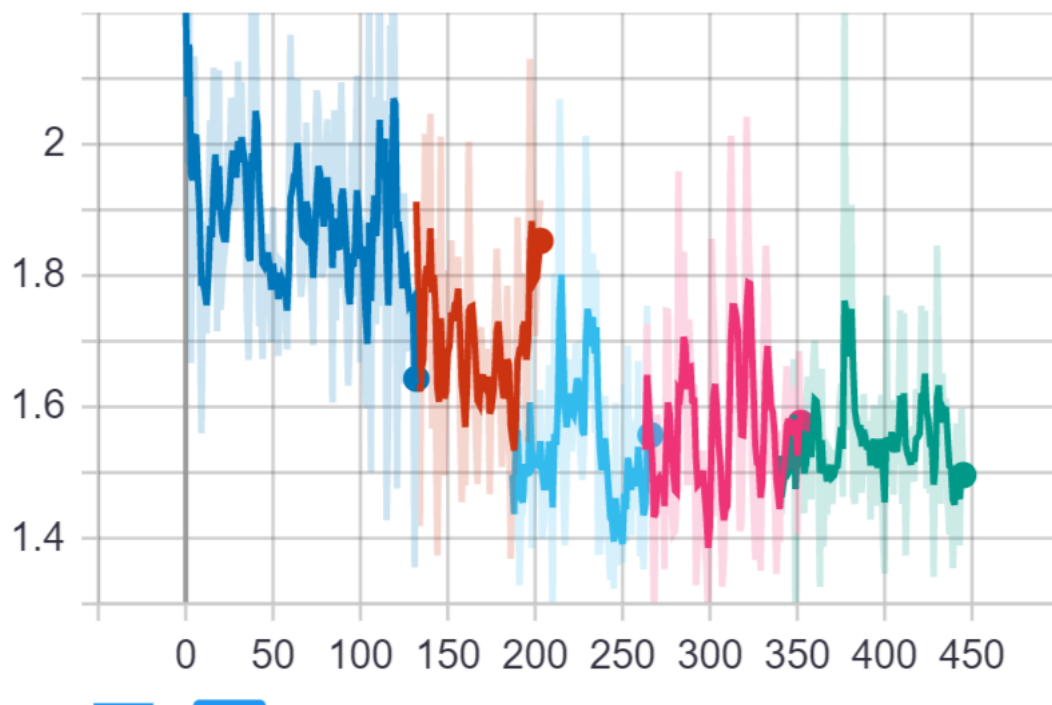


Figure 7.10 illustrates the loss validation loss function.

The validation loss is combined as is mentioned in the training loss section. Then choosing the weight to continue the training I choose the weights with the lowest validation loss, as that is more important than the training loss. The validation loss is a better representation of how well the model performs on external data, thus I choose the weight with the lowest validation loss to continue the training from.

- Rpn_class_loss = RPN classifier anchor loss: This loss checks for correct and incorrect classification generated by the Region proposal Network anchor boxes.
- Rpn_bbox_loss = RPN bounding box loss graph: This loss explains the accuracy of the placement location of the generated boxes.
- Mrcnn_class_loss = loss for the classifier head of mask R-CNN: This loss refers to the classification of a specific object inside a generated region proposal.

- `Mrcnn_bbox_loss` = loss for Mask R-CNN bounding box refinement: This loss value refers to the accuracy of the generated bounding box of the specific 'script' in this projects case.
- `Mrcnn_mask_loss` = This loss value indicates the accuracy of the generated mask overlapping the identified object.

Loss: The loss value of the model is a combination of all of the loss values above. Meaning that the model with the lowest loss has the lowest loss in all of the losses seen above.

8. EVALUATION

Experimental Outcome:

Instance Segmentation:

The Mask R-CNN was compared with the utilization of state-of-art methods in the instance segmentation. All the instantiations which are involved in this model outperform variants which were involved in state of art models. The Mask R-CNN with a ResNet-101-FPN backbone outperforms FCIS +++. The FCIS +++ contains multi-scale train test, horizontal flip test, and OHEM which is an abbreviation of online hard example mining. However, there are improvements required.

Table 8.1 Mask R-CNN compared with the utilization of state-of-art methods

	backbone	AP	AP_{50}	A	P_{75}	AP_S	AP_M	AP_L
MNC	ResNet-101-C4	24.6	44.3		24.8	4.7	25.9	43.6
FCIS [26] + OHEM	ResNet-101-C5dilated	29.2	49.5		-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5dilated	33.6	54.5		-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9		34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0		37.8	15.5	38.1	52.4
	ResNeXt-	37.1	60.0		39.4	16.9	39.9	53.5

	101-FPN						
--	---------	--	--	--	--	--	--



Figure 8.1 Mask R-CNN Results

The mask R-CNN shows good results even under the conditions which were not favorable. On instances such as overlapping systematic artifacts are displayed by FCIS +++. Mask R-CNN displayed no artifacts like the one which are displayed by FCIS +++.

