Scanning Single Shot Detector for Math in Document Images

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Richard Zanibbi, Supervisor

Dr. Zack Butler, Reader

Dr. Joe Geigel, Observer

Scanning Single Shot Detector for Math in Document Images

by

Parag Shrikrishna Mali

THESIS

Presented to the Faculty of the Department of Computer Science

Golisano College of Computer and Information Sciences

Rochester Institute of Technology

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

Rochester Institute of Technology

August 2019

Dedicated to my family and lifetime friends

Acknowledgments

I would like to extend my deepest gratitude to my advisor and director of Document and Pattern Recognition Lab (DPRL) Dr. Richard Zanibbi for his unwavering guidance in my academic pursuit, constructive criticism, insightful suggestions, and, profound belief in my work. I am extremely grateful to him for welcoming me in the DPRL and letting me work on different research projects in the lab while helping me understand the importance of following the scientific procedure in computer science research. I would also like to extend my sincere thanks to Dr. Zack Butler and Dr. Joe Geigel for serving on my thesis committee and providing me invaluable feedback on my research.

I am also grateful to DPRL members Puneeth Kukkadapu and Mahshad Mahdavi for working with me in the creation of TFD-ICDAR 2019 dataset for math expression detection. I would like to thank other DPRL members Abishai Dmello, Behrooz Mansouri, Gavin Nishizawa, and, Wei Zong for very productive discussions and ingenious suggestions. I also had the great pleasure of working with Douglas W. Oard (University of Maryland), Dr. C. Lee Giles (The Pennsylvania State University), Dr. Anurag Agarwal (Rochester Institute of Technology), and, Dr. Jian Wu (Old Dominion University) during my thesis. I would like to thank Shaurya Rohatgi and Neisarg Dave for their

suggestions during our meetings.

I am deeply indebted to my sister Aarti Gorade and brother-in-law Hanumant Gorade for their unparalleled support and guidance in my life. I cannot begin to express my gratitude to my parents, Shrikrishna Mali and Sunanda Mali, who are the source of my strength and my tireless supporters.

Abstract

Scanning Single Shot Detector for Math in Document Images

Parag Shrikrishna Mali, M.S.
Rochester Institute of Technology, 2019

Supervisor: Dr. Richard Zanibbi

We introduce the Scanning Single Shot Detector (ScanSSD) for detecting both embedded and displayed math expressions in document images using a single-stage network that does not require page layout, font, or, character information. ScanSSD uses sliding windows to generate sub-images of large document page images rendered at 600 dpi and applies Single Shot Detector (SSD) on each sub-image. Detection results from sub-images are pooled to generate page-level results. For pooling sub-image level detections, we introduce new methods based on the confidence scores and density of detections. ScanSSD is a modular architecture that can be easily applied to detecting other objects in document images.

For the math expression detection task, we have created a new dataset called TFD-ICDAR 2019 from the existing GTDB datasets. Our dataset has 569 pages for training with 26,396 math expressions and 236 pages for testing with 11,885 math expressions. ScanSSD achieves an 80.19% F-score at IOU50 and a 72.96% F-score at IOU75 on TFD-ICDAR 2019 test dataset. An earlier version of ScanSSD placed 2^{nd} in the ICDAR 2019 competition on the Typeset Formula Detection (TFD). Our data and code are publicly available at `https://github.com/MaliParag/TFD-ICDAR2019` and `https://github.com/MaliParag/ScanSSD`, respectively.

Table of Contents

List of Tables

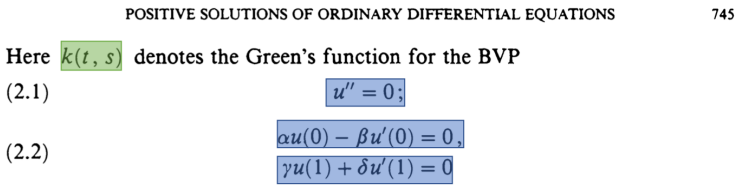
List of Figures

Chapter 1

Introduction

As PDF format greatly facilitates document sharing and printing, it has become a standard for publishing scientific documents. The latest PDF specifications make it possible to embed structural information (such as identification of special elements like figures, tables, and, footnotes), but most existing software which generates PDF documents choose not to make use of this feature and only focus on the visual representation of the documents. The lack of such structural information challenges automatic information retrieval and information extraction. Multiple methods have been proposed for the detection of tables, diagrams, headers/footers, cross-references, paragraphs, algorithms, and, mathematical expressions. These methods have many applications including but not limited to the image reconstruction, diagram classification, text line transcription, document format conversion, screen readers, and, math recognition.

Math expressions in the PDF documents can not be easily located, searched, and, reused as most of the PDF documents only contain page rendering information. So, it has become important to come up with the math expression recognition method for the PDF documents. The mathematical ex-


POSITIVE SOLUTIONS OF ORDINARY DIFFERENTIAL EQUATIONS 745

Here $k(t, s)$ denotes the Green's function for the BVP

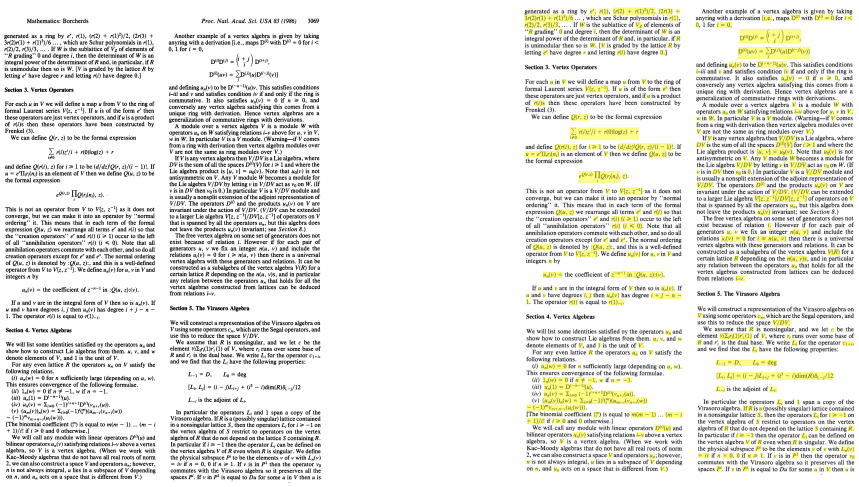
(2.1) $$u'' = 0;$$

(2.2) $$\alpha u(0) - \beta u'(0) = 0,$$
$$\gamma u(1) + \delta u'(1) = 0$$

Figure 1.1: Examples of embedded and displayed math expressions. Green highlight shows embedded (or in-line) math expressions and blue highlight shows displayed (or offset) math expressions.

pression recognition is an active area of research since 1967 [1]. Zanibbi et al. [56] describes four problems that a math recognition system should address: math detection, math symbols extraction and recognition, layout analysis, and, mathematical content interpretation. The detection of math expressions is the first step in the recognition process.

Many proposed methods for math detection in PDF documents operate on the document page images. These methods use additional information available from PDF documents like page layout, character labels, character locations, font sizes, etc. However, there are multiple challenges while extracting information from the PDF documents. PDF documents are generated by different tools and their character information quality can be very different. Lin et al. [37] point out that a math expression element can be composed of several types of objects (e.g. text, image, graph). For example, the square root sign in a PDF generated from LaTeX contains the text object representing a radical sign and a graphical object representing the horizontal line. As a result, it becomes difficult to directly match the PDF object to mathematical elements in the math expressions.



(a) The input to the system will be a PDF document

(b) Yellow regions show the bounding boxes for the displayed and embedded math expressions

Figure 1.2: Input and expected output of a math detection system for a PDF document page

Math expressions in the documents can be displayed expressions that are offset from the text paragraphs or expressions that are embedded in the text lines. Figure 1.1 shows examples of embedded and displayed math expressions. Displayed expression detection is relatively easy, as they differ significantly in height and width of the line, character size, and, symbol layout [17]. Embedded mathematical expressions can include complex mathematical structures like Σ besides symbols or variables. Many embedded math expressions also have their explicit natural language definitions like 'where w is the set of words'. Iwatsuki et al. [27] concluded that distinguishing dictionary words that appear in italics and embedded mathematical expressions is a non-trivial task. As a result, many approaches have been proposed specifically for embedded math expression detection [36, 27] whereas others detect only displayed math expressions [10, 15].

The methodology proposed in this thesis is inspired by the following **research questions:**

1. Can we detect both embedded and displayed math expressions in typeset documents using only one framework?

2. Can we detect both embedded and displayed math expressions in typeset document images without the use of additional information like page layout, character labels, character location or OCR codes?

3. Are object detection methods like Single Shot Multibox Detector (SSD) used for detecting objects in natural scenes applicable for detecting math

expressions in typeset document images? If yes, which changes should be made in object detection architectures to make it well-suited for this task?

Thesis Statement: Both embedded and displayed math expressions in typeset document images can be detected accurately in one framework using deep learning-based object detection methods without the use of additional information like page layout, character labels, character locations, font size, font type, or OCR codes.

We use intersection over union (IOU) metric to calculate the accuracy of the system. Please refer Chapter 3 for more details on the evaluation method.

Contributions: We have created a new dataset for math expression detection in typeset documents. In Chapter 3, we define the problem statement, evaluation methods, and, discuss the creation of a new dataset for math expression detection. We make our dataset publicly available at `https://github.com/MaliParag/TFD-ICDAR2019`.

In Chapter 4, we propose ScanSSD architecture to detect both embedded and displayed math expressions in the typeset document images. Our architecture uses sliding windows to generate sub-images of the large document page image and detects math expressions in the generated sub-images. We introduce pooling methods that use the sub-image level detections to generate page level detections. Our method achieves F-score of 80.19% at IOU50

and 72.96% at IOU75 on TFD-ICDAR 2019 test dataset. Our code is publicly available at `https://github.com/MaliParag/ScanSSD`.

We discuss different rule-based and learning-based methods for page object detection Chapter 2. Chapter 5 discusses different models and their performance on our validation and test datasets. Also, we provide character level results and compare our results with other methods. Details about the validation set are provided in Appendix A, and additional results are provided in Appendix B and Appendix C.

Chapter 2

Background

Document image understanding includes detection of objects like the math expressions, figures, text-lines, tables, etc. from the document image. Different layouts of documents and many variations in the page's structure make it difficult to design an efficient and accurate system for the detection of page objects. The document image understanding problem can be simplified if we detect one page object at a time. In this chapter, we disuss methods for detecting one as well as multiple page objects at a time. We mainly focus on math expression detection.

2.1 Page Object Detection

Li et al. [31] describe the two major problems with page object detection. First, document images differ greatly from the natural images in terms of color and texture which makes texture-based region proposal generation and selective search methods not useful for this task. Second, page objects have a more diverse scale and aspect ratio than natural scene objects. Furthermore, the background information inside the page objects may be larger than those between objects, which makes the grouping of object elements difficult.

In this section, we summarize methods for detecting different page objects like algorithms, headers/footers, cross-reference, paragraphs, diagrams, and, tables. In the next section we describe methods for math expression detection.

Algorithm Detection For algorithm detection, Bhatia et al. [2, 3] propose a three-step method. In the first step, they use PDFTextStream [1] to extract all the text from the PDF documents. In the second step, they use grammar for algorithm captions to find out if algorithms are present in the text. In the third step, if the algorithm is present in the text, the text is further processed to extract the lines which describe the algorithm using a naive bayes classifier trained on the variety of content and context features.

Tuarob et al. [50] use PDFBox[2] to extract text lines from the documents along with the font size information from each text line. They use heuristic rules that detect algorithms which accompanied the captions. They also propose a machine learning-based method that directly detects the algorithms instead of the captions. Their method detects the sparse boxes in the document and classifies them as an algorithm or not using features like font style, context, content, and, structure.

[1] http://snowtide.com/PDFTextStream
[2] http://pdfbox.apache.org/

Header/Footer Detection Lin proposed a robust header and footer extraction method using fuzzy string matching and page association [34]. They base their method on the observation that the header and footer are text lines and have a similar structure with the neighboring pages. In their method, they need bounding boxes and text from the document. In their experiments, they look at the eight neighboring pages and for each page, they find a similarity score between each line on these pages based on edit distance cost and geometric similarity.

Dejan et al. [8] propose a header/footer detection method based on the textual variability score. They claim that the header/footer zones have much lower textual variety than the body of the page. Using a few heuristic rules, they find the header and footer regions which minimize the textual variability score. This method works well when the headers and footers are homogeneous in the document and partially fails for the documents that are composed of parts with different headers and footers.

Cross-reference Detection Footnotes/endnotes, figure/table captions, and references are very common in PDF documents. However, the detection of these cross-references is seldom addressed. Li et al. [30] describe a two-step process for footnote identification. They used different features, thresholds, and rules for different types of cross-references. In the first step, they use a PDF parser to get the basic results of page segmentation and layout analysis. In the next step, they extract features based on the cross-reference type and

cluster the results. Finally, they match the references to their corresponding entities, i.e. footnotes, tables or figures.

Paragraph Detection Darvishy et al. [7] describe an algorithm for paragraph detection for an open-source PDF Accessibility Validation Engine (PAVE). They examine the distribution of element heights and elements that exceed the median of the element height with a small error are ignored. It is possible to extract the information regarding element heights, widths, etc. from the born-digital PDF documents. In the next step, they group text elements into blocks based on spatial proximity. To create the paragraphs in the correct order, they use the XY-cut algorithm [41] to determine the ordering on the text blocks. There are other document format conversion methods like [8] which also use XY-cut algorithm for paragraph detection.

Diagram Detection Futrelle et al. [13] describe a method for extraction of vector-based[3] diagrams. Vector-based diagrams are defined by set of paths and symbols. Vector-based diagrams can be rendered at any scale while maintaining their appearance. In their method, they parse PDF command sequences comprising each document to get a sequence of objects. In the next step, they identify visible objects and render them in 1D or 2D spatial indexes. Finally, they use a recursive algorithm to extract diagrams and sub-diagrams by looking at the vertical and horizontal white space bands.

[3]Raster diagrams are comprised of individual pixels of color. JPG, PNG, GIF are examples of raster formats.

Table Detection Hao et al. [22] use visual features of the table areas for CNN while taking into account the non-visual information in the PDF documents. They select the initial areas with some heuristic rules. Faster R-CNN based methods described by the Schreiber et al. [47] and Gilani et al. [18] use techniques like data augmentation and transfer learning for getting a good performance. He et al. [24] proposed a multi-scale multi-task fully convolution neural network (FCN) for page segmentation. They also propose a conditional random field (CRF) on top of segmentation and contour detection to refine the segmentation output. TableSeer [50] automatically detects and extracts the tables in digital documents. BioText[4] search engine is a specialized search engine for biology documents and can extract the tables and figures.

Multiple Page Object Detection In their method Li et al. [31], they detect multiple types of page objects in a single framework. After extraction of the connected components, they segment the page into column regions and then line regions. The line regions are classified and clustered with the CRF based classification models. Once the regions are classified, they group the regions belonging to the same class and cluster to form a page object.

Gao et al. [14] conducted a competition on page object detection at ICDAR 2017 where the displayed math was one of the targets. One group used Single Shot Detector (SSD), one group used CNN and the rest of the groups who participated in the competition used faster R-CNN in their system as one

[4]http://biosearch.berkeley.edu

of the stages. In this competition, they observe that the recall of the systems which used faster R-CNN directly was decreased when the IOU threshold was increased. Hence, they mention that the precision of the faster R-CNN is not high enough for detection of the math expression.

Yi et al. [53] proposed a three-phase framework for page object detection. In the first phase, they use the custom-designed component-based region proposal method for document images for generating candidate regions. In the next phase, they give all the candidate regions as input to CNN to classify. In the third phase, they use a dynamic programming algorithm to redistribute the labels and confidence coefficients obtained from the CNN instead of using non-maximal suppression (NMS) used by earlier methods like SSD. We would like to try similar dynamic programming algorithm instead of NMS in our architecture in future.

2.1.1 Summary of Page Object Detection

Most of the page object detection methods described in this section use visual information from the page images and non-visual information (like character locations, character labels etc.) available from the PDF documents. These methods rely on the page layout, spacing, element heights, font, character locations, character lables etc. to find the candidate regions. Use of non-visual information from the PDF documents simplifies the detection task. Methods which do not require additional non-visual information from the PDF documents, use heuristic rules, XY-cut algorithm, projection-profile cutting to

find the candidate regions. Finally, based on the type of object these methods are trying to detect they use a set of heuristic rules or a classifier trained on the visual/non-visual information.

Tha main disadvantage of the methods that use non-visual information is that they can not be used for documents where only page images are available. Most of the scanned documents that are available in the PDF format do not contain the information required by these methods. Absence of such information requires us to use OCR before applying these methods.

2.2 Math Expression Detection

Lin et al. classify the math expression detection methods into three categories based on the features used [37]. These categories are character-based, image-based, and, layout-based. Character-based methods use OCR engine to recognize the characters and the characters which are not recognized by the OCR engines are considered as the candidates for the math expression elements. The second category of methods uses image segmentation techniques. Most of these methods require finding the segmentation threshold values. Setting the threshold values is very difficult, especially for the unknown documents. The layout-based methods detect the math expressions using features like line height, line spacing, alignment, etc. Many of the published methods use the combination of character features, layout features, and context features. We can also classify the math expression detection methods as rule-based or learning-based. Many traditional detection methods use

heuristic rules to find the math regions. Methods published recently use traditional machine learning and deep learning to detect math regions. Yet another way to classify the math detection methods is based on the type of input that the method expects. Many methods expect document images and character information, whereas few others work only with document images.

2.2.1 Rule Based Methods

Lee et al. [29] use run-length image representation and extract adjacent black pixels as one segment. They merge adjacent segments into one if the distance between the segments is less than the threshold value. Then they label each line either as text or expression based on the line height and line spacing above and below. After labeling lines, they extract connected components from the image and treat each connected component as a character. They recognize each character. Finally, they use heuristic rules to generate symbol relation trees that define math expressions.

Garain and Chaudhari did a manual statistical survey of over 10,000 document pages and found out the frequency of each mathematical character in the expression [4]. They used the information found in this survey to develop a method for the detection of the embedded mathematical expressions [16]. They scan each text line and decide if the line contains one of the 25 most frequent mathematical symbols. After finding the leftmost word which contains the mathematical symbols, they grow the region around the word on the left and the right based on a few rules to get the complete math area. For detection

of the displayed math expressions, they use two features. First, the white space around the math expressions. Second, the standard deviation of the left lowermost pixels of the symbols of the text line. They base this feature on the observation that for a math expression the leftmost pixels of each symbol is not on the same line, for text line it is on the same line. One disadvantage of their method for embedded math expression detection is that it requires symbol recognition, which adds complexity to the system. In contrast to Lee et al. [29] and Garain and Chaudhari[4], we do not manually design features. Our method learns visual features during training. Also, our method does not require character recognition.

Toumit et al. [49] introduce the concept of mathematical objects to characterize the characters that are not plain text. They define an elementary mathematical object as an elementary mathematical entity that cannot be broken up into smaller objects without losing its mathematical meaning (e.g. 'a', '+', 'lim'). They define multiple classes for mathematical objects: simple object (e.g. 'a', '+'), a composite object (e.g. 'i', '='), an implicit object (e.g. multiplication in '$2a$'), operator (e.g. '$\cap$'), comparator (e.g. '>', '='), spatial connector(e.g. '[]'). Based on the class of objects detected in the text line, they use context-specific rules for region growing. For example, for parentheses, they check symbols between left and right locations. In contrast to their method, our method does not require defining additional mathematical object categories or any context-specific region growing rules. It completely depends on the visual information available in the document image.

Kacem et al.[28] propose local and global segmentation for detecting math expressions. The global segmentation uses visual and layout features of the adjacent connected components to detect the displayed math expressions. Local segmentation uses a region growing approach for the detection of the embedded math expressions. They use characters that are known to be mathematical (e.g., operators) as seeds for growing regions based on few heuristic rules. Similar to Kacem et al.[28] our method uses features from different levels including the global context. In contrast to this method, we use the same framework for the detection of embedded and displayed math expression.

Drake and Baird [10] use the pruned Delaunay triangulation of a set of locations of black connected components as input and classify each vertex and each edge either as math or text. They applied this method for the detection of the displayed math expressions, not the embedded math expressions. Similar to Drake and Baird [10], we make use of black connected components. But in contrast to Drake and Baird [10], black connected components are not central to our method. We only use them in postprocessing (refer to Chapter 3, Section 3.4.1). Also, our method works for both embedded and displayed math expressions.

2.2.1.1 Summary of Rule Based Methods

Rule based methods use manually designed heuristic rules that rely on the occurrence of the mathematical operators or symbols. To check the occurrence of the mathematical operators or symbols they need character labels.

They either use character labels available from the PDF documents or use OCR to get character labels. Extraction of character labels from PDF documents or use of OCR is not 100% accurate that adds noise to the system. Moreover, designing heuristic rules for all types of mathematical operators or symbols is a difficult task as different symbols are used for same math operations by different groups. Also, rule based methods either do not detect embedded math expressions or use a different set of rules for embedded math expressions than displayed math expressions.

2.2.2 Traditional Machine Learning Based Methods

Traditional machine learning methods use information extracted from the documents for training a classifier like support vector machine (SVM) using manually designed features. These methods generally use different classifiers for detecting displayed and embedded math expressions.

Lin et al. [37] proposed a method for math detection, which is a four-step process that uses a combination of various features. In the first step, they extract the locations, bounding boxes, baselines, fonts, etc. and use them as the character and layout features in the following steps. They also process the math symbols that comprise multiple objects. For example, the vertical delimiter is made up of multiple vertical short line objects. They detect the named mathematical functions like sin, cos, etc. and the numbers. In the next step, they distinguish between the text lines and non-text lines. They find the displayed math expression in the non-text lines using geometric layout

features (for example, line-height), character features (for example, is it the part of named math function like sin), and context features (for example, whether the preceding and the following character is a math element). In the last step, they use another classifier that classifies the characters into math and non-math characters. They find embedded math expressions by merging the area of the characters that are tagged as math characters. They used SVM classification for both isolated math expression detection and character classification into math and non-math.

For characterization of embedded and displayed math expression, Chu and Liu [6] propose novel features based on centroid fluctuation information of non-homogeneous regions. They adopt the method proposed for sign detection in natural images for text localization and text segmentation. They trained the SVM classifier to check if a line was displayed math expression or not. To detect the embedded math expression, they segment the text lines that are detected not be displayed math expression into words. They merge all in-line connected component bounding boxes that are within the threshold to form a word. They assume that the space between characters is less than the space between the words. They extract handcrafted layout features like centroid fluctuation, the height of the word, etc. and train SVM classifier to check if a word belongs to an embedded math expression.

For text line segmentation, Lin et al. [35] proposed a method based which uses projection profile cutting (PPC) for getting original text lines. Given a list of text lines, they extract layout features (e.g. vertical distance),

content features (e.g. is there a fraction sign between this line and next line) and text line features of consecutive text lines (e.g. height, the width of the text line). They use a two-step merging process to merge consecutive text lines. In the first stage, they check if two consecutive text lines should be merged or not. In the second stage, they use another classifier trained on similar features to merge/split text lines in a multi-line math expression. They report results using different classifiers including SVMs.

2.2.2.1 Summary of Traditional Machine Learning Methods

All the traditional machine learning methods used SVM classification, and two different techniques to detect embedded and displayed math expressions. Methods proposed by Lin et al. [37] and Chu and Liu [6], and Lin et al. [35] have one common disadvantage. These methods use manually designed features for training two different classifiers. The method proposed by Lin et al. [37] has one more disadvantage. Their method is designed to work only with born-digital PDF documents. It needs additional information like locations, bounding boxes, baselines, fonts, etc. from the PDF document for detecting math expressions which makes it impossible to use where additional information is not available.

2.2.3 Deep Learning Based Methods

For born-digital PDF papers, Iwatsuki et al. [27] created a manually annotated dataset and applied conditional random field (CRF) for the math-

zone identification using both layout features (e.g. font types) and linguistic features (e.g. n-grams) extracted from PDF documents. For each word, they used three labels: the beginning of math expression, inside math expression, and the end of the math expression. Each word is annotated using conditional random fields (CRF). They concluded that the words and fonts are important for distinguishing math from the text. This method has limitations as it requires a specially annotated dataset that has each word annotated with either beginning, inside or end of the math expression label. Their method works only for PDF documents that have layout information.

Gao et al. [15] use a combination of CNN and RNN for the detection of the math areas. In the first step, they extract text, graph and image stream from the PDF document. Next, they perform top-down (based on XY cut algorithm) and bottom-up (based on connected component analysis) layout analysis to generate the candidate expression regions. They use candidate math expression regions as the input to the next step, where feature extraction networks extract the features from the candidates. Next, they assign each candidate a class. In the final step, they adjust and refine the incomplete math expression areas. Similar to their method, we use a CNN model (VGG16 [48]) in our network architecture for feature extraction. In contrast to their method, we do not depend on the layout analysis of the page.

2.2.3.1 Summary of Deep Learning Methods

Deep learning methods used for page object detection are deep convolutional neural networks (CNNs). These methods expect images and some of them expect additional page layout information. Many CNN based object detectors perform far better than traditional methods. Object detctors learn features automatically and thereby avoid the need of feature engineering. On the other hand, we need to carefully decide the values for the hyper-parameters.

2.2.4 Unsupervised Learning

All the math detection methods discussed so far are supervised learning methods. Wang et al. [52] propose an unsupervised learning algorithm that is based on the observation that in one PDF document math expresssions tend to use one particular style of font setting. To detect math expressions they start with detecting lines and columns using projection-profile cutting. They find posterior probability of each character being math or non-math using font setting information available from the PDF documents. They extract group of characters which they call none-separable character set (NSCS) and use character level posterior probability for applying Bayesian inference to decide if NSCS is math or non-math. They claim that their method is singificantly faster than the supervised learning algorithms.

2.2.5 Summary of Math Expression Detection

Similar to page object detection methods discussed in the Section 2.1, most math detection methods rely on the additional page layout information available from the born-digital PDF documents which makes them unusable when PDF documents are made up of scanned pages. Methods which work with images use OCR to recognize the characters to get character-level information. Many methods use projection-profile cutting and XY-cut algorithm to find the candidate regions. Moreover, these methods use different techniques for embedded math expressions and displayed math expressions. For math expression detection, both supervised and unsupervised learning methods have been proposed.

We wanted to develop a method which works only from page images without use of any additional information or any heuristic rules and detects both embedded and dsplayed math expressions in one framework. Deep learning based object detection methods need only images and target regions for training. We apply one such method for detecting both embedded and displayed math expressions in one framwork.

In the next section, we summarize different object detection methods and our motivation behind using Single Shot MultiBox Detector (SSD) [38] in our architecture (described in Chapter 4).

2.3 Object Detection Using Deep Learning

First deep learning algorithm which achieved significantly better results on the object detection task was R-CNN [20] (region proposal with CNN). They used selective search algorithm [51] to generate $\approx 2k$ region proposals per input image. They wrap these region proposals into squares and use VGG16 [48] (refer to Section 2.3.3) to extract features. They use extracted features to train SVM for classification of objects. Their algorithm predicts the class for each region proposal and offsets for adjusting the bounding box of the proposal. There are two main disadvantages to this method. First, they use $\approx 2k$ region proposals per image that slow down the network. Second, the selective search algorithm is fixed. It is not a learning algorithm.

Girshick [46] addressed problems with the R-CNN with Fast R-CNN. Unlike R-CNN that feeds $\approx 2k$ regions to CNN per image, only the original input image is fed to CNN to extract feature maps. They use feature maps for region proposals and fully connected layers at the end of the network for feature extraction. Finally, they use extracted features to predict the class and bounding box offset values. The region proposal from feature maps uses selective search algorithm. Selective search is the bottleneck in the Fast R-CNN. As mentioned before, selective search is a fixed algorithm and not learned. Faster R-CNN [46] uses feature maps for region proposals just like Fast R-CNN. Unlike Fast R-CNN, Faster R-CNN uses a different network called a region-proposal network for proposing regions. This network learns to propose regions while training. Faster R-CNN removes the bottleneck introduced

because of selective search.

In contrast to R-CNN, Fast R-CNN, and Faster R-CNN that use region proposals, YOLO [43] is a single-stage network for object detection. It is significantly faster than the methods that require region proposals. They divide the input image into a grid. Each cell in the grid has an associated set of bounding boxes. They train a network that outputs class confidences and offsets for each bounding box. Bounding boxes that have class confidences higher than the threshold value are selected to locate the object within the image. Improved versions of the YOLO architectures like YOLO9000[44] and YOLOv3[45] have been proposed.

Similar to YOLO, Single Shot MultiBox Detector (SSD) [38] is a single-stage network. SSD is faster and more accurate than YOLO [38]. Similar to YOLO, SSD divide the input image into grid and each cell in the grid has an associated set of bounding boxes. Unlike YOLO, SSD uses multiple grids with different sizes instead of just one grid size. Network learns to predict the offsets for each bounding box associated with grid cells in different grids. Different grid sizes allow network to divide the responsibility of detecting objects at different scales. Just like R-CNN, SSD uses VGG16 architecture. Unlike R-CNN, Fast R-CNN, and, Faster R-CNN SSD does not require selective search, region proposals, or multi-stage networks. Even though SSD is more accurate than YOLO, performance of SSD dips for small object detection.

In our method for math detection, we use SSD architecture with sliding windows to generate smaller sub-images to zoom-in on the math regions to

Figure 2.1: Example of NMS. The left image shows multiple detections for one math expression and right image shows the output generated after NMS.

improve detection rates.

2.3.1 Non-Maximal Suppression

Object detection algorithms use non-maximal suppression (NMS) which is a hand crafted postprocessing step that is not present while training. NMS is a iterative, greedy, locally optimal strategy which keeps only one detection per group (ideally obtaining only one detection per object) from relatively dense detection outputs near the correct location of the object generated by object detection methods while inferring. For each iteration of NMS a group is defined by selecting current maximum confident detection and other detections overlapping with the maximum confident detection. For each group a maximum confident detection is selected and other detections with IOU greater than threshold with maximum confident detection are suppressed. NMS does not modify any of the input bounding boxes, it just selects best bounding boxes based on the confidence and IOU threshold (refer to Chapter 3). Figure 2.1 shows an example of NMS where only one box is selected from a group of detections. Other detections are suppressed because they have higher IOU with selected detection than the threshold.

2.3.2 Discrete Convolution

Discrete convolution is an operation on functions with real-valued arguments. The first argument to convolution is referred to as the input and the second argument is referred to as the kernel, and the output is referred to as the feature map. We can use equation 2.1 to calculate value of location (i, j) of the feature map for two dimensional input image I of size $x \times y$ and two dimensional kernel K [21].

$$S(i,j) = \sum_{x} \sum_{y} I(x,y)K(i-x, j-y) \tag{2.1}$$

2.3.3 VGG16

Figure 2.2 shows the VGG16 architecture. VGG16 is a 16 layer deep convolutional neural network with 13 convolutional (CONV) layers and 3 fully connected (FC) layers. VGG16 uses 2×2 maxpool layers with stride 2 and RELU activation. We explain the FC layer and CONV layer with the help of an example in Figure 2.3 and Figure 2.4, respectively. FC layer in Figure 2.3 flattens $2 \times 2 \times 3$ input image to 1D vector of size 12. It maintains 2 vectors of size 12 for performing dot products with the input 1D vector and generates vector of size 2 as output. This FC layer needs to learn 24 parameters to convert input of size $2 \times 2 \times 3$ to output of size 2. Figure 2.4 shows convolution process. CONV layer in Figure 2.4 uses two filters of size $2 \times 2 \times 3$ (same size as input image) for dot product and generated output of size $1 \times 1 \times 2$.

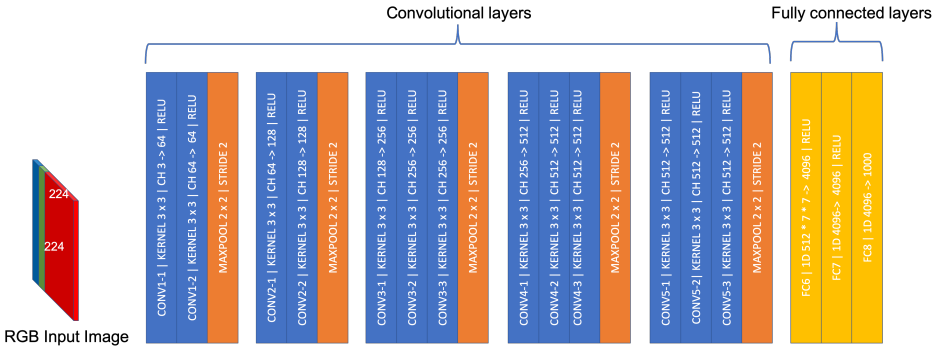


Figure 2.2: VGG16 architecture. We show all the convolutional (CONV) layers in blue, max pool layers in orange and fully connected (FC) layers in yellow. All CONV layers use padding 1, kernel size 3×3, and, ReLU activation. CH $i \rightarrow j$ represents that the convolution layer expects i channel input and generate j channel output. VGG16 expects image of size $224 \times 224 \times 3$ as input. This architecture was evaluated on ImageNet [9] dataset which has 1000 classes.