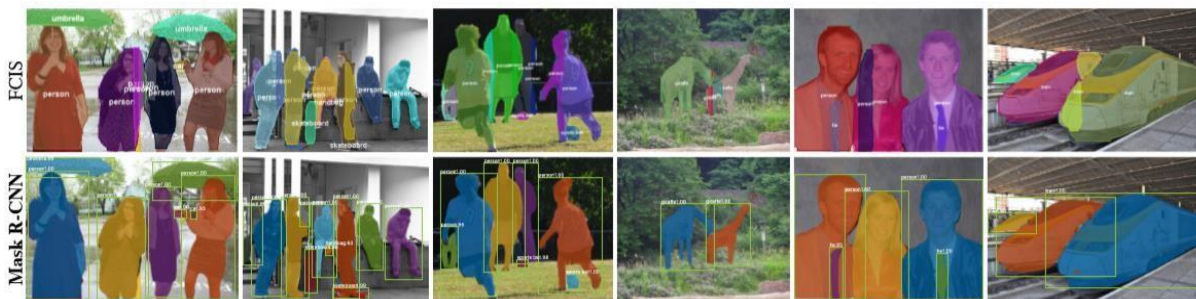


Figure 8.2 Mask R-CNN displayed no artifacts like the one which are displayed by FCIS +++

8.1 Ablation experiment

For the analyzing of Mask R-CNN number of ablations are run. The results are found by attaching Mask R-CNN with multiple backbones. Deeper networks (50 vs 101) and advanced designs which includes FPN and RexNeXT offer benefits to Mask R-CNN with numerous backbones. It is to be noted that all deeper and advanced networks do not provide benefits to all frameworks automatically.

Table 8.2 Mask R-CNN results are found by attaching Mask R-CNN with multiple backbones

Net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-50-FPN	32.7	54.2	34.3
ResNet-101-C4	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	36.7	59.5	38.9

8.1.1 Multinomial vs independent mask

Without initiating any competition within the classes, it generates a mask for each class as soon as the class labels are predicted by the existing box branch. Softmax per-pixel and multinomial loss are utilized for comparison. The tasks which are related to mask and class prediction are coupled by this alternative which results in the form of a severe loss in the AP mask. It is enough to predict a binary mask without considering categories as the classification of instance is performed at once as it makes training of the model easier.

Table 8.3 Softmax and Sigmoid comparison

	AP	AP_{50}	AP_{75}
Softmax	24.8	44.1	25.1
Sigmoid	30.3	51.2	31.5
	+5.5	+7.1	+6.4

8.1.2 Class specific vs class-agnostic masks

29.7 mask AP vs 30.3 for the counter parts which are specific to the class on ResNet-50-C4 this is the effectiveness of Mask R-CNN with masks which are agnostic to class. By default, it instantly predicts masks which are class specific one M*M mask per class. The division of labor which is utilized in our approach results in decoupling the classification and segmentation.

RoI-Align:

We have utilized ResNet-50-C4 as a backbone which has a stride 16. The AP is improved by almost 3 points over the RoIPool which offers gain at high IoU (AP_{75}). There is insensitivity to the average pool in the RoIAlign. It also compares RoIWrap which is proposed in MNC which adopts bilinear sampling too. As the RoIWarp quantizes the RoI alignment with the input gets loser. RoIWrap performs worse with RoIPool as compared to

RoIAlign. Thus, the point which is to be considered is that proper alignment is the key. It is also capable of evaluating RoIAlign with a backbone which is ResNet-50-C5 and it has a stride which is larger and is of 32 pixels. As it can be seen in the table which is below that mask AP is improved by points which are massive that is 7.3 by the RoIAlign.

Table 8.4 RoIPool and RoIAlign comparison

	AP	AP_{50}	AP_{75}	AP_{bb}	AP_{bb50}	AP_{bb75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	30.9	51.8	32.1	34.0	55.3	36.4
	+7.3	+5.3	+10.5	+5.8	+2.6	+9.5

When utilized with FPN with strides which are fine and multilevel a gain of 1.5 mask AP and 0.5 AP box is shown by RoIAlign. RoIAlign show larger gain even with the FPN in case of key point detection where fine alignment is required.

Table 8.5 RoIPool and RoIAlign comparison with FPN

	AP_{kp}	AP_{kp50}	AP_{kp75}	AP_{kpM}	AP_{kpL}
RoIPool	59.8	86.2	66.7	55.1	67.4
RoIAlign	64.2	86.6	69.7	58.7	73.0

8.2 Mask branch-segmentation

It is a pixel to pixel task which utilizes using an FCN to exploit the layouts of masks which are spatial. As it can have been seen in the table below with the utilization of ResNet-50-FPN the MLP which are Multi-layer perceptron and FCN are compared. 2.1 mask AP is gained over MLP with the utilization of FCN. It is to be noted that to make sure that conv

layers which are the head of FCN are not pre trained this backbone is chosen to make sure the comparison with MLP is fair.