CONV layer uses discrete convolution operation explained in Section 2.3.2. Just like FC layer, CONV layer needs to learn 24 parameters to convert input of $2 \times 2 \times 3$ to output of size 2. This property of FC layer and CONV layer is used to convert FC to CONV layer. Conversion of FC to CONV layer is explained in the next section.

2.3.4 Conversion of FC Layers to CONV Layers

VGG16 is not a fully convolutional network as it uses 3 fully connected layers. But, it is possible to convert a network to a fully convolutional network by converting fully connected layers to convolutional layers. Consider fully connected layer from Figure 2.3 and convolutional layer from Figure 2.4. The input and output of these two layers are the same. $Y1$ and $Y2$ are the dot

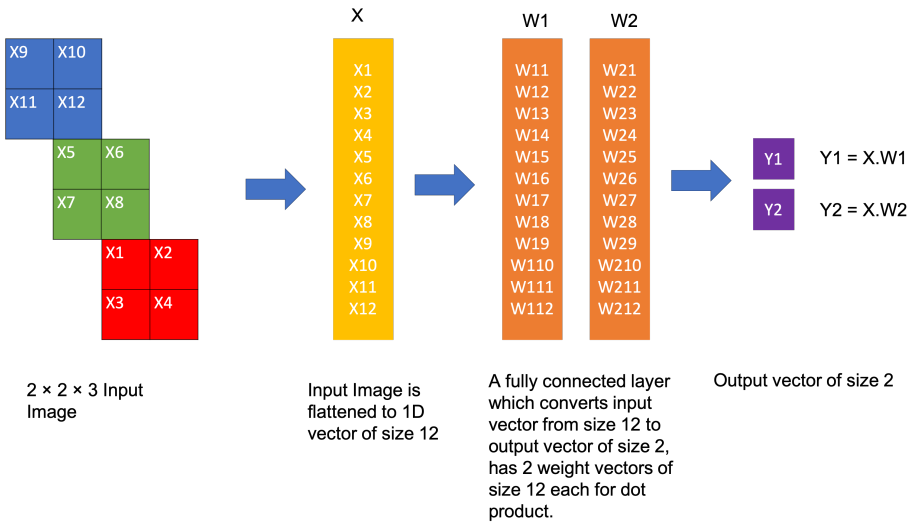


Figure 2.3: An example of a fully connected layer which receives an input of size $2 \times 2 \times 3$ and generates 1D output of size 2.

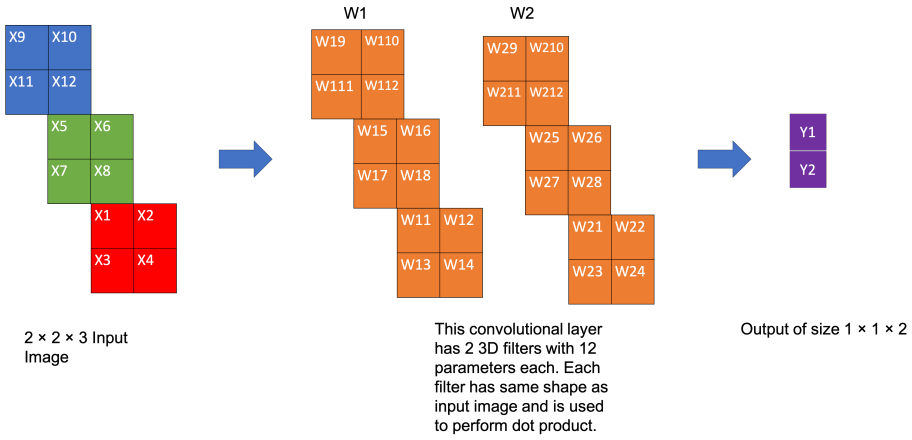


Figure 2.4: An example of a convolutional layer which receives an input of size $2 \times 2 \times 3$ and generates output of size $1 \times 1 \times 2$.

products of the input image and layer parameters (like weights or filter values). It means a fully connected layer with input image of size $H \times W \times C$ and output of size N is equivalent to a convolutional layer with the same input and kernel size of $H \times W$ with N output channels, provided the same set of parameters are used in convolutional filter and fully connected layer weights. Hence, a fully connected layer can be converted to a convolutional layer by reshaping the parameters used in the fully connected layer. In VGG16, FC6 receives an input of size $7 \times 7 \times 512$ and generated output of size 4096. Hence, we can convert FC6 to CONV6 with total 4096 filters of size $7 \times 7 \times 512$. Similarly, other FC layers can be converted to CONV layers (FC7 to CONV7 and FC8 to CONV8).

2.3.5 Dilated Convolution

The VGG16 needs to learn 102,760,448 ($= 7 \times 7 \times 512 \times 4096$) parameters for CONV6 layer and 16,777,216 ($= 1 \times 1 \times 4096 \times 4096$) paramters for CONV7 layer. In practice, many state-of-the-art architectures which use VGG16 for feature extraction use lesser number of small-sized filters while converting FC layers to CONV layers [38, 5]. For example, instead of using 7×7 kernel, 3×3 kernel is used by picking every third parameter along a particular dimension. Picking every m^{th} parameter along a particular dimension is known as decimation. It reduces the number of parameters to learn. But we need a 7×7 kernel for CONV6 to be equivalent to FC6. So, a process called dilated convolution is used instead of discrete convolution (refer to

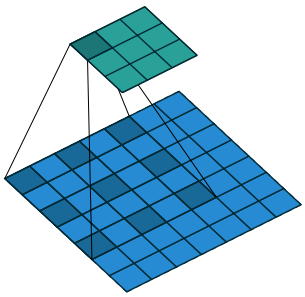
2.3.2). The term dilated convolution was first used by Yu and Koltun [54]. We call the fully convolutional VGG16 with the lesser number of small-sized filters as FC-reduced VGG16.

Figure 2.5 shows the dilated convolution process with dilation factor of 2. Dilation factor controls the spacing between the kernel points. The convolution performed in this way is also known as the à trous algorithm. Different convolution operations for deep learning are described by Dumoulin et al. [11] and open source code[5] provided by Dumoulin et al. was used for generating Figure 2.5. We use FC-reduced VGG16 with à trous algorithm in our architecture. We describe the parameters we used for conversion of FC6 and FC7 in Chapter 4.

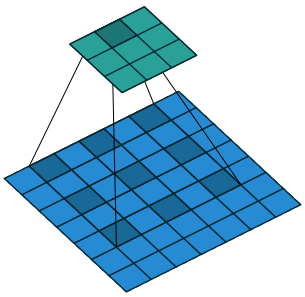
2.4 Summary

In this Chapter, we mainly summarized math expression detection methods and some methods for other page object detection. Page obejct detection methods whether rule based or learning based, supervised or unsupervised use both visual (page image) and non-visual information (character labels, locations, font size, font type etc.) available from the PDF documents. Also, methods that do not use non-visual information from the PDF documents use projection profile cutting or XY-cut algorithms for finding candidate regions. Additionally, they require character recognition to find character labels. More-

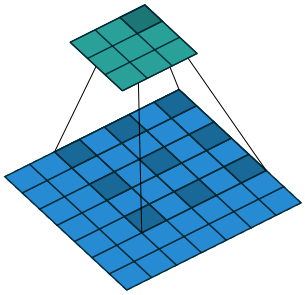
[5]Figure 2.5 was generated using the open-source code from `https://github.com/vdumoulin/conv_arithmetic`



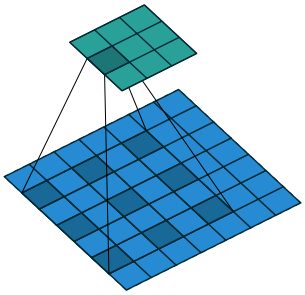
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure 2.5: The dilated convolution operation. The blue matrix is input and the green matrix is the output. The dark blue locations are the locations where the filter is applied to get the value at the dark green location. The filter is not applied to consecutive locations in the input, but it is applied by skipping some locations. The dilated factor decides the number of locations to skip.

over, most of the methods use different techniques to find the embedded and displayed math expressions.

Method designed for born-digital PDF documents that use additional information available from the PDF, can not be used for the PDF documents which do not contain this additional information (e.g. PDF documents generated using scanned document pages). Use of projection profile cutting, XY-cut algorithms, or, character recognition adds complexity to the system. Different methods for detection of embedded and displayed math expressions requires us to train two different classifiers, decide two different thresholds, or, design two different set of rules.

In Chapter 4, we propose *ScanSSD* architecture that needs only document page images as input and detects both embedded and displayed math expressions in a single framework without use of any additional non-visual information like character locations, character labels, etc or heuristic rules, projection profile cutting, or, XY-cut algorithms. Our architecture is based on object detection method called Single Shot MultiBox Detector (SSD) [38]. We use sliding windows to generate sub-images of whole page image for detecting small math expressions and extensive data augmentation for generating data for training a deep network. Unlike R-CNN networks, SSD [38] does not require selective search, region proposals, or multi-stage networks. Just like YOLO, SSD is a single-stage object detector but, SSD is faster and more accurate than YOLO [38]. Chapter 4 describes our architecture in depth.

Before introducing our architecture in Chapter 4, in Chapter 3 we for-

mally define the math detection task and the steps we took for creating a new dataset for math expression detection.

Chapter 3

Math Detection Task and Dataset Preparation

In this chapter, we define the math detection task and describe how we prepared the dataset for this task. Our dataset was used for the Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection (CROHME + TFD 2019) at the 15th International Conference on Document Analysis and Recognition (ICDAR 2019) [40].

3.1 Math Detection Task

Our aim is to detect math expressions in typeset documents. We define math expression detection task as follows: Given an input image I, generate a set of math expression bounding boxes D_I where each bounding box $d \in D_I$ is given by top-left and bottom-right co-ordinates of the box $\{l, t, r, b\}$ (l=left, t=top, r=right, b=bottom). If G_I is a set of ground truth bounding boxes $g \in G_I$, we can define an injective (or one-to-one) function [1] $f : D \rightarrow G$ that maps each detection with a ground truth with which the detection has the highest IOU score. The IOU score is defined by Equation 3.1. f is defined in

[1]The function f is said to be injective provided that for all a and b in input domain X, whenever $f(a) = f(b)$, then $a = b$; i.e, f(a) = f(b) implies a = b.

Algorithm 1.

$$IOU(S_i, S_j) = \frac{\mid S_i \cap S_j \mid}{\mid S_i \cup S_j \mid} \tag{3.1}$$

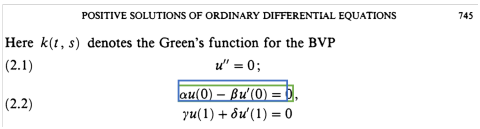
Algorithm 1: Definition of f

Result: One to one mapping from detections D_I to ground truths G_I

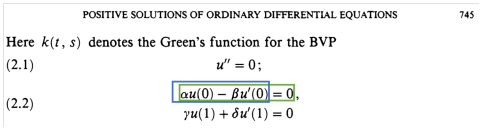
$avail(d) = G_I, \forall d \in D_I$;
$f(d) = \text{argmax}_g \, IOU(d, g) \forall g \in G_I, d \in D_I$;
while *all detections are mapped to ground truth* **do**
  if $f(d_i) = f(d_j)$ *and* $i \neq j$ **then**
    if $IOU(d_i, f(d_i)) \geq IOU(d_j, f(d_j))$ **then**
      $avail(d_j) = avail(d_j) - f(d_i)$;
      $f(d_j) = \text{argmax}_g \, IOU(d_j, g) \forall g \in avail(d_j)$;
    else
      $avail(d_i) = avail(d_i) - f(d_j)$;
      $f(d_i) = \text{argmax}_g \, IOU(d_i, g) \forall g \in avail(d_i)$;
    end
  end
end

where, $S_i \cap S_j$ denotes the set of shared pixels between S_i and S_j and $S_i \cup S_j$ denotes the set of all pixels in S_i and S_j.

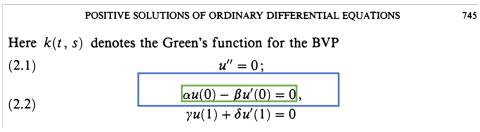
For image I, if f maps $d \in D_I$ to $g \in G_I$, we say d matches g if $IOU(d, g) \geq T$, where $T \in [0, 1]$. As T increases, the matching criteria becomes more strict. Figure 3.1 shows matched and unmatched detected bounding box for $T = 0.75$ while illustrating the nature of IOU score. We report our results for $T \in \{0.5, 0.75\}$ in Chapter 5.



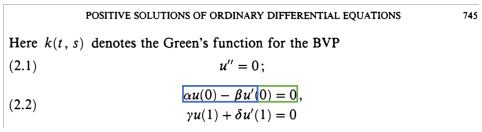
(a) Detection and ground truth has $IOU \geq 0.75$



(b) Detection and ground truth has $IOU < 0.75$



(c) Detection and ground truth has $IOU < 0.5$



(d) Detection and ground truth has $0.5 \leq IOU \leq 0.75$

Figure 3.1: The figure shows the nature of IOU score. The detected region is shown in blue and the ground truth is shown in green. If $T = 0.75$, the detection from Figure 3.1b is not considered as a valid match as $IOU \leq T$. Even though the detection from the Figure 3.1c covers the entire math expression, it is not considered as a valid match as $IOU \leq T$.

The function f, maps one detection bounding box to exactly one ground truth box. We represent each match as an ordered pair (d, g) and set of all such matches as M_I. We define precision, recall and F1-score to evaluate the performance of the math detector in Equation 3.2, Equation 3.3, and Equation 3.4, respectively.

1. Precision (P): The fraction of matched bounding boxes among all detected bounding boxes.

$$P = \frac{\mid M_I \mid}{\mid D_I \mid} \tag{3.2}$$

2. Recall (R): The fraction of matched detected bounding boxes over the total amount of ground truth bounding boxes.

$$R = \frac{\mid M_I \mid}{\mid G_I \mid} \tag{3.3}$$

3. F1-score (F): Considers the trade-off between precision and recall, which is the harmonic mean of precision and recall.

$$F = \frac{2 \times P \times R}{P + R} \tag{3.4}$$

3.2 Math Detection in PDF Documents

Math detection in born-digital PDF is a simplified version of math detection in images. In born-digital PDF documents, we have information related to character locations, character labels, font name, font size, etc. The availability of additional information makes math detection in PDF documents a simpler task than math detection in document images. A math detection system which detects math from input images can be easily used for born-digital PDF documents. Each page from the PDF can be rendered as an image and can be used as an input to the math detection system. Hence, we concentrate on the math detection in document images.

3.3 GTDB Datasets

Table 3.1: GTDB datasets statistics

	GTDB-1	**GTDB-2**
Number of articles	32	16
Number of pages	544	343
Number of math symbols	162,406	115,433
Number of text symbols	646,714	507,412

GTDB datasets are publicly available[2] that provide annotations for document page images collected from scientific journals and textbooks. The GTDB-1 dataset contains 32 articles[3] on mathematics and GTDB-2 dataset contains 16 articles. Table 3.1 shows the details of these two datasets. Diverse font faces and mathematical notation styles are used in these articles. They provide ground truth annotations in CSV files. GTDB datasets do not provide the PDF files directly, but they provide the links to the files.

Example ground truth annotations shown in Figure 3.2. The first line is a file header showing the format version. Record starting with 'Sheet' denote the beginning of a new page. It has four entries: Sheet, page id, filename, -1. Record stating with Text, Line, or Image represents a component. It has six entries: Text, Line, or Image, component id, bounding box co-ordinates. Record starting with Chardata denotes a character. It has ten entries: Chardata, character id, bounding box co-ordinates, text mode (0: text, 1: math), relationship

[2]Available from `https://github.com/uchidalab/GTDB-Dataset`
[3]GTDB repository mentions 31, but there are actually 32 articles in GTDB-1

label, parent character id, ocr code. The link label is one of the following: horizontal(0), right-superscript(1), right-subscript(2), left-superscript(3), left-subscript(4), upper(5), lower(6), the first character in expressions or ordinary texts (-1).

We show an example of the output generated by our visualization tool for the ground truth annotations in Figure 3.3. The ground truth permits comparison of detected regions at the raw image or character levels. GTDB datasets do not include bounding boxes for the math regions. Moreover, the annotations provided in the dataset do not match perfectly with the page regions. So, we performed data cleanup and aligned the ground truth with the page objects it corresponds to. We used a cleaned version of the dataset to generate the bounding boxes for math regions for our experiments. We explain our data cleanup process and generation of ground truth for the math expression detection in the next section.

```
Infty GT-Data Format Ver.1.1
Sheet,1,ASENS_1997_367.png,-1 Text,1,202,114,1202,309
Line,1,259,114,1049,221 Chardata,1,259,162,297,206,0,-1,-1,4141
Chardata,2,304,178,333,207,0,-1,1,416E
Chardata,3,338,177,367,207,0,-1,2,416E
Chardata,4,373,201,379,206,0,-1,3,142E
Chardata,5,379,114,413,221,0,-1,4,0E81
Chardata,9,489,177,514,206,0,-1,8,4165
Chardata,10,519,178,548,207,0,-1,9,416E
Chardata,11,1600,1585,1643,1624,1,0,10,4178
Image,1,1482,634,1980,968
```

Figure 3.2: Example of ground truth annotations from GTDB dataset

Figure 3.3: Visualization of ground truth annotations from GTDB dataset. The red boxes are for text regions, magenta boxes are for images, blue boxes are for textlines, green highlights are for text characters, and, yellow highlights are for math characters. Even though the diagram in the magenta box contains math regions, they are not annotated in the GTDB dataset.

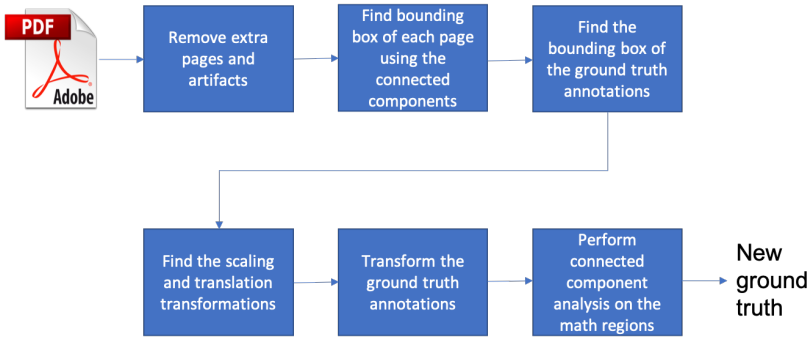


Figure 3.4: GTDB data cleanup process

3.4 Dataset Preparation for Math Detection

We used the links provided by the GTDB datasets to download the files manually. Some of the links were broken, and we had to use a different source to download the document. We could not find a source to download two documents[4] in GTDB-1 dataset, hence we used 30 documents from this dataset. The annotations given for all the documents from GTDB dataset and many documents from the GTDB-2 dataset were not in perfect alignment with the page structure. Hence, we started data cleanup. Figure 3.4 outlines the steps in our data cleanup process.

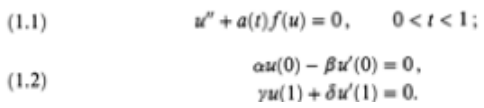
We use pdf to image conversion tool[5] to render 600 dpi images for each downloaded pdf file. Most of the PDF files in this dataset are scanned documents. We observed two types of artifacts in the PDFs, first introduced while scanning and second introduced while downloading from the internet. To remove artifacts (like small black regions) introduced while scanning, we apply median filtering. Median filtering is a non-linear digital filtering technique that is known for removing salt-and-pepper noise. Some PDF documents have vendor information added as a footnote. We manually painted that information white. Next, we find the bounding box of each connected component in the image. Then we find a page bounding box B_{pg} which contains all the bounding boxes for connected components found in the previous step. B_{pg} is a matrix in homogeneous co-ordinates. From the ground truth annotations, we

[4]We could not find AM_chapter12 and AnnMS_1971_157_173.
[5]python3 package: pdf2image version 1.5.4

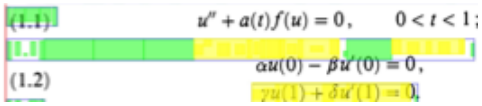
find another page bounding box B_{gt} for the same page represented as a mtrix in a homogeneous co-ordinates. Ideally, $B_{pg} = B_{gt}$. But, as we mentioned earlier the annotations from the GTDB dataset do not align with the page objects. Hence, we find scaling and translation transformations T such that $T = B_{gt}B_{pg}^{-1}$. We apply the same set of transformations T to each bounding box in the original ground truth to generate the new ground truth for the images. We use new ground truth to generate the bounding boxes for the math expressions.

GTDB datasets do not provide bounding boxes for math expressions, but they provide the character level annotations. We use the character level information to find the bounding boxes for the math regions. We perform connected component based postprocessing for the math regions as described in Section 3.4.1. In this step, we fit all the connected components inside or intersecting the current math bounding box perfectly by growing or shrinking the bounding box. We used the obtained annotations for our dataset. Example of the initial ground truth and output of the cleanup process is shown in Figure 3.5. The dataset created by us was used for CROHME + TFD 2019 competition at the ICDAR 2019 conference. We refer to the dataset used in CROHME + TFD 2019 as TFD-ICDAR 2019 dataset. For ICDAR 2019, we provided math bounding boxes and character bounding boxes. We did not provide line, image or text region bounding boxes.

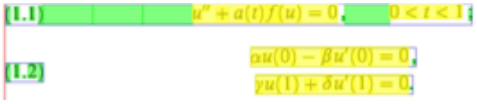
After data cleanup, we split the dataset in train data and test data. The train and test dataset statistics are given in the Table 3.2. For math or text in



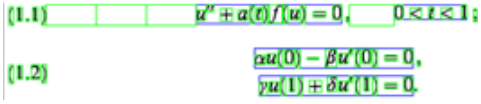
(a) The input image to the data cleanup process



(b) Original ground truth annotations as an input to the data cleanup process



(c) The output ground truth annotations after data cleanup



(d) The ground truth annotations from TFD-ICDAR 2019

Figure 3.5: Inputs and outputs of the data cleanup process. We highlight text characters in green and math characters in yellow. For TFD-ICDAR 2019 dataset, we maintained only character information and math region bounding boxes.

Table 3.2: ICDAR2019 dataset statistics

Statistics	TFD-ICDAR 2019		TFD-ICDAR 2019 Version 2	
	Train	Test	Train	Test
Number of documents	36	10	36	10
Number of pages	569	236	569	236
Single character math regions	7506	2556	7506	2556
Multi character math regions	18890	9329	18947	9350
Total math regions	26396	11885	26453	11906

the figures GTDB datasets have no annotations and hence the TFD-ICDAR 2019 dataset does not have annotations for the same.

3.4.1 Postprocessing

Figure 3.6 shows the example of postprocessing of the boxes. We expand or shrink the detected boxes so they perfectly fit the connected components inside them. This makes sure that there is no unnecessary white space

Figure 3.6: Example of postprocessing of a box so that it perfectly fits the connected components inside it or connected components it is touching

inside the detected boxes. If the detected box intersects a connected component, we expand the detected box to include that connected components in it, with the goal of capturing entire characters belonging to the detection region.

Character bounding boxes and OCR codes are available in the GTDB dataset, which can be used in the future to improve the detector. We discuss different methods in which character data can be used in Chapter 6.

3.4.2 Dataset Release

Our dataset (TFD-ICDAR 2019 dataset) and evaluation tools are publicly available at `https://github.com/MaliParag/TFD-ICDAR2019`. With the dataset, we also provide additional scripts to download dataset PDF files, render page images from PDFs, and visualize annotations. For the CROHME + TFD 2019 competition we observed that participating teams faced problems while using annotations, as the 600 dpi images rendered on their machines had slightly different sizes. It is possible that participating teams used different image rendering tools or different version of the same image rendering tools and got different image sizes. Also, we observed that there are few pages mostly in the test dataset where few math bounding boxes are not correct or

missing.

We fix the problems with TFD-ICDAR 2019 in version 2 of the dataset. In version 2, we provide the size of each image used while generating the dataset annotations to avoid image size related issues. We did not use normalized coordinates while providing the annotations to avoid potential aliasing issues because of the floating-point arithmetic. In version 2, we added missing boxes and fixed the invalid boxes. Statistics of the version 2 of the dataset are given in Table 3.2.

All the results presented in this thesis use the TFD-ICDAR 2019 dataset to benchmark against other systems. We did not use version 2 of the dataset as it was developed after the experiments were done.

Chapter 4

Methodology

We aim to detect both embedded math regions and displayed math regions using only one framework. We can treat math expression detection as an object detection task. Among the CNN-based object detectors, SSD is faster than most of the models and has competitive accuracy to models that use an additional object proposal step like R-CNN and Faster R-CNN [38][26]. Liao et al.[32] with their TextBoxes architecture have shown that SSD with few modifications can detect wide regions. Math regions can be very wide and with similar modifications used in TextBoxes, SSD should be able to detect them. Hence, we used SSD architecture for math expression detection. In this chapter, we describe our system architecture and how it addresses the challenges in math expression detection in typeset documents.

In the next section, we give overview of the ScanSSD architecture.

4.1 Overview

Figure 4.1 shows overview of the ScanSSD architecture. First, we use sliding windows to obtain image patches while maintaining overlap between the patches. We modify the ground truth bounding boxes to represent math

regions in the extracted patches using sliding windows and cut the bounding boxes if the sliding window method splits a math region into multiple patches. Next, we use patches generated using sliding windows as input to SSD. In the next stage, we get confidence and bounding boxes at the patch level as the output of the SSD. In the final stage of SSD, we apply non-maximal suppression over detected bounding boxes. NMS is a greedy, locally optimal strategy which keeps only one detection per group (ideally keeping only one detection per object) from relatively dense detection outputs near the correct location of the object generated by object detection methods.

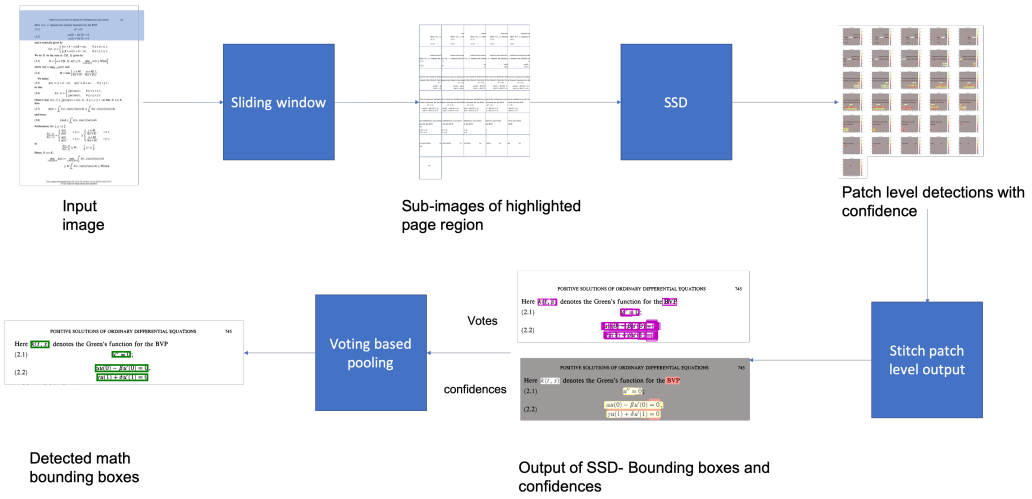


Figure 4.1: Overview of the ScanSSD architecture. The figure shows the steps in the detection of math expressions in the highlighted area of an input image.

We stitch the output of SSD, generated at the patch level, to get the bounding boxes at the page level. Then we use a pooling method to obtain the final detection results for the page. We tried different pooling methods and we

describe each pooling method in section 4.4. In pixel-level pooling methods, we use detected boxes and their confidences to assign a pixel level score and get a score map. We apply a threshold on the score map and convert it into a binary mask. We assign a value of 1 to the pixels that have a higher score than the threshold and a value of 0 to others. In the next step, we extract the bounding boxes of the connected components of pixels with a value of 1. Finally, we expand or contract the detected bounding box, so it perfectly fits the connected components inside it or connected components it is touching.

4.2 Methods

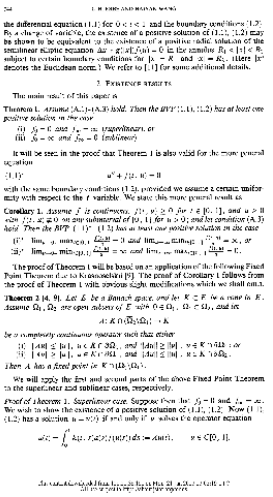
In this section, we discuss all the methods used in our network architecture like sliding windows, data augmentation, detection regions and their visualization, multi-scale feature maps and their importance, matching strategy for default boxes, and, use of convolution kernels.

4.2.1 Sliding Windows

In this section, we describe the sliding window method we used to generate sub-images from the original page images. We observed that converting the 600 dpi images from the GTDB dataset to 300×300 or 512×512 loses too much information. Figure 4.2a and 4.2b show the sample images of size 300×300 and 512×512 respectively. We generated these images using linear interpolation. We can see that these images do not have enough resolution for the detection of math expressions. We trained the SSD model using low-

resolution images and found that the network only detects bold characters from the input image as math regions.

(a) Resizing 600 dpi image to size 300×300 image



(b) Resizing 600 dpi image to size 512 × 512 image

Figure 4.2: The results of conversion of 600 dpi images to lower sizes using linear interpolation. Other interpolation methods can be used but there still will be loss in information. We could not use larger images with sizes like 1024×1024 and above for training because limited GPU memory size.

TFD-ICDAR 2019 dataset provides links to the PDF files. We use PDF to image conversion tool[1] to render 600 dpi images for each downloaded PDF file. We found that all the PDF files were generated by scanning the document pages and no protocol was followed regarding PDF page size while creating PDF files from the scanned images. Hence, each generated image is of different size. We slide a window of size 1200×1200 with a vertical and horizontal shift

[1]python3 package: pdf2image version 1.5.4

of 120 pixels i.e. stride of 120 pixels (or 10%), over the image to obtain sub-images of size 1200×1200. 1200×1200 window makes math expressions big enough for SSD to detect them effectively (As discussed in Chapter 2, Section 2.3 that SSD is better at detecting bigger objects rather than smaller objects). Figure 4.3 shows the example of the sub-images generated using our sliding windows. While using the sliding windows on the training data, we also update the ground truth math bounding boxes to represent the math expressions in these sub-images, this involves cutting the wide ground truth math bounding boxes into multiple smaller bounding boxes, scaling, and translation. For SSD300 we resize 1200×1200 sub-images to 300×300 and for SSD512 we resize them to 512×512. We explain the SSD architecture later in Section 4.3. Next, we discuss the advantages and disadvantages of the sliding window method.

Advantages There are four main advantages of using the sliding window method. The first advantage of the sliding window method is data augmentation. The number of images available for training is only 569, which is a tiny number for training a deep neural network. Using 1200×1200 window size and a stride of 120 pixels, we get 656,717 sub-images with math regions. Second, we limit the loss of information by converting sub-image to 300×300 or 512×512 instead of directly converting original image to 300×300 or 512×512. Third, as we maintain the overlap between the windows, the network can see the same region multiple times and get multiple chances to detect a math region. This

can increase the overall recall of the network because regions appear in more parts of a detection window. Finally, Liu et al. [38] mention that SSD is not that good at detecting small objects. Math expressions with just one or two characters can be a small region. It would be very difficult for SSD to detect these regions if we use the whole page as an input to the SSD. Creation of smaller sub-images increases the size of smaller math regions, which makes it easier for SSD to detect them. Therefore, we use the sliding window method to generate smaller sub-images for this task.

Disadvantages There are a few disadvantages to the sliding window method. Windowing cuts the math regions if they do not fit the window. This means that one big math expression might be split into multiple sub-images. For example, refer to Figure 4.3 where a math expression (2.2) in the highlighted image is split into multiple sub-images. Splitting math expressions into multiple sub-images makes it impossible to train the SSD network to detect big math expression as an object, so we train the network to detect the small parts of big math expressions as objects in different sub-images. Furthermore, we need a method to stitch the results from all the sub-images for a page together to get final detection results for the original page. We discuss how we address these problems using pooling methods in section 4.4.

In the Section 4.2.2, we describe other data augmentation methods that we used while training the models.