Table 8.6 Utilization of ResNet-50-FPN the MLP which are Multi-layer perceptron and FCN are compared

	mask branch	AP	<i>AP</i> ₅₀	<i>AP</i> ₇₅
MLP	fc: 1024->1024->80 · 28 ²	31.5	53.7	32.8
MLP	fc: 1024->1024->80 · 28 ²	31.5	54.0	32.6
FCN	Conv: 256->256->256->80	33.6	55.2	35.3

8.3 Bounding box detection results

As it can be seen in the table below the Mask R-CNN are compared to the state of art COCO bounding box detection.

Table 8.7 Mask R-CNN are compared to the state of art COCO bounding box detection

	backbone	<i>AP</i> _{bb}	<i>AP</i> _{bb50}	<i>AP</i> _{bb75}	<i>AP</i> _{bb_S}	<i>AP</i> _{bb_M}	<i>AP</i> _{bb_L}
Faster RCNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-	ResNet-	36.2	59.1	39.0	18.2	39.0	48.2

CNN w FPN [27] Faster R-	101-FPN						
CNN by G-RMI [21]	Inception- ResNet- v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R- CNN w TDM [39]	Inception- ResNet- v2TDM	36.8	57.7	39.2	16.2	39.8	52.1
Faster R- CNN, RoIAlign	ResNet- 101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R- CNN	ResNet- 101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R- CNN	ResNeXt- 101-FPN	39.8	62.3	43.4	22.1	43.2	51.2

Full Mask RCNN model is trained to achieve these results. However, classification and box outputs are only utilized at the inference. A gap of 2.7 points between 37.1 and 39.8 between Mask and box AP is attained by Mask R-CNN. Thus the approach which it have taken is closing the gap between the object detection and the instance segmentation task which is more challenging.

8.3.1 Evaluation of MASK R-CNN for human pose estimation

The evaluation is done on the key points of the person which are AP^{kp} and experiment is performed with the utilization of ResNet50-FPN as a backbone. As it can be seen in the table below the results AP^{kp} is shown is higher than the COCO with 0.9 points.

Table 8.8 AP^{kp} experiment performed with the utilization of ResNet50-FPN

	AP^{kp}	$AP^{kp_{50}}$	$AP^{kp_{75}}$	AP^{kp_M}	AP^{kp_L}
CMU-Pose+++ [42]	61.8	84.9	67.5	57.1	68.2
G-RMI [49] †	62.4	84.0	68.5	59.1	68.1
Mask R-CNN, keypointonly	62.7	87.0	68.4	57.4	71.1
Mask R-CNN, key point & mask	63.1	87.3	68.7	57.8	71.4

The method which is adopted is simple and fast. Boxes, segments, and key points can be simultaneously can be predicted while running at 5 fps by this model which is unified. On test-dev an improvement in AP^{kp} to 63.1 is shown with the improvement of a segmented branch. Addition of a branch to only box or key points only version can show consistent improvement in these tasks.

8.3.2 Over-simplified supervision

The MASK R-CNN based methods still have some drawbacks. The text detected with the R-CNN method are not so clear and not so much expressive. The text presents in natural areas that are in square are more useful for the text detection. But the R-CNN method

separates the text and background differently rather than to produce a text mask for the specific shape thus it can't take advantage of the given label.[39]

8.3.3 Imprecise segmentation labels

The other drawback is that the R-CNN methods implementation includes the conversation of the square text area that is detected into the pixel binary supervision signals. The Mask R-CNN applies directly the semantic segmentation for the square text detection. Unfortunately, this procedure has some flaw. The quality of generated pixel-level labels results is not satisfactory. In the figure 8.5.1 you can see that many pixels that are open to the background and represented in the text region are incorrectly defined as foreground pixels which may hurt the performance of this method.[39]



Figure 8.5.1 Illustrates Paradigms of imprecise segmentation labels [39]

8.3.4 Error propagation

In the Mask R-CNN method, the Mask usually assume the square boundary for the text and then the whole process performs in which it performs the semantic segmentation within the selected square boundary for the text and then produces the result. When this method is implemented in the real-scenarios, it just works properly for the simple text images like four words text in straight line or eight-word text in a straight line but when the image is not so simple or in a curve or may be a sentence and the square box is failed to cover or detect the whole text it broke the words area. Simply said, this method is not sufficient with large text like paragraph written in the italic style. It probably due to the text selected in the area the semantic segmentation just processes those words which presented in that square text box. [39]

9. RESULTS

The method which is proposed was tried and tested while considering the images which were 4400 in number. The 4400 images were first inserted in the platform which is ICDAR. The ICDAR platform evaluated images who have value which is non zero. Other images which were 6600 in images in number. Those 6600 images were excluded from the evaluation because they have value which is zero. As 6600 images took value which is zero which is future became problem in the form of uploading of those images which have displayed zero value.

Table 9.1 Mask R-CNN Method results

H-mean	Precision	Recall	Average precision
0.57452943	0.63082802	0.59836401	0.53466988

For Latin text

Table 9.2 Mask R-CNN Method results for Latin text

H-mean	Precision	Recall	Average precision
0.70753003	0.71198217	0.75048871	0.70146512

The method which is utilized in this research is called DSIT-UOA.

The image below is showing is showing good prediction which was achieved with the utilization of this technique.