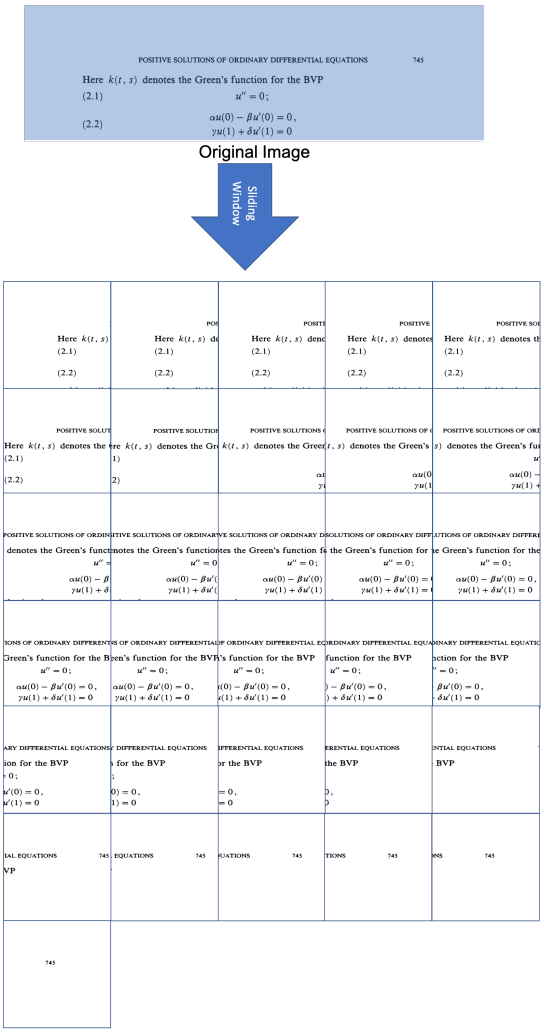


Figure 4.3: Sub-images generated for the highlighted region with 10% stride. For page image of size 6000×4800 one such highlighted region generates 31 sub-images. There are 41 such highlighted regions. It results in 1271 sub-images. Page images of different sizes will generate a different number of sub-images.

4.2.2 Data Augmentation

Data augmentation explained in this section is performed when a batch of images is created for training. One image goes through a series of operations explained in this section to generate a new image. We use photometric distortion and random sample crop to make the model robust to various object sizes and shapes just like original SSD model [38]. In photometric distortion, we convert RGB image to HSV color space and assign random hue and saturation to the image before converting it back to RGB. Next, we apply a random contrast, random brightness and random lighting noise (i.e., we shuffle image channels randomly) to the image. In the case of the random crop, we randomly sample each training image using one of the following options:

- Use original input image as it is
- Sample a patch so that the minimum Jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7, or 0.9.
- Randomly sample a patch

As there are 7 options for the random crop, probability that an option is selected is $(1/7)^{th}$. Randomly sampled patches have two contraints. First, width and height of generated patch should be at least 30% of the original width and height, respectively. Second, the aspect ratio of the generated patch should be between 0.5 and 2.

We use the zoom out operation proposed in the original SSD model that creates more small object examples. In this operation, we place the input image on a canvas of size up to 16× the size of the image before applying the random crop. Unlike original SSD, we do not perform mirroring of the image because we do not think it will help improve the results for the math expression detection.

In this data augmentation process one input image generates only one output image in one epoch. But, the same input image might generate different output image in next epoch as we randomly select values for the series of operations, i.e., we might use different values in different epochs for hue, saturation, brightness, patch size, etc. We do not increase the available number of training images in one epoch, but the number of distinct images network has seen during training increase as we train for multiple epochs.

4.2.3 Multi-scale Feature Maps for Detection

In original SSD architecture [38], after the base network VGG16, additional feature layers are added. These layers decrease in size and allow predictions to be made at multiple scales. Feature maps are obtained after applying convolution on each feature layer. The convolution process is explained in Section 4.2.6. Lower layer feature maps (like a featue map from Conv4_3) that appear earlier in the network capture more fine-grained details and can improve the segmentation quality [39][23]. By utilizing the feature layers of different scales in the same network we can mimic the effect of processing the

image at different sizes and combining the results afterward. Just like Liu et. al [38], for SSD300 we use feature maps extracted from Conv4_3, FC7, Conv8_2, Conv9_2, Conv10_2, and, Conv11_2 and for SSD512 we use feature maps extracted from Conv4_3, FC7, Conv8_2, Conv9_2, Conv10_2, Conv11_2, and, Conv12_2. We use total 6 feature maps in case of SSD300 and 7 feature maps in case of SSD512. One of the feature maps is extracted from the VGG16 as shown in Figure 4.7. VGG16 is a deep convolutional neural network used for feature extraction. Refer to Chapter 2, Section 2.3.3 for more details on VGG16. Size of each feature layer is written on each feature layer in Figure 4.7.

4.2.4 Detection Regions

In SSD, we define potential locations for the objects of interest in the feature maps. We call these initial potential locations for objects of interest as default boxes. While training network predicts four offset values $\Delta(cx, cy, h, w)$, two for the center of the default box (cx, cy) and one for the height (h) and width (w) each. Using predicted offset values, we get a detection hypothesis per default box. We show the default boxes for feature maps from SSD512 architecture in the Figure 4.4. These default boxes cover every area of the image and should be able to detect an object at any location. The lower-level feature maps like 64×64 have small default boxes and learn to detect small objects. The higher-level default boxes like 4×4 use big default boxes and learn to detect relatively big objects.

Default Box Tiling Our default box tiling method is exactly the same as the original SSD model [38]. We place default boxes so that different feature maps learn to be more responsive to objects of different scales. If there are m feature maps for the prediction, we calculate the scales of the default boxes using Equation 4.1. The lowest feature layer has a scale of 0.2 and it is denoted by s_{min}. The highest feature layer has the scale of 0.9 and it is denoted by s_{max}. In Equation 4.1, we use $s_{min} = 0.2$ and $s_{max} = 0.9$.

$$s_k = s_{min} + \frac{s_{min}}{m-1}(k-1), k \in [1, m] \tag{4.1}$$

We define the aspect ratios a_r of the default boxes before training. We center each default box at $(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|})$, where $|\ f_k\ |$ is the size of one side of the k^{th} square feature map and $i, j \in [0, |f_k|)$. We calculate the width and height of each default box for given feature map as $w_k^a = s_k\sqrt{a_r}$ and height $h_k^a = s_k/\sqrt{a_r}$. For aspect ratio 1, one additional default box of scale $s_k = \sqrt{s_k s_{k+1}}$ is added.

Example Let us understand the tiling of default boxes with an example. Consider we want to find the default boxes for feature map of size 3×3 for SSD300 architecture shown in Figure 4.7a. We have $|f_k| = 3$. As 3×3 is the fifth feature map (38×38 being first), $k = 5$. SSD300 uses total 6 feature maps, hence $m = 6$. SSD300 uses aspect ratios $\{1, 1/2, 2\}$ in the fifth feature map. Now, we can calculate the s_5:

$$s_5 = 0.2 + \frac{0.2}{6-1}(5-1) = 0.36 \tag{4.2}$$

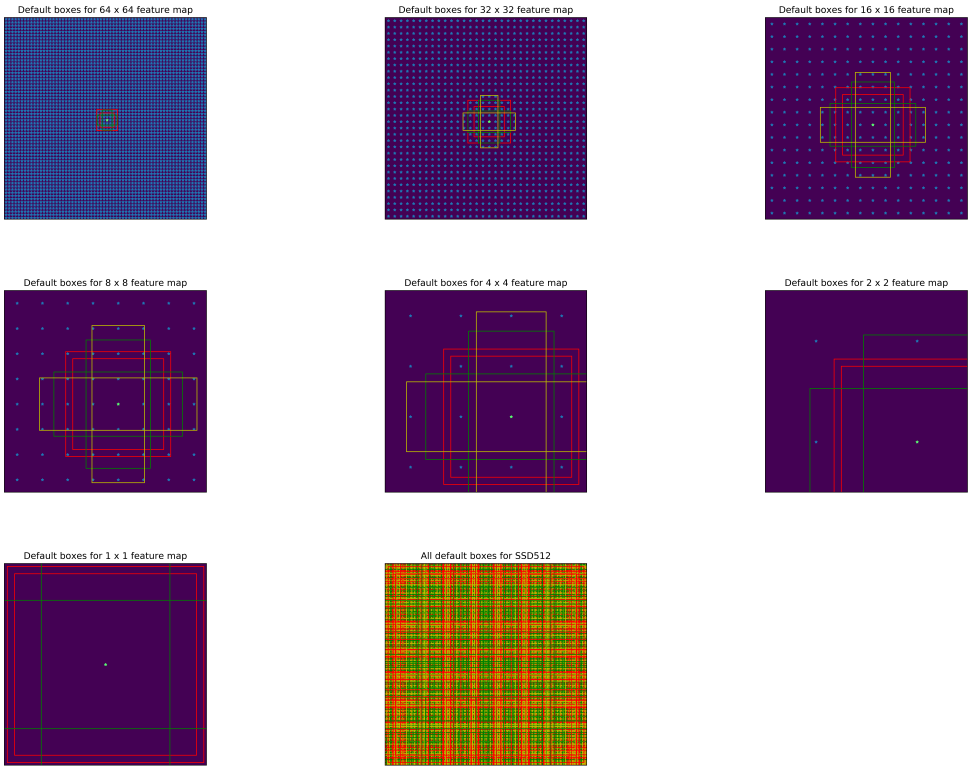


Figure 4.4: Default boxes visualization for SSD512 architecture. Each $\star$ represents the center of default boxes. Each $\star$ has a few default boxes associated with it. For example, each $\star$ in feature maps $64 \times 64, 32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4, 2 \times 2$, and 1×1 have $4, 6, 6, 6, 6, 4$, and 4 default boxes associated with them, respectively. For simplicity, we show default boxes around only one $\star$ (green). We show default boxes with same aspect ratios in the same color (e.g. red for aspect ratio 1, green for aspect ratio 2 and 1/2, yellow from aspect ratio 3 and 1/3) across all feature maps. We show all default boxes associated with every $\star$ from all feature maps at once in the last image. Size of feature maps is equal to number of default box centers for that feature map, i.e., for $x \times x$ feature map there will be x rows with x default box centers.

We define 9 locations on the feature map of size 3×3 to be the centers of our default boxes using formula $(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|})$. For example, for i=0 and j=0, we define center at $(\frac{0+0.5}{3}, \frac{0+0.5}{3})$ i.e., $(\frac{1}{6}, \frac{1}{6})$. For feature map 3×3, $i \in [0, 3)$ and $j \in [0, 3)$. For each of the 9 locations defined as the center, we add four default boxes. For aspect ratio $a = 1$, we get width $w_5^1 = s_5\sqrt{1}$ and height $h_5^1 = s_5/\sqrt{1}$. For aspect ratio $a = 1$ we also add another default box with $s_5 = \sqrt{s_5 s_6}$. So, we get a default box with width $\sqrt{s_5 s_6}$ and height $\sqrt{s_5 s_6}$. s_6 can be calculated just like s_5 using Equation 4.1. For aspect ratio $a = 2$, we get width $w_5^2 = s_5\sqrt{2}$ and height $h_5^2 = s_5/\sqrt{2}$. Similarly, for aspect ratio $a = 1/2$, we get width $w_5^{1/2} = s_5\sqrt{1/2}$ and height $h_5^{1/2} = s_5/\sqrt{1/2}$. As there are 9 center locations and 4 default boxes per location, we get total 36 default boxes for feature map 3×3. Here, we calculated the center locations, width, and, height for unit size image. For SSD300, input image size is 300×300. Therefore, we need to scale the calculated values of center locations, width, and, height by 300.

Similar to the feature map of size 3×3, we can calculate centers, widths, and heights of the default boxes for other feature maps.

4.2.4.1 Counting Total Number of Default Boxes

Original SSD300 [38] architecture uses feature maps of size $38 \times 38, 19 \times 19, 10 \times 10, 5 \times 5, 3 \times 3$, and 1×1 with default boxes $4, 6, 6, 6, 4, 4$. Hence, total number of default boxes in original SSD300 architecture can be calculated as:

$$
\begin{aligned}
\text{total default boxes for original SSD300} &= 38 \times 38 \times 4 \\
&+ 19 \times 19 \times 6 \\
&+ 10 \times 10 \times 6 \\
&+ 5 \times 5 \times 6 \\
&+ 3 \times 3 \times 4 \\
&+ 1 \times 1 \times 4 \\
&= 8732
\end{aligned}
\tag{4.3}
$$

Similarly, for original SSD512:

$$
\begin{aligned}
\text{total default boxes for original SSD512} &= 64 \times 64 \times 4 \\
&+ 32 \times 32 \times 6 \\
&+ 16 \times 16 \times 6 \\
&+ 8 \times 8 \times 6 \\
&+ 4 \times 4 \times 6 \\
&+ 2 \times 2 \times 4 \\
&+ 1 \times 1 \times 4 \\
&= 24564
\end{aligned}
\tag{4.4}
$$

So, original SSD300 and SSD512 use 8732 and 24564 default boxes respectively[2]. We describe the number of default boxes used for each experiment in Section 4.1.

[2]Please refer original caffe implementation and documentation regarding number of de-

In the next section we discuss the matching strategy and why it is important to use multi-scale feature maps with multiple aspect ratios.

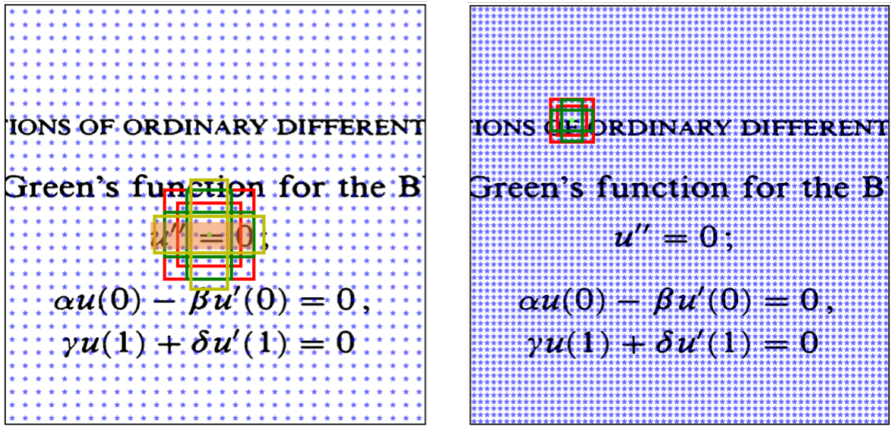
4.2.5 Matching Strategy

Our matching metric is same as original SSD [38]. Each ground truth box is matched to a default box with best IOU (refer to Chapter 3). IOU is defined as the area of intersection over the area of union for given shapes. It is also known as the Jaccard index or Jaccard overlap. Also, each default box is matched with a ground truth box with Jaccard overlap of more than a threshold (0.5). Matching more than one default boxes to one ground truth box simplifies the learning problem by allowing the network to predict higher scores for more default boxes. The matched default boxes are considered as positive examples (POS) and other default boxes are considered as negative examples (NEG).

Figure 4.5 show default boxes overlaid on the input image of size 512×512. Each feature map is a pixel grid, but the associated default boxes are defined by the co-ordinates in the original image space. The lower level feature map 64×64 has smaller default boxes associated with it. The higher level default box 32×32 has larger default boxes associated with it. Hence, 32×32 feature map learns to detect bigger objects as compared to 64×64 feature map. If we only use features from 32×32 default box for detection we might

fault boxes: `https://github.com/weiliu89/caffe/tree/ssd`. Calculated number of default boxes matches perfectly.

miss smaller objects. Hence, it is important to use feature maps of different sizes for prediction. For highlighted ground truth box in the Figure 4.5 wide yellow box has the maximum IOU. Hence, while training a wide yellow box will be matched with the highlighted ground truth.



(a) Default boxes for 32×32 feature map overlaid on input image of size 512×512.

(b) Default boxes for 64×64 feature map overlaid on input image of size 512×512

Figure 4.5: Example of default boxes for 32×32 and 64×64 overlaid on the input image of size 512×512. Each location denoted by a blue $\star$ has the same set of associated default boxes for each feature map. Figure 4.5a shows a highlighted ground truth.

Original SSD [38] architecture uses aspect ratios (width/height) of $\{1, 2, 3, 1/2, 1/3\}$. But, Figure 4.6 shows that there the many wide default boxes with aspect ratio more than 3 in the dataset. It is important to use default boxes with an aspect ratio of more than 3 for effective training. Wide default boxes will have higher chance of matching with wide ground truths as per matching strategy explained in Section 4.2.5. Hence, along with default

boxes used in the original SSD, we also experiment with wider default boxes used in TextBoxes[32] of aspect ratios $\{5, 7, 10, 1/5, 1/7, 1/10\}$. Also, we experiment with only wide default boxes with aspect ratios $\{1, 2, 3, 5, 7, 10\}$. We explain the details of our experiments in Section 4.5.