Figure 9.1 Displaying high score achievement for text detection

The images below is showing 100% results which is achieved with the utilization and implementation of this technique.

Sample Ranking	Average Precision	Recall	Precision	Hmean
DSIT-UQA	100.00%	100.00%	75.00%	85.71%
TH-DL-v2	100.00%	100.00%	50.00%	66.67%
PSENet_v1	100.00%	100.00%	100.00%	100.00%
TH-DL-v1	100.00%	100.00%	50.00%	66.67%
RRPN	100.00%	100.00%	50.00%	66.67%
Tencent-DPPR Team (Method_v0.2)	100.00%	100.00%	60.00%	75.00%
mm-maskrcnn_v2	100.00%	100.00%	60.00%	75.00%

Figure 9.2 Displaying high score achievement for text detection

The images below is showing 100% results which is achieved with the utilization and implementation of this technique.

Sample Ranking	Average Precision	Recall	Precision	Hmean
DSIT-UOA	100.00%	100.00%	100.00%	100.00%
Tencent-DPPR Team (Method_v0.3)	100.00%	100.00%	100.00%	100.00%
RRPN	100.00%	100.00%	100.00%	100.00%
multi-stage_text_detector	100.00%	100.00%	100.00%	100.00%
Tencent-DPPR Team (Method_v0.2)	100.00%	100.00%	100.00%	100.00%
text-mountain	100.00%	100.00%	83.33%	90.91%
mm-maskrcnn_v2	100.00%	100.00%	100.00%	100.00%

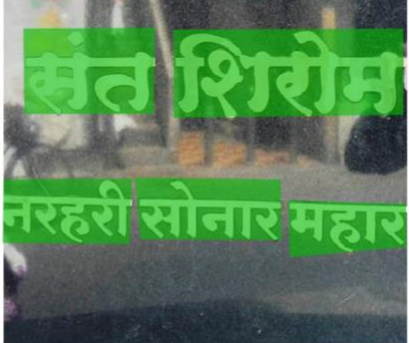
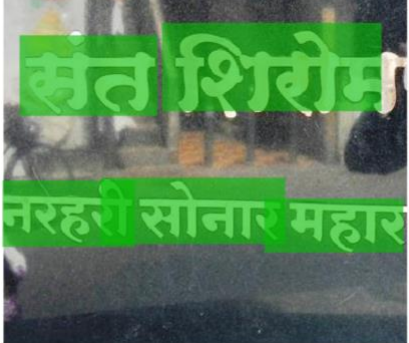
Ground Truth	Detection
	

Figure 9.3 Displaying high score achievement for text detection

The result below is showing 100% results which is achieved with the implementation and utilization of the implemented Mask-RCNN model..



DSIT-UOA	12.50%	25.00%	50.00%	33.33%
Ground Truth				
Detection				

Figure 9.4 Displaying law score achievement for text detection

The image below is representing images which have shown poor results or you can say that the images which this proposed technique failed to utilize to produce better results.

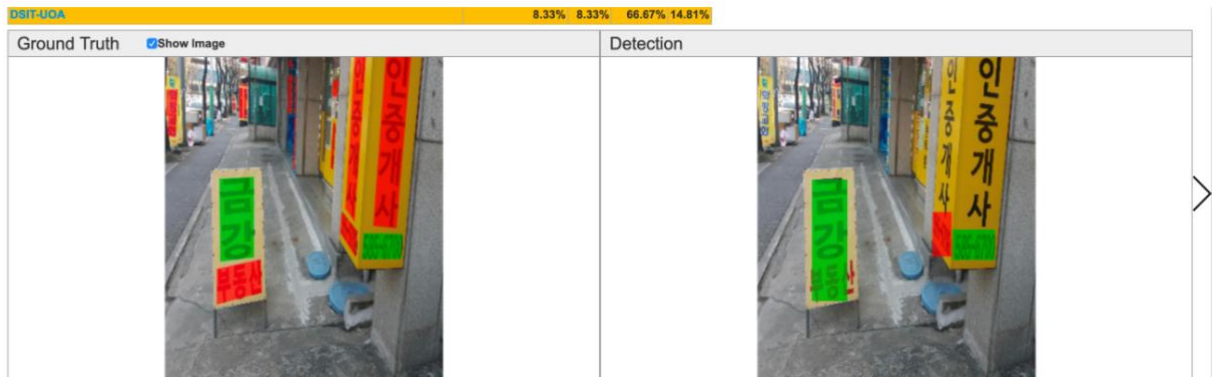


Figure 9.5 Displaying the prediction from the Mask-RCNN model to the right and the correct ground truth to the left.

The image below is representing differences in terms of detection and the differences are for the texts which are handwritten.



Figure 9.6 Displaying hand written prediction

The low prediction results might be because the amount of tested hyperparameters might be insufficient, meaning that the more combinations might need to be tested to get the global minimum. Moreover, some other explanations can be seen in the future work section.

10. FUTURE WORK

The methods which are suggested in this research proposal can offer applications which include Multilingual text detection and Scene text detection. There is a requirement of adjustments. Since these methods are alleviating the inaccuracy in the labeling of the boundary of the instance so there is a requirement of innovation which can help in achieving better results in terms of boundary detection. Following are the dimensions which needs to be addressed in the future work;

- Augmentation of the image

To increase the dataset, augmentation is a good and cheap technique. By flipping the images and labels horizontally for example, it would increase the dataset by 2. To be sure of the optimal augmentation techniques one would have to test out multiple and check the change in accuracy on external test data once the model is trained on the data.

- Removal of noise

Noise removal is a good technique to use in order to decrease irregularities in the background which might decrease the accuracy of the model in the end. By using noise removal filters such as Averaging filter, Median Filter or Gaussian filter the images might have “cleaner” data for the model to extract the information from. To be sure which denoising filter technique suits this dataset the best one would have to test out all of the preprocessing techniques in order to be sure of which is the best.

- Normalization of the images

Normalization of images is done to once again increase the models predictive accuracy. This is to not get the model to bias the classes towards specific color or pixel intensity if the dataset at hand is unbalanced.

- Enhancement in terms of the contrast

By increasing the contrast after denoising the image and normalization, the model would probably make clearer distinction between the background and the script classes.

- Bayesian hyperparameter optimization

By applying a hyperparameters optimization method as Bayesian optimization it would be much more effective search for the optimal hyperparameters than using random search. Bayesian optimization tests more hyperparameters around the area which seems to be optimal, instead of testing at random.

11. ANNEX I

Installation Manual

The code is accessible in gitlab at: <https://gitlab.com/nnaoum/master-dsit.git>

Installation Steps:

Create the following folders in your google drive:

- test_maskRCNN

This folder is to put your images that you want to predict on.

- mask_weight

If you want to train the model by your own, then the last weight from each training session will be saved in this folder.

Put the file: mask_rcnn_root_0339.h5 in your mask_weights folder.

- Mask_RCNN_data/train_images

Here are the images for the training. The images got split into a training dataset (70%) and validation (30%). In each of the folder there is a CSV-file with the annotation for each image classes.

Put the following files in your main google drive folder:

- Maskrcnn2.zip

This zip-file contain the whole model that I got from https://github.com/matterport/Mask_RCNN. In google colab you will extract this file to either train/predict.

- rootX1.zip

This zip-file contains the modified file which takes in the training images and CSV-files(lables) to train the model.

- train_MaskRCNN.ipynb

This is the google colab file which runs the model training.

- Detection.ipynb

This is the google colab file which runs the model detection/prediction

Train model.

Run google colab in GPU mode

Open train_MaskRCNN.ipynb with google colab. Then run the code blocks. It will ask you to give it authorization to connect to your google colab.



Figure 11.1 Google colab running

There on run all the code blocks. The first time in a session that you run the code block which starts the training of the model this error pops up:

```

2020-10-27 00:27:47.645126: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but there must be at least one positive value.
2020-10-27 00:27:47.645653: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1767] Adding visible gpu devices: 0
2020-10-27 00:27:47.649842: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.1
2020-10-27 00:27:47.651389: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1180] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-10-27 00:27:47.651419: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1186] 0
2020-10-27 00:27:47.651427: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:  N
2020-10-27 00:27:47.651852: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but there must be at least one positive value.
2020-10-27 00:27:47.652488: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but there must be at least one positive value.
2020-10-27 00:27:47.653079: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:39] Overriding allow_growth setting because the TF_FORCE_GPU_ALLOW_GROWTH environment variable is enabled.
2020-10-27 00:27:47.653134: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 15.0GiB memory)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:172: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:181: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:188: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

Traceback (most recent call last):
  File "rootX1.py", line 417, in <module>
    train(model)
  File "rootX1.py", line 236, in train
    dataset_train.load_balloon(args.dataset, "train")
  File "rootX1.py", line 109, in load_balloon
    self.annotations = pd.read_csv(os.path.join(dataset_dir, file_name))
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 686, in read_csv
    return read(filepath_or_buffer, kwds)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 452, in read
    parser = TextFileReader(fp_or_buf, **kwds)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 946, in __init__
    self._make_engine(self.engine)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 1178, in _make_engine
    self._engine = CParserWrapper(self.f, **self.options)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 2008, in __init__
    self._reader = parsers.TextReader(src, **kwds)
  File "pandas/_libs/parsers.pyx", line 382, in pandas._libs.parsers.TextReader._cinit_
  File "pandas/_libs/parsers.pyx", line 674, in pandas._libs.parsers.TextReader._setup_parser_source
ValueError: [Errno 5] Input/output error: /content/gdrive/My Drive/Mask RCNN data/train images/train/data train.csv
    
```

Figure 11.2 Start Running

But you just have to rerun the code block and then it works fine. It takes around 1 hour or more before you can see that the model is training. It takes a lot of time before it starts as it has to load 10.000 images and create the annotation boxes for each word that it has to detect.