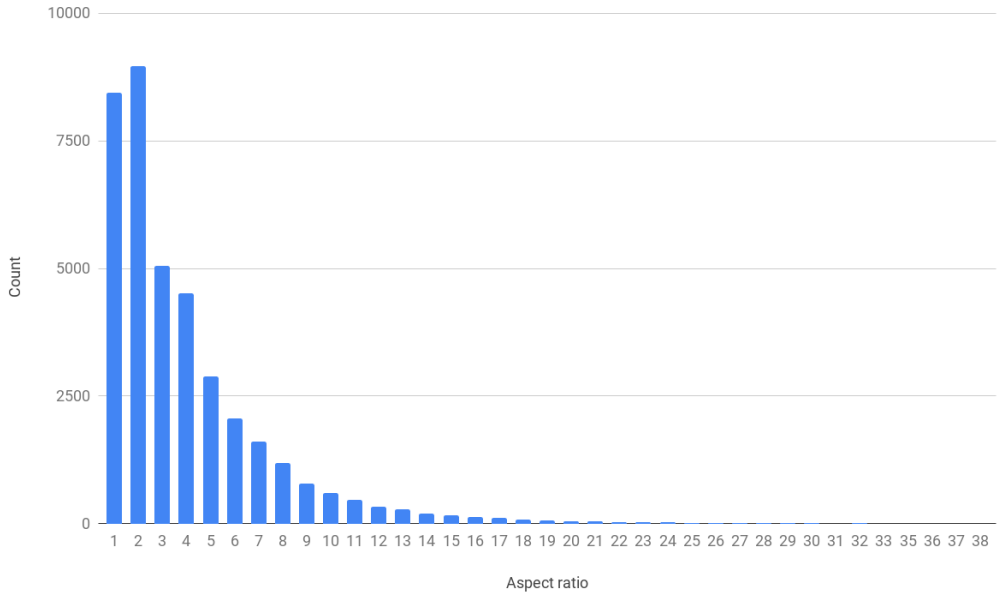


Figure 4.6: Histogram of aspect ratios of math expression bounding boxes in the GTDB dataset. We round all the aspect ratios up while generating this histogram. There are many math regions with aspect ratios over 3.

SSD300 architecture proposed in the paper [38] used 8732 default boxes. In our experiments, the use of additional wider default boxes results in far more default boxes than original SSD. As the number of non-math regions is higher than the number of math regions, the number of the default boxes that match math regions is lower than the number of default boxes that match the non-

math regions, i.e., POS ¡ NEG. It results in the class imbalance problem. In our experiments, we tried the focal loss[33] and hard negative mining to reduce the effect of class imbalance (refer to Section 4.3.2) .

4.2.6 Kernel Sizes for Detection

The architecture in 4.3 only uses convolution and pooling layers i.e. the network is fully convolutional. In case of SSD architecture, at each location in the feature layer of size $m \times n$ with p channels a kernel of size $a \times b \times p$ is applied to produce an output value. Original SSD [38] uses $3 \times 3 \times p$ kernel. We try $3 \times 3 \times p$ and $1 \times 5 \times p$ kernels in our experiments. p changes depending on the feature map. Liao et al. [32] claim that kernels of size 1×5 yield rectangular receptive fields which better-fit words with larger aspect ratios, also avoiding noisy signals that a square-shaped receptive field would bring in. They base their architecture for text detection in natural scenes on the SSD. Math expressions are usually wide, and hence we tried the kernel of size 1×5. We explain the details for each experiment in Table 4.1.

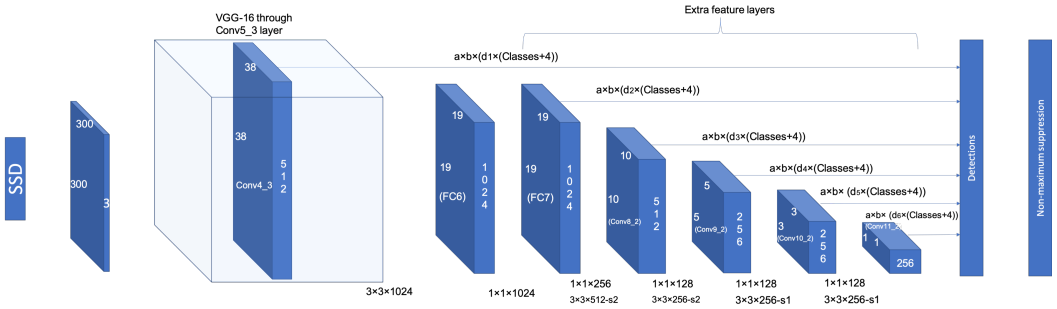
Now, let us look at the complete SSD architecture.

4.3 SSD Architecture

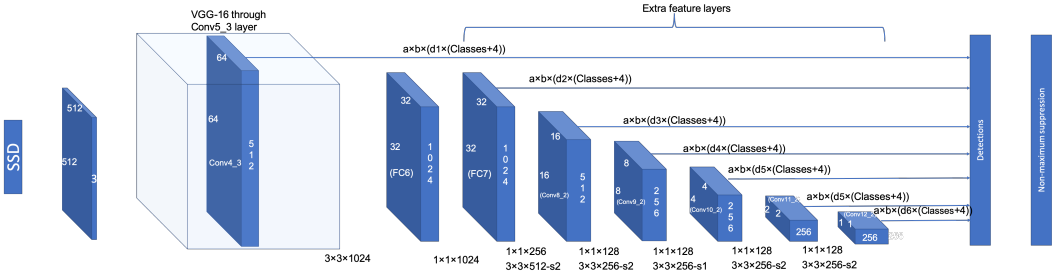
Figure 4.7 shows the SSD300 and SSD512 architectures. These architectures are feed forward convolution networks that produce a fixed number of bounding box hypotheses, each with an associated probability. Non-maximal suppression (NMS) is applied to the bounding boxes to get the final SSD out-

put results. NMS is a greedy, locally optimal strategy which keeps only one detection per group (ideally obtaining only one detection per object) from relatively dense detection outputs near the correct location of the object generated by object detection methods. Section 2.3.1 from Chapter 2 describes the NMS in detail. These SSD architectures use VGG16[48] pre-trained on a ImageNet[9] dataset as a base network. VGG16 is a deep convolutional neural network with 13 convolutional layers and 3 fully-connected layers. VGG16 architecture is described in Chapter 2, Section 2.3.3. We use the first 13 convolutional layers from the VGG16 network as it is. The last fully-connected layer FC8 is removed. The reamining two fully-connected layers are converted to convolutional layers using a method described in Chapter 2, Section 2.3.4. The Figure 4.7 shows FC6 and FC7 layers converted to convolutional layers. We mention the sizes of FC6 and FC7 on the layers. As compared to SSD300, SSD512 is deeper with one extra feature layer (Conv12_2) and has input image size 512×512 instead of 300×300.

Both original SSD300 and SSD512 [38] uses $a = 3$, $b = 3$. Original SSD300 uses $d_1 = 4, d_2 = 6, d_3 = 6, d_4 = 6, d_5 = 4, d_6 = 4$. SSD512 uses $d_1 = 4, d_2 = 6, d_3 = 6, d_4 = 6, d_5 = 6, d_6 = 4, d_7 = 4$. Conv4_3, FC7, Conv8_2, Conv9_2, Conv10_2, Conv11_2 are used as feature layers in SSD300. SSD512 adds one more feature layer Conv12_2. 'Classes is the number of classes to be predicted by the network. In our case, Classes=2 (math and non-math). For i^{th} feature map of size $m \times n$ the number of default boxes is calculated as $m \times n \times d_i$. For each default box, network predicts class confidences and 4 offset values.



(a) A SSD300 network with VGG16 as a base network



(b) A SSD512 network with VGG16 as a base network

Figure 4.7: Simplified SSD300 and SSD512 architectures with VGG16 as base network. $a \times b$ is the size of the kernel. d_i represents the number of default boxes used in the feature map. Detailed architecture is shown in Figure 4.8.

Hence, each feature layer is used to predict $m \times n \times d_i \times (Classes + 4)$ values. Refer to Section 4.2.4 for more information about default boxes.Convolution layers applied applied after each feature layer are shown at the bottom of each feature layer with stride used (s1 = stride 1 and s2 = stride 2). Default stride is 1. We mention the parameters we used while training in the Table 4.1.

The detailed SSD300 architecture is shown in the Figure 4.8. We show the changes in the original VGG16 architecture, extra layer, localization, and, confidence layers with number of input and output channels, kernel sizes, stride, and, padding. Also, we show the output size after each layer and input and output sizes for the localization (LOC) and confidence (CONF) layers. There are 6 feature maps, 6 LOC layers (one per feature map), and, 6 CONF layers (one per feature map).

4.3.1 Loss Functions

We use the loss function proposed by Liu et al. in the SSD paper [38]. Let $x_{ij}^p = \{1, 0\}$ be an indicator for matching the i^{th} default box to the j^{th} ground truth box of category p. In the matching strategy used in the model (refer to section 4.2.5 for more details), we can have $\sum_i x_{ij}^p \geq 1$. It means that more than one default box can match one ground truth box. The overall loss is a weighted sum of localization loss and confidence loss. Next, we describe the bounding box regression problem and the localization loss.

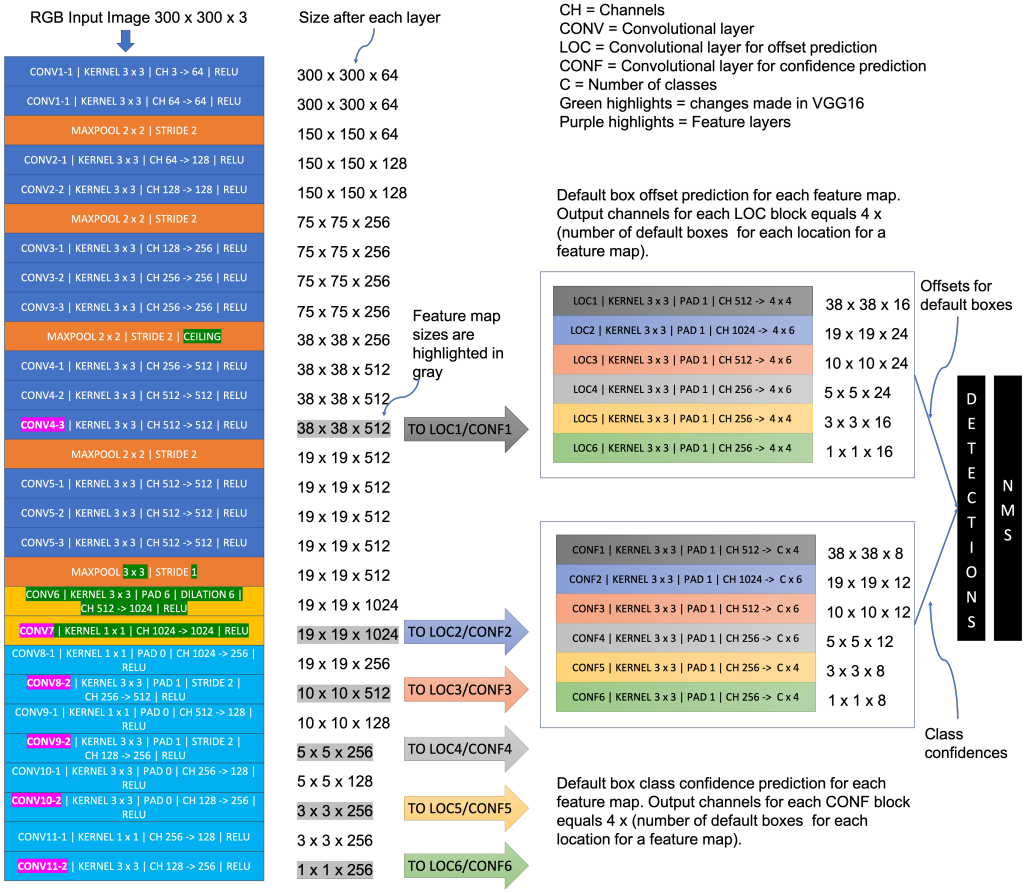


Figure 4.8: Detailed SSD300 architecture. The FC6 and FC7 layers from VGG16 are converted to yellow boxes: CONV6 and CONV7. The green highlights show the difference between original VGG16 (refer to Section 2.2) and VGG16 used by SSD300. FC8 layer from VGG16 is removed. The light blue boxes show the extra layers added after VGG16. The purple highlights show the feature map layers. Feature maps extracted from feature layers are used by LOC and CONF layers to predict the four bounding box offsets $\Delta(cx, cy, w, h)$ and class confidences for each default box at each location.

Bounding Box Regression Let us define the predicted box l as $l = \{l^{cx}, l^{cy}, l^{w}, l^{h}\}$, where superscript ($cx$, cy) is the x and y co-ordinate of the center of the box, w is the width, and, h is the height. Similarly, the ground truth bounding box g can be defined as $l = \{l^{cx}, l^{cy}, l^{w}, l^{h}\}$. We configure the bounding box regressor to learn four functions, f_{cx}, f_{cy} for scale-invariant transformation between centers of l and g and f_w, f_h for log-scale transformation between widths and heights. In general, we define f_i, where $i \in \{cx, cy, w, h\}$ as a bounding box correction function. f_i takes l as the input. The target values $\hat{g}$ for bounding box correction functions f_i are defined below:

$$\begin{aligned}
\hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^{w} \\
\hat{g}_j^{cy} &= (g_j^{cy} - d_i^{cy})/d_i^{h} \\
\hat{g}_j^{w} &= \log(\frac{g_j^{w}}{d_i^{w}}) \\
\hat{g}_j^{h} &= \log(\frac{g_j^{h}}{d_i^{h}})
\end{aligned} \tag{4.5}$$

We use $smooth_{L1}$ loss defined by Girshick[19] for the bounding box regression problem. It is given by Equation 4.6. It is claimed to be less sensitive to outliers.

$$smooth_{L1}(t) = \begin{cases} 0.5t^2, & \text{if } \mid t \mid < 1; \\ \mid t \mid -0.5, & \text{otherwise} \end{cases} \tag{4.6}$$

Localization Loss The localization loss is the smooth L1 loss [12] between the predicted box(l) and the ground truth box(g). We calculate the overall

localization loss as:

$$L_{loc}(x, l, g, f) = \sum_{i \in POS} \sum_{m \in \{cx,cy,w,h\}} (x_{ij}^{p} smooth_{L1}(f_m(l_i) - \hat{g}_j^m)) \tag{4.7}$$

where POS are positive examples and $\mid POS \mid = N$. POS and NEG (negative examples) are defined in the Section 4.2.5. x is the indicator function, l is the predicted box, g is the ground truth, and, f_i is the set of bounding box correction functions. From Equation 4.7 we can see that the localization loss is calculated only for positive examples.

Confidence Loss The confidence loss is the cross entropy loss over multiple class confidences (c). It is given by Equation 4.8.

$$\begin{aligned} L_{conf}(x, c) &= - \sum_{i \in POS} x_{ij}^{p} \log \hat{c}_i^p - \sum_{i \in NEG} \log(\hat{c}_i^0) \\ &\text{where,} \\ \hat{c}_i^p &= exp(c_i^p) / \sum_k exp(c_i^k) \end{aligned} \tag{4.8}$$

Overall Loss The overall objective loss function used in our model is given by Equation 4.9:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g, f)) \tag{4.9}$$

where N is the number of matched default boxes. If N is 0, the loss is set to 0. We set the weight term α to 1 like original SSD[38].

4.3.2 Dealing with Class Imbalance

In this section we discuss two techniques we used to address the class imbalance problem we get from having many more non-math than math regions during training: focal loss and hard negative mining.

4.3.2.1 Focal Loss

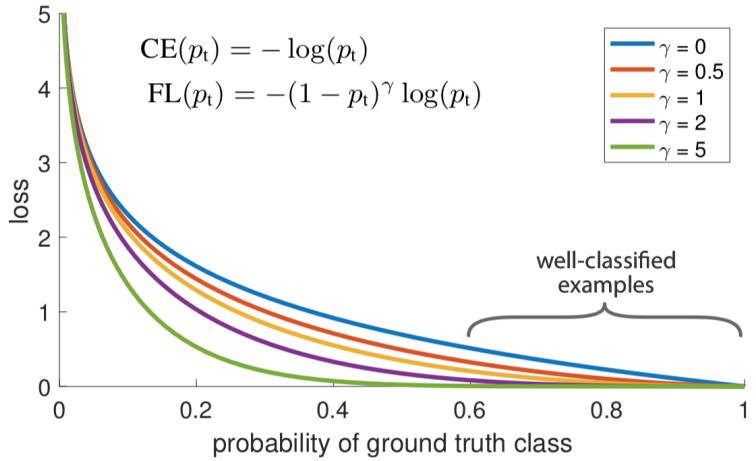


Figure 4.9: Behavior of focal loss function for different values of γ. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > 0.5$), putting more focus on hard, misclassified examples. The blue line is the CE loss. The purple line shows the curve for the FL with $\gamma = 2$ which was found to be working best by Lin et al [33]. This figure is taken from Lin et al. [33].