When the model is training you can see it like this:

```
+ Kod + Test
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:675: calling Constant._init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version. Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:940: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:700: The name tf.summary.merge_all is deprecated. Please use tf.compat.v1.summary.merge_all instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:700: The name tf.summary.FileWriter is deprecated. Please use tf.compat.v1.summary.FileWriter instead.
/usr/local/lib/python3.6/dist-packages/keras/engine/training.py:1203: UserWarning: Using a generator with 'use_multiprocessing=True' and multiple workers may duplicate your data. Please consider using the 'keras.utils.Sequence' class
UserWarning: Using a generator with 'use_multiprocessing=True'
Epoch 67/600
2020-10-26 19:51:53.554227: I tensorflow/stream_executor/platform/default/dso_loader.cc:44: Successfully opened dynamic library libcudnn.so.10
2020-10-26 19:51:55.167889: I tensorflow/stream_executor/platform/default/dso_loader.cc:44: Successfully opened dynamic library libnvidia-ml.so.1
139/140 [=====] - ETA: 1s - loss: 1.7057 - rpn_class_loss: 0.0513 - rpn_bbox_loss: 0.6743 - mrcnn_class_loss: 0.4154 - mrcnn_bbox_loss: 0.2957 - mrcnn_mask_loss: 0.2690/usr/local/lib/python3.6/dist-packages/keras/callbacks.py:791: The name tf.summary is deprecated. Please use tf.compat.v1.Summary instead.
140/140 [=====] - 236s 2s/step - loss: 1.7052 - rpn_class_loss: 0.0511 - rpn_bbox_loss: 0.6729 - mrcnn_class_loss: 0.4175 - mrcnn_bbox_loss: 0.2949 - mrcnn_mask_loss: 0.2687 - val_loss: 1.7052
Epoch 68/600
74/140 [=====] - ETA: 57s - loss: 1.6616 - rpn_class_loss: 0.0428 - rpn_bbox_loss: 0.6240 - mrcnn_class_loss: 0.4218 - mrcnn_bbox_loss: 0.2845 - mrcnn_mask_loss: 0.2686
107/140 [=====] - ETA: 28s - loss: 1.6617 - rpn_class_loss: 0.0440 - rpn_bbox_loss: 0.6560 - mrcnn_class_loss: 0.4174 - mrcnn_bbox_loss: 0.2930 - mrcnn_mask_loss: 0.2706
Traceback (most recent call last):
  File "/root/.Mask_RCNN/mrcnn/model.py", line 1709, in data_generator
    use_mini_mask_config=USE_MINI_MASK)
  File "/root/.Mask_RCNN/mrcnn/model.py", line 1212, in load_image_gt
    mask_class_id = dataset.load_mask(image_id)
  File "/rootX1.py", line 187, in load_mask
    mask[r, 0, 0, 0] = 1
IndexError: index 0 is out of bounds for axis 1 with size 914
139/140 [=====] - ETA: 0s - loss: 1.6667 - rpn_class_loss: 0.0415 - rpn_bbox_loss: 0.6421 - mrcnn_class_loss: 0.4134 - mrcnn_bbox_loss: 0.2977 - mrcnn_mask_loss: 0.2720
2020-10-26 19:52:15.1552 bytes == 0x00000000 0x7f3ab2d22001 0x7f3ab2d47463 0x7f3ab2e0a2b0 0x7f3ab2e0ca0f 0x7f3ab2e30248 0x010a7f3 0x010cfd6 0x0107224 0x0109260 0x010a64d 0x010c1f4 0x0107216 0x0109260 0x010a64d
140/140 [=====] - 130s 19/step - loss: 1.6663 - rpn_class_loss: 0.0414 - rpn_bbox_loss: 0.6420 - mrcnn_class_loss: 0.4137 - mrcnn_bbox_loss: 0.2973 - mrcnn_mask_loss: 0.2717 - val_loss: 1.6663
Epoch 69/600
87/140 [=====] - ETA: 1:03 - loss: 1.7124 - rpn_class_loss: 0.0432 - rpn_bbox_loss: 0.6486 - mrcnn_class_loss: 0.4304 - mrcnn_bbox_loss: 0.3092 - mrcnn_mask_loss: 0.2813
121/140 [=====] - ETA: 51s - loss: 1.7501 - rpn_class_loss: 0.0455 - rpn_bbox_loss: 0.6573 - mrcnn_class_loss: 0.4567 - mrcnn_bbox_loss: 0.3076 - mrcnn_mask_loss: 0.2826
139/140 [=====] - ETA: 0s - loss: 1.7543 - rpn_class_loss: 0.0474 - rpn_bbox_loss: 0.6599 - mrcnn_class_loss: 0.4683 - mrcnn_bbox_loss: 0.3012 - mrcnn_mask_loss: 0.2774

~/Mask_RCNN/dataset/weights
from google.colab import files
!ls
import shutil
import os
fold=os.listdir()
list_files=os.listdir(fold)
list_weights=list_files[-1]
```

Figure 11.3 Displaying training part

The weights from the training is then saved in the folder mask_weights in google drive.

Detect script / Predict on images Run google colab in GPU mode

Open the file Detection.ipynb in your google colab.

The model uses the weights located in the folder mask_weights, now you can use the weights:

'mask_rcnn_root_0339.h5' as test weights.