The original SSD architecture[38] used the cross-entropy (CE) function (refer to Equation 4.8) for confidence loss calculation. Cross-entropy is defined by Equation 4.10. Here $y \in \{\pm 1\}$ which is the ground truth class and $p \in [0, 1]$ is the model's estimated probability.

$$CE(p, y) = \begin{cases} -\log p, & \text{if } y = 1 \\ -\log(1 - p), & \text{otherwise} \end{cases} \tag{4.10}$$

The FL is modification of CE. Lin et al.[33] define p_t as given in Equation 4.11 and rewrite the cross-entropy function (Equation 4.10) as Equation 4.12. The FL is defined in the Equation 4.13.

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases} \tag{4.11}$$

$$CE(p, y) = CE(p_t) = -\log p_t \tag{4.12}$$

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{4.13}$$

If $\gamma = 2$ and $p_t = 0.9$, then FL will be 100 times lower than the CE. When $p_t = 0.968$, FL will be 1000 times lower than the CE. This allows the network to focus more on correcting the misclassified examples. Also, when $\gamma = 0$, FL is equivalent to CE. Figure 4.9 shows the behavior of FL for different γ values. We call $(1 - p_t)^{\gamma}$ a modulating factor. As $p_t \to 1$, the $(1 - p_t)^{\gamma} \to 0$ and effect of easy examples is down-weighted. The focusing factor γ smoothly adjusts

the rate at which the examples are down-weighted. Here, by easy examples we mean examples were classified with high confidence by the network. Easy examples can include math and non-math regions.

In practice α balanced variant of focal loss is used. $\alpha \in [0, 1]$ is a weighting factor for class 1 and $1 - \alpha$ for class -1.

$$\alpha_t = \begin{cases} \alpha, & \text{if } y = 1 \\ 1 - \alpha, & \text{otherwise} \end{cases} \tag{4.14}$$

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{4.15}$$

Focal loss is applied on all the examples while training, none of the examples are left out while training. Lin et. al [33] claim that $\gamma = 2$ and $\alpha = 0.25$ worked best in their experiments. We experimented with both CE and FL. For more details on the experiments, please refer to Chapter 5.

Let us see a method used in the original SSD architecture [38] to deal with the class imbalance problem called hard negative mining.

4.3.2.2 Hard Negative Mining

After matching step described in Section 4.2.5, we end up with positive examples and a large number of negative examples, especially when there is a large number of default boxes. Hence, while training instead of using all negative examples, they are sorted by their confidence in descending order and top examples are selected such that the ratio between positive and negative

examples is at most 1:3 as per original SSD model [38]. It means that only the negative examples with the highest confidence are used in the computation of total loss. If hard negative mining (HNM) is used, in our loss function (refer to Section 4.3.1), when number of positive examples is equal to N (i.e. $\mid POS \mid = N$), the number of negative examples must be less than or equal to $3N$ (i.e., $\mid NEG \mid \leq 3N$). We apply HNM every time we compute the confidence loss. Liu et al. [38] claim that, HNM results in faster optimization and more stable learning.

4.4 Pooling Detected Math Regions

We show the steps in the pooling process in Figure 4.10. As shown in Figure 4.10, at the inference time we generate smaller sub-images from the input image. We pass these sub-images as input to the model and get the predicted bounding boxes with prediction confidence. Figure 4.11 shows patch level detection results with confidence. The same region can be seen multiple times because of the sliding window method used and multiple bounding boxes can be predicted for the same region. To get final detection results, we use the predictions from the network for voting at the pixel level. We consider each pixel in the bounding box to have the same confidence as the bounding box.

Let B be a set of page level bounding boxes and C be set of confidences obtained after patch level detection results. $B_i \in B$ is a i^{th} bounding box with confidence $C_i \in C$. Let each pixel in image I of size $h \times w$ be represented by set of pixels P_{ab}, where $a \in [0, h)$ and $b \in [0, w)$. We say that a pixel $P_{ab} \in B_i$

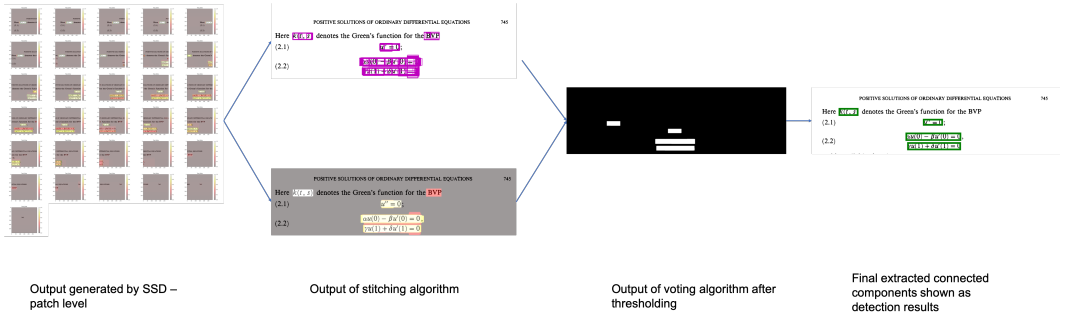


Figure 4.10: Voting based pooling method to get final detection results

if it is inside the bounding box B_i. Let us define $L_{ab}^i = 1$ if $P_{ab} \in B_i$ and 0 if $P_{ab} \notin B_i$. It is possible that $\sum_i L_{ab}^i \geq 1$ which means that P_{ab} can belong to more than one bounding box. To calculate the score S_{ab} at P_{ab} we tried different voting functions. They are defined as follows:

- Uniform weighting: Each bounding box is given the same weight. So, the number of bounding boxes in which the pixel exists is the number of votes for that pixel.

$$S_{ab} = \sum_{i=0}^{|B|} L_{ab}^i \tag{4.16}$$

- Maximum score: The maximum confidence the network had for a pixel is the score for that pixel.

$$S_{ab} = Max(L_{ab}^i C_i), \text{where } i \in [0, \mid B \mid) \tag{4.17}$$

- Average score: The average of confidences of all bounding boxes which contain P_{ab}.

$$S_{ab} = \frac{\sum_{i=0}^{|B|} L_{ab}^{i} C_i}{\sum_{i=0}^{|B|} L_{ab}^{i}} \tag{4.18}$$

- Sum score: We sum the confidences of all bounding boxes surrounding the pixel and assign it as a score to that pixel.

$$S_{ab} = \sum_{i=0}^{|B|} L_{ab}^{i} C_i \tag{4.19}$$

After assigning the pixel scores, we apply a threshold. Figure 4.12 shows how thresholding is applied at the page level. We decide the threshold value using training data by performing grid search over possible threshold values and finding a value for which network generates best detection results (f-score). We found that the average score does not perform as good as other methods. In Chapter 5, Figure 5.2a, 5.2b, and, 5.2c show the f-score for different threshold value when threshold is 50% and 75% for maximum score algorithm, sum score algorithm and equal score algorithm respectively. For uniform weighting and sum score, we try thresholds from 0 to 55 with an increment of 1. For maximum score the minimum value is 0 and maximum value is 100. So, we try threshold values from 0 to 100 with an increment of 1. We choose the threshold value for which we get maximum F-score for IOU75. For uniform weighting, we get the highest F-score for IOU75 of 76.8% at the threshold of 30. F-score for IOU50 for the same threshold is 86.8%. We observed that the algorithm sum score has comparable highest F-score of 75.6% for IOU75 and 85.92% for IOU50 for the threshold of 13. For the max

Figure 4.11: The bounding boxes and confidences for sub-images predicted by the SSD. It can be observed that the confidences of math expressions changes as the window shifts. The network is sensitive to the position of the object in the input image. The confidence scale: $gray \approx 0$, $red \approx 0.5$, $white \approx 1.0$

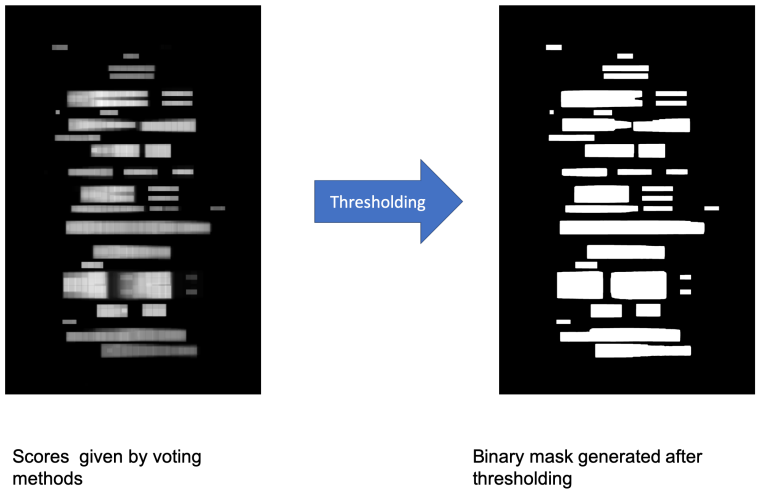


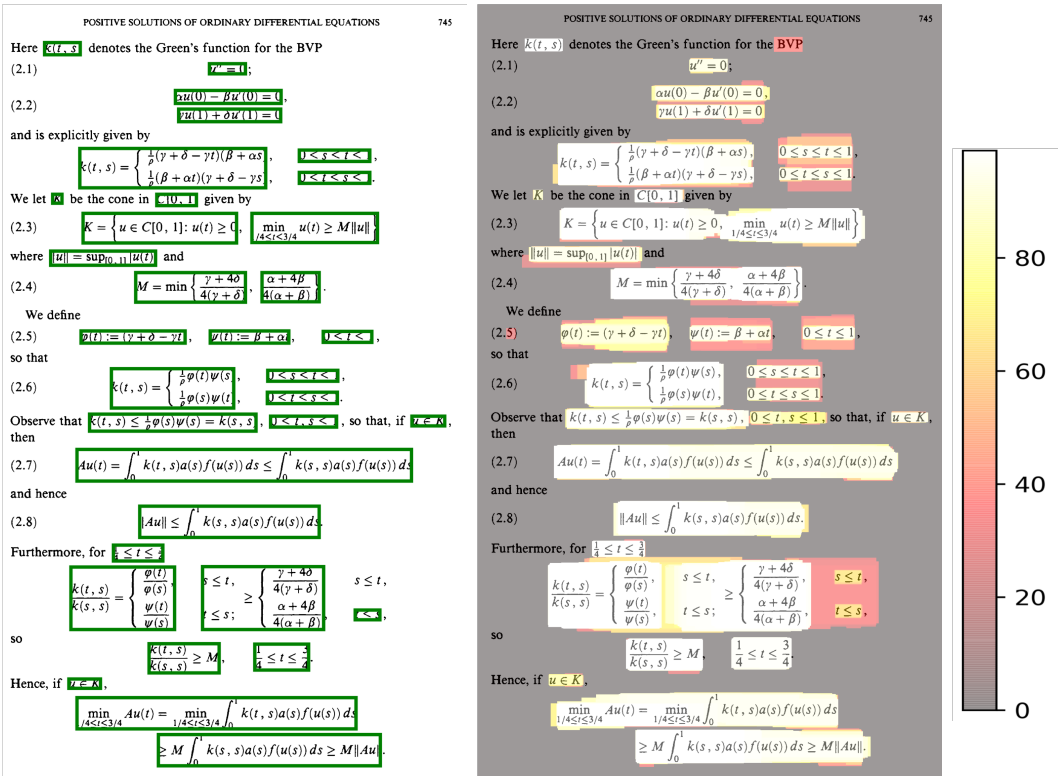
Figure 4.12: Thresholding operation. The left image shows the votes given to each pixel by the voting algorithm and the right image shows the binary mask generated after thresholding

score, we get the highest F-score of 62.72% for IOU75 and 72.59% for IOU50 at the threshold of 81. For further explanation and results refer to Chapter 5.

We generate a final binary mask using the threshold value. Pixels which have scored more than the threshold are assigned a value of 1 and others are assigned 0. Finally, we extract bounding boxes of the connected components from the binary mask as math expressions. Figure 4.13 shows the full page detection results and confidence scores after pooling.

4.5 Training

Table 4.1 shows the parameters we used in our experiments. Sub-image generated using sliding window of size 1200×1200 is converted to input image size. Aspect ratios (AR) used are A_1,A_2, A_3, and, A_4 . A_1



(a) Full page detection results after pooling

(b) Full page confidence score. Only maximum confidence is shown.

Figure 4.13: Full page results after pooling

Table 4.1: Parameters used in the experiments

Experiment	**Input size**	**AR**	**Kernel**	**HNM**	**LF**
SSD300	300×300	A_1	3×3	1:3	CE
SSD512	512×512	A_2	3×3	1:3	CE
ASPECT512	512×512	A_3	3×3	1:3	CE
HBOXES512	512×512	A_4	3×3	1:3	CE
FLNEG512	512×512	A_3	3×3	1:3	FL
FL512	512×512	A_3	3×3	OFF	FL
MATH512	512×512	A_3	1×5	1:3	CE

represents the aspect ratios used in the original SSD300[38]. For feature map $38 \times 38, 3 \times 3$, and 1×1 we use 3 aspect ratios $1, 2, 1/2$. For feature maps 19×19, 10×10, 5×5 we use additional $\{3, 1/3\}$ aspect ratios. A_2 represents the aspect ratios used in the original SSD512[38]. For feature maps 64×64, 2×2, and, 1×1 we use $\{1, 2, 1/2\}$. For feature maps 32×32, 16×16, 8×8, and, 4×4 we use additional $\{3, 1/3\}$ aspect ratios. A_3 denotes use of $\{1, 1/2, 1/3, 1/5, 1/7, 1/10, 2, 3, 5, 7, 10\}$ for all feature maps. A_4 denotes use of $\{1, 2, 3, 5, 7, 10\}$ for all the feature maps. Note that for aspect ratio 1×1 we add two default boxes as explained in the Section 4.2.4. Exactly one default box is added for other aspect ratios. Suffix 300 and 512 in the experiment name denotes use of 6 feature layers like SSD300 and 7 feature layers like SSD512 respectively. The kernel sizes shown in the table are used for extraction of feature maps from the feature layers. These kernel sizes are denoted by the $a \times b$ in Figure 4.3. HNM is hard negative mining. The ratio shown in the HNM column is the ratio of positive examples to the negative examples. LF is loss function used for calculating the classification loss. CE is cross entropy and FL is focal loss.

The VGG16 base network is pre-trained[3] on the ImageNet [9] dataset. ImageNet is a large image dataset with average of 500 images per category. We fine-tune the resulting model using stochastic gradient descent (SGD) with an initial learning rate 10^{-4}, 0.9 momentum, 5×10^{-4} learning rate and a

[3]Weights are available at `https://s3.amazonaws.com/amdegroot-models/vgg16_reducedfc.pth`

batch size of 16. We also use a learning rate decay policy. We decay learning rate by factor of 10 at iteration 80k, 100k and 120k. We train our network for 132k iterations (4 epochs). For non-maximal suppression we use IOU threshold of 0.45 and confidence threshold of 0.25.To avoid overfitting we use data augmentation (refer to Section 4.2.2) and validation set. We split each PDF from training data in 80% and 20%. The pages that were used for validation are given in Appendix A.

We train the best performing model using training and validation dataset for 165k iterations (4 epochs) and test the generalization on the test dataset.

4.5.1 Number of Default Boxes Per Experiment

Figure 4.14 shows the total number of default boxes used in each experiment. The method to calculate the number of default boxes is explained in the Section 4.2.4.1.