By running all of the code blocks you will do predictions on the images located in the folder test_maskRCNN which is in your main google drive directory.

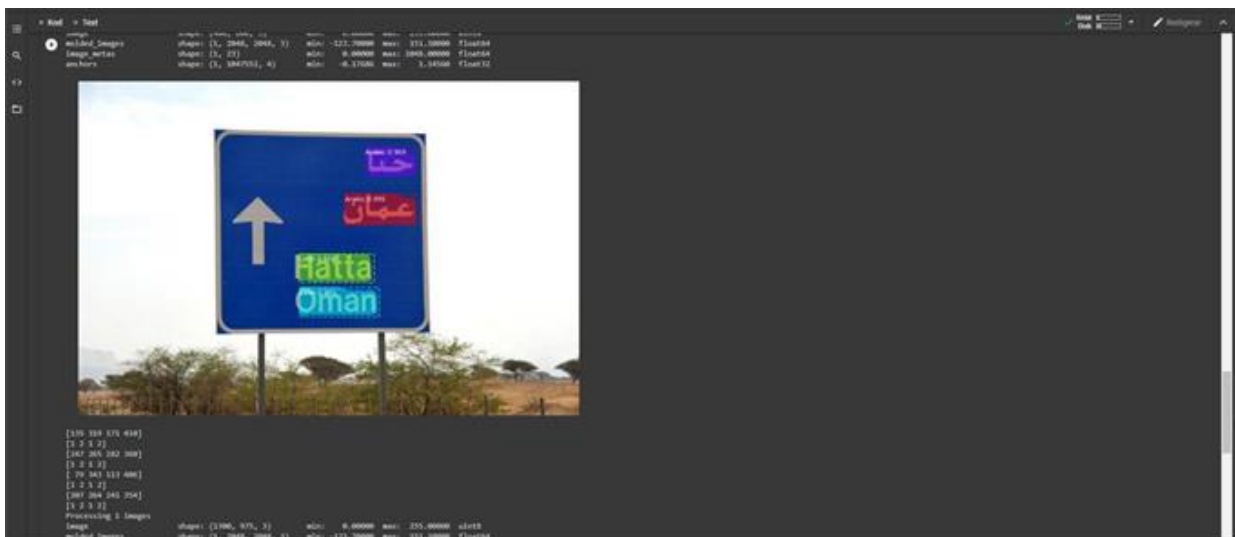


Figure 11.4 Displaying text prediction

Under the predicted image you can see one print with the list [1 2 1 2] which is the class id. 1 for Arabic, 2 for Latin. And the other lists like [135 319 172 410] is the [x1, y1, x2, y2] for each segmentation box which it detects on the image.

12. REFERENCES

- [1] Wang, Y., Yan, J., Sun, Q., Li, J., & Yang, Z. (2019). A MobileNets Convolutional Neural Network for GIS Partial Discharge Pattern Recognition in the Ubiquitous Power Internet of Things Context: Optimization, Comparison, and Application. *IEEE Access*, 7, 150226-150236.
- [2] Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 589-597).
- [3] Guan, S., Meng, C., Xie, Y., Wang, Q., Sun, K., & Wang, T. (2019). Deformable cardiovascular image registration via multi-channel convolutional neural network. *IEEE Access*, 7, 17524-17534.
- [4] Wang, Z., Liang, M., & Delahaye, D. (2018). A hybrid machine learning model for short-term estimated time of arrival prediction in terminal manoeuvring area. *Transportation Research Part C: Emerging Technologies*, 95, 280-294.
- [5] Yuan, X., Qi, S., Shardt, Y., Wang, Y., Yang, C., & Gui, W. (2020). Soft sensor model for dynamic processes based on multichannel convolutional neural network. *Chemometrics and Intelligent Laboratory Systems*, 104050.
- [6] Arik, S. Ö., Jun, H., & Diamos, G. (2018). Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1), 94-98.
- [7] Nie, D., Zhang, H., Adeli, E., Liu, L., & Shen, D. (2016, October). 3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. In *International conference on medical image computing and computer-assisted intervention* (pp. 212-220). Springer, Cham.
- [8] Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*.
- [9] Cao, X., Wang, Z., Zhao, Y., & Su, F. (2018). Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 734-750).
- [10] Singh, U. P., Chouhan, S. S., Jain, S., & Jain, S. (2019). Multilayer convolution neural network for the classification of mango leaves infected by anthracnose disease. *IEEE Access*, 7, 43721-43729.
- [11] Fan, R., Feng, R., Wang, L., Yan, J., & Zhang, X. (2020). Semi-MCNN: A Semisupervised Multi-CNN Ensemble Learning Method for Urban Land Cover Classification Using Submeter HRRS Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 4973-4987.
- [12] Wang, H., Ding, S., Wu, D., Zhang, Y., & Yang, S. (2019). Smart connected electronic gastroscope system for gastric cancer screening using multi-column convolutional neural networks. *International Journal of Production Research*, 57(21), 6795-6806.

- [13] Shen, W., Zhou, M., Yang, F., Yang, C., & Tian, J. (2015, June). Multi-scale convolutional neural networks for lung nodule classification. In International Conference on Information Processing in Medical Imaging (pp. 588-599). Springer, Cham.
- [14] Petrozziello, A., Redman, C. W., Papageorghiou, A. T., Jordanov, I., & Georgieva, A. (2019). Multimodal convolutional neural networks to detect fetal compromise during labor and delivery. *IEEE Access*, 7, 112026-112036.
- [15] Petrozziello, A., Redman, C. W., Papageorghiou, A. T., Jordanov, I., & Georgieva, A. (2019). Multimodal convolutional neural networks to detect fetal compromise during labor and delivery. *IEEE Access*, 7, 112026-112036.
- [16] B. Kisacanin, V. Pavlovic, and T. S. Huang. "Real-Time Vision for Human-Computer Interaction. Springer", Heidelberg, 2005.
- [17] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor. "Vision-based target geolocation using a fixed-wing miniature air vehicle". *Journal of Intelligent and Robotic Systems*, 47(4):361-382, 2006.
- [18] Y. K. Ham, M. S. Kang, H. K. Chung, R. H. Park, and G. T. Park. "Recognition of raised characters for automatic classification of rubber tires". *Optical Engineering*, 34(1):102-109, 2005.
- [19] G. N. DeSouza and A. C. Kak. "Vision for mobile robot navigation: A survey". *IEEE Trans. PAMI*, 24(2):237-267, 2002.
- [20] S. Tsai, H. Chen, D. Chen, G. Schroth, R. Grzeszczuk, and B. Girod. "Mobile visual search on printed documents using text and low bit-rate features". In *Proc. of ICIP*, 2011.
- [21] Z. Yingying, Y. Cong, and B. Xiang. "Scene Text Detection and Recognition: Recent Advances and Future Trends". *Frontiers of Computer Science*, 2015.
- [22] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: "An efficient and accurate scene text detector". In *CVPR*, pp. 2642-2651, 2017.
- [23] Y. Liu and L. Jin. "Deep matching prior network: Toward tighter multi-oriented text detection". In *CVPR*, pp. 3454-3461, 2017.
- [24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. "SSD: Single shot multibox detector". In *ECCV*, pp. 21-37, 2016.
- [25] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In *CVPR*, pp. 3431-3440, 2015.
- [26] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In *NIPS*, pp. 91-99, 2015.
- [27] Z. Huang, Z. Zhong, L. Sun, and Q. Huo. "Mask R-CNN with Pyramid Attention Network for Scene Text Detection". *Conference Paper*, January 2019.

- [28] Y. Dai, Z. Huang, Y. Gao, and K. Chen. "Fused text segmentation networks for multi-oriented scene text detection". arXiv preprint arXiv:1709.03272, 2017.
- [29] Q. Yang, M. Cheng, W. Zhou, Y. Chen, M. Qiu, and W. Lin. "Inceptext: A new inception-text module with deformable psroi pooling for multi-oriented scene text detection". arXiv preprint arXiv:1805.01167, 2018.
- [30] D. Deng, H. Liu, X. Li, and D. Cai. "Pixellink: Detecting scene text via instance segmentation". In AAAI, pp. 6773-6780, 2018.
- [31] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. "Fully convolutional instance-aware semantic segmentation". arXiv preprint arXiv:1611.07709, 2016.
- [32] C. Yao, X. Zhang, X. Bai, W. Liu, Y. Ma, and Z. Tu. "Rotation-invariant features for multi-oriented text detection in natural images". PLoS One, 8(8), 2013.
- [33] C. Yao, X. Bai, B. Shi, and W. Liu. "Strokelets: A learned multi-scale representation for scene text recognition". In Proc. of CVPR, 2014.
- [34] Nayef, N., Patel, Y., Busta, M., Chowdhury, P. N., Karatzas, D., Khlif, W., ... & Ogier, J. M. (2019, September). 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) (pp. 1582-1587). IEEE.
- [35] Saha, S., Chakraborty, N., Kundu, S., Paul, S., Mollah, A. F., Basu, S., & Sarkar, R. (2020). Multi-lingual scene text detection and language identification. Pattern Recognition Letters, 138, 16-22.
- [36] Nguyen, D. C., Delalandre, M., & Conte, D. (2020, February). Fast scene text detection with RT-LoG operator and CNN. In 15th International Conference on Computer Vision Theory and Applications.
- [37] Long, S., & Yao, C. (2020). UnrealText: Synthesizing realistic scene text images from the unreal world. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5488-5497).
- [38] Lundgren, A., Castro, D., Lima, E., & Bezerra, B. (2019, September). OctShuffleMLT: A Compact Octave Based Neural Network for End-to-End Multilingual Text Detection and Recognition. In 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW) (Vol. 4, pp. 37-42). IEEE.
- [39] Jingchao Liu , Xuebo Liu* , Jie Sheng , Ding Liang , Xin Li , Qingjie Liu. "Pyramid Mask Text Detector". arXiv preprint arXiv: arXiv:1903.11800, 2019