4.6 Implementation

We modified open source PyTorch [42] implementation of SSD[4] to use it for math expression detection. Max deGroot who works with Amazon Alexa developed this repository. It supports training and testing SSD300 model on the VOC 2007, VOC 2012 and COCO dataset. It uses Visdom[5] for plotting training curves. Visdom is a flexible tool for creating, organizing, and sharing

[4] `https://github.com/amdegroot/ssd.pytorch`
[5] For more information refer to `https://github.com/facebookresearch/visdom`

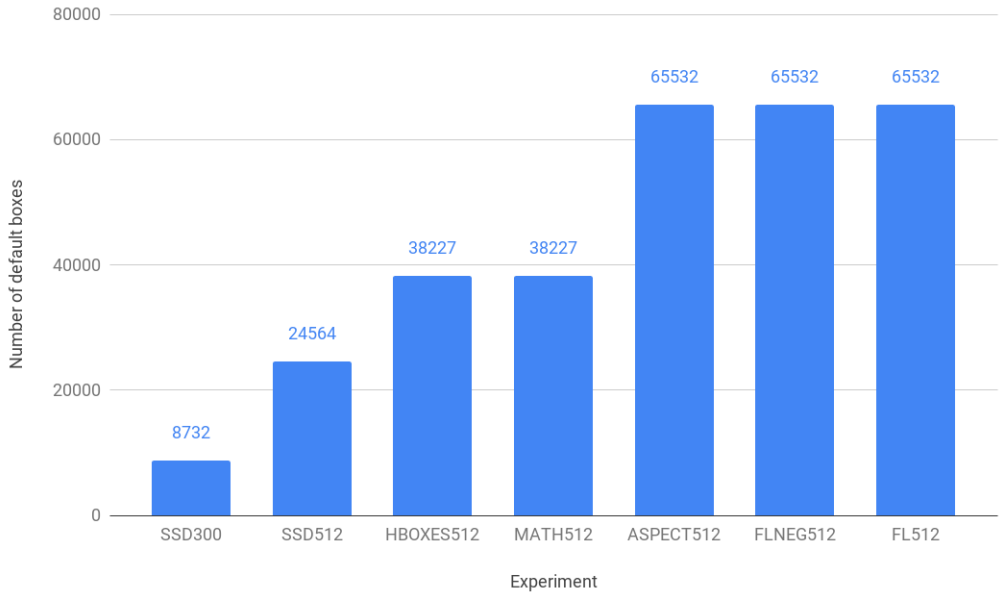


Figure 4.14: Total number of default boxes used in each experiment

visualizations of live data and supports Torch[6] and Numpy[7]. Below is a list of major changes in the original implementation.

1. We added support to train and test on the GTDB dataset.

2. We added support to train and test the SSD512 model.

3. We added multiple command-line arguments and configuration file to easily modify the hyper-parameters for experiments.

[6]Torch is a scientific computing framework with wide support for machine learning algorithms. Read more on `http://torch.ch/`

[7]Numpy is a libray in Python programming language that supports large multidimensional arrays. Read more on `https://numpy.org/`

4. We added data reader for GTDB dataset which uses a sliding window method described in Section 4.2.1 to generate sub-images and new annotations.

5. We added a confidence map and detection results visualization method.

6. We added scripts to perform the grid search for threshold values.

7. We implemented the stitching algorithm described in Section 4.4 including uniform weighting, average score, maximum score, and, sum score.

8. We added support for using a validation set while training and automatic generation of validation loss vs. epoch graphs using Visdom.

9. The original code only supported testing one image at a time, so we added support for batch testing.

10. We use the logger to store the logs generated while performing the experiments.

11. We added functionality to measure the training and testing time.

12. We also added other functionalities for better organization of generated outputs, like changing the titles of generated training plots as per the given experiment name, storing weights in a directory named experiment name, etc.

13. We implemented a focal loss function and ability to easily select between cross-entropy and focal loss using command line arguments.

14. We implemented functionality to turn on or off the hard negative mining and tall default boxes.

15. In the original code, we got $-\infty$ values in the loss sometimes. It was introduced while calculating $\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w})$ and $\hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$ in Equation 4.7. If g_j^w or g_j^h is 0, we replace it with a very small value [8].

16. The random crop logic[9] was not the same as original SSD paper [38]. We modified the code to make it same as original SSD paper.

[8]For more information refer to `https://github.com/amdegroot/ssd.pytorch/pull/116`

[9]For more information refer to `https://github.com/alamdegroot/ssd.pytorch/issues/119`

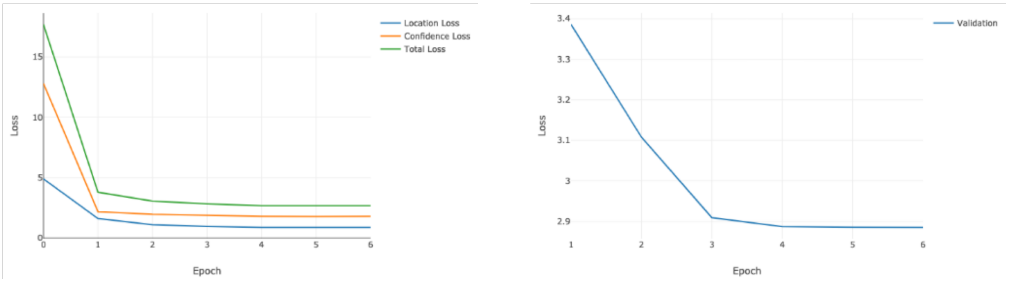
Chapter 5

Results

In this chapter, we measure the effectiveness of the proposed method using the TFD-ICDAR2019 dataset. We designed the experiments to find the hyper-parameters for SSD model and chose a set of parameters that performed best on the validation set for measuring the performance of ScanSSD on the test dataset. We also analyze results at the character level. We provide the number of sub-images generated using sliding windows for training, validation, and, test dataset. We provide training time and compare precision, recall, and, F-score of a single character and multi-character math expressions. Moreover, we find how effectively our model detected the math characters in the document images. We compare the performance of our models on the validation set and decide the model with the highest F-score on the validation set as the final model. We train our final model with training and validation data and test the generalization on the test dataset. We use a grid search to find out the best pooling method and the best threshold for the pooling method. Also, we studied the effect of postprocessing on the detection results.

5.1 Effect of Data Augmentation

It is common in the object detection literature to use extensive data augmentation to avoid overfitting on the training data [43, 38]. In addition to data augmentation while training as described in Section 4.2.2, sliding windows generate a large number of sub-images from a small number of pages. For training and validation, we get 524,718 and 131,999 sub-images with math regions respectively. Validation split is provided in Appendix A. We trained our model for 6 epochs (more than 185k iterations) and found out that the validation loss continued to decrease.



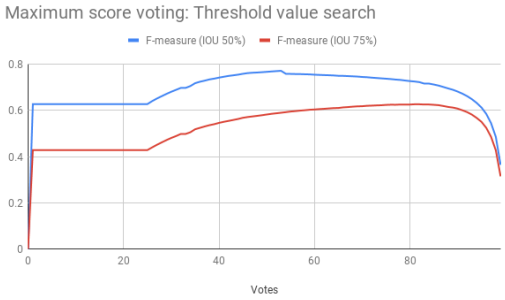
(a) Loss on training set per epoch (b) Total loss on validation set per epoch

Figure 5.1: The figure shows continued decrease in the total loss on validation set. The rate of decrease in loss on validation set is almost 0 after epoch 4.

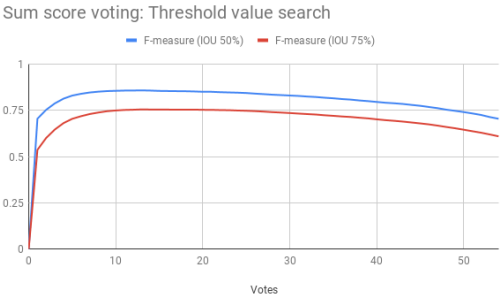
Figure 5.1 shows the continued decrease in validation loss while training. The rate of decrease in validation loss is almost zero after epoch 4. So, we train our models for 4 epochs ($\approx$ 132k iterations on training dataset) while performing our experiments. Each iteration uses a batch size of 16. Each batch is generated by sliding a window of size 1200×1200 on a page image maintaining 90% overlap till we get 16 sub-images. We apply data augmentation

explained in Chapter 4, Section 4.2.2 for each batch.

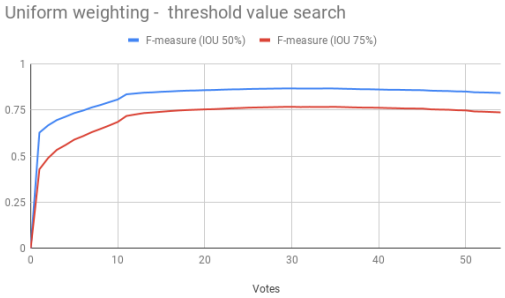
5.2 Finding a Good Threshold for Pooling Methods



(a) Threshold value search for maximum score algorithm. Highest F-score obtained for threshold 53 for IOU50 and threshold 81 for IOU 75.



(b) Threshold value search for sum score algorithm. Highest F-score obtained for threshold 13 for both IOU 50 and IOU 75.



(c) Threshold value search for uniform weighting algorithm. Highest F-score obtained for threshold 30 for both IOU 50 and IOU 75.

Figure 5.2: The grid search for the threshold value for different voting based algorithms

We decide the threshold value for pooling algorithms explained in Section 4.4 by performing grid search over possible threshold values and finding a value for which network generates best detection results (F-score) on the training data. We choose the threshold value for which we get maximum F-score for IOU75. Figure 5.2a, 5.2b, and, 5.2c show the F-score for different threshold value when threshold is 50% and 75% for maximum score algorithm, sum score algorithm and uniform weighting algorithm respectively. For uniform weighting and sum score, we try thresholds from 0 to 55 with an increment of 1. We observed a continued decrease in the score after threshold 30 for uniform weighting and after threshold 13 for sum score. For maximum score, the minimum threshold value is 0 and the maximum value is 100 as it is the network confidence which ranges from 0 to 100. So, we try threshold values from 0 to 100 with an increment of 1. For uniform weighting, we get the highest F-score for IOU75 of 76.8% at the threshold of 30. F-score for IOU50 for the same threshold is 86.8%. We observed that the algorithm sum score has comparable highest F-score of 75.6% for IOU75 and 85.92% for IOU50 for the threshold of 13. For the max score, we get the highest F-score of 62.72% for IOU75 and 72.59% for IOU50 at the threshold of 81. It is not shown in the figure, but the precision and recall gradually decrease as the number of votes increase for the pooling methods. We use uniform weighting with threshold 30 for all experiments as it obtains highest F-score among the scoring methods that we tried.

5.3 Results on Validation Dataset

To decide which hyper-parameters work well for math detection in SSD, we designed a set of experiments. All the experiments were performed exactly once. In this section, we provide results obtained on the validation dataset for all the experiments. The total number of sub-images in the validation set is 207,856. Out of 207,856 sub-images, 131,999 sub-images have at least one math region.

5.3.1 Precision, Recall, and F-score

Figure 5.3 summarizes the results on the validation dataset. We started with SSD300 which takes input image of size 300×300 as our baseline. We changed the input image size to 512×512 for SSD512 and observed the significant improvement in the F-score for both IOU50 and IOU75. Refer to Chapter 4, Section 4.5 for detailed explanation of each experiment. ASPECT512 used additional wider default boxes. We used exactly 12 default boxes for each grid center in a feature map. We were expecting improvement in the F-score with additional wider default boxes but the F-score decreased as compared to SSD512. It is possible that the initial random initialization of weights was not good for ASPECT512.

In HBOXES512, we removed all the tall default boxes and just used the wide default boxes. HBOXES512 performed better than the ASPECT512. Removal of tall default boxes significantly reduced the number of default boxes per input image. Most of the math regions are wider, use of only wide default

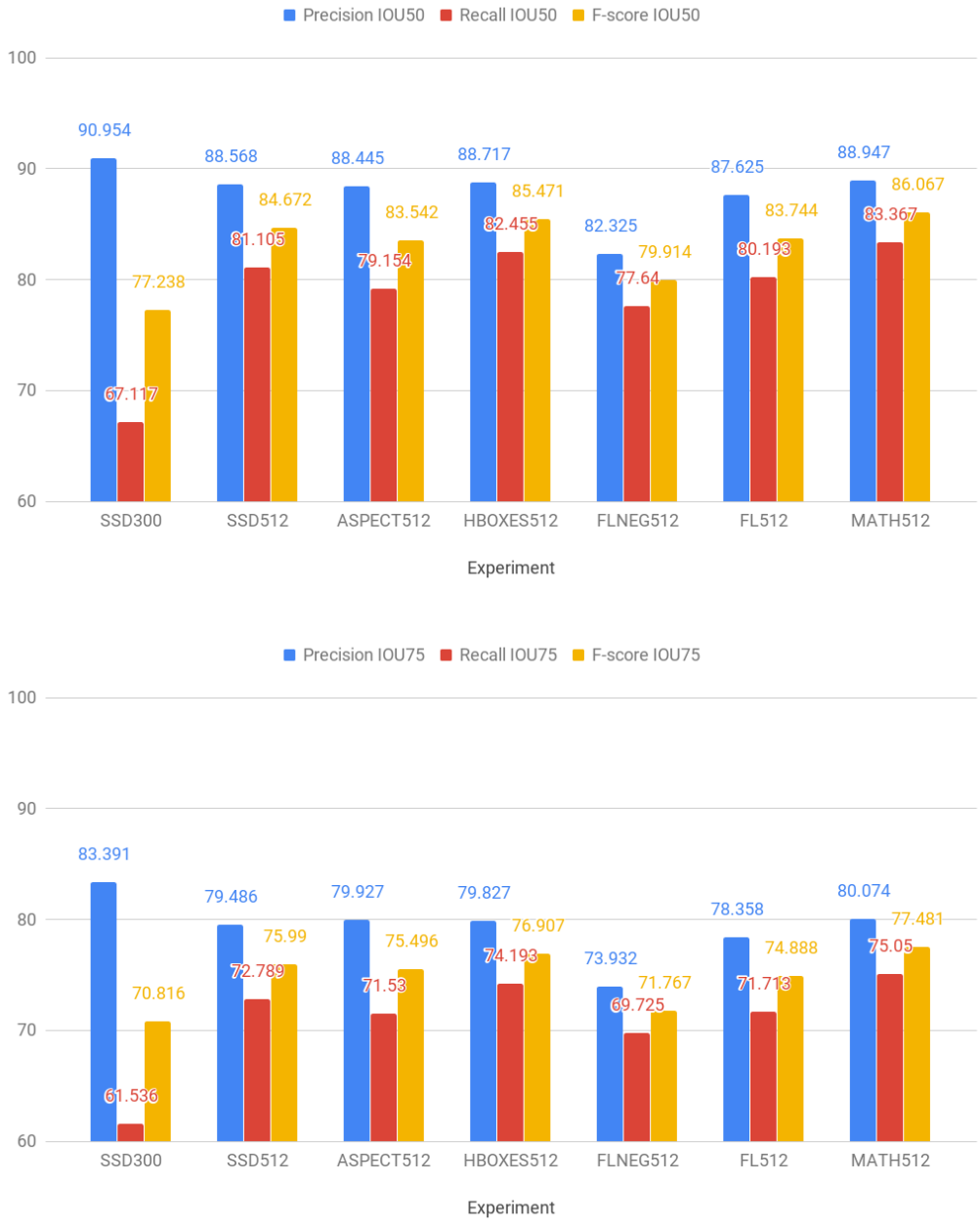


Figure 5.3: Formula detection results on the validation set for IOU50 and IOU75

boxes in case of HBOXES512 might have reduced the noise introduced by tall default boxes and helped HBOXES512 learn better than ASPECT512.

We used focal loss with negative mining for model FLNEG512 and used focal loss without negative mining for model FL512 instead of cross-entropy loss for calculating classification loss. As expected FL512 performed better than the FLNEG512 as the focal loss is designed to replace hard negative mining and FLNEG512 uses focal loss with hard negative mining.

Finally, we used MATH512 model which is the same as HBOXES512 except we replaced a kernel of size 3×3 with 1×5 in the confidence (CONF) and localization (LOC) layers in SSD architecture (refer to Figure 4.8 in Chapter 4). We observed that the MATH512 has the highest recall and second-highest precision (after SSD300). It has the highest F-score. One of the reasons can be that the use of a wider 1×5 kernel creates rectangular receptive fields which better represent math regions in the original image.

To summarize, we observed that focal loss does not perform better than cross-entropy with hard negative mining for our architecture. Focal loss performs worse if combined with hard negative mining. Use of a very large number of default boxes can affect performance because of noise introduced due to unnecessary default boxes. Aspect ratios of default boxes should be carefully chosen as they play an important role in the effective training of SSD. Use of kernel size in CONF and LOC layers of SSD (shown in Chapter 4, Figure 4.8) that creates the shape of receptive field similar to the shape of object that we want to detect results in performance improvement.

5.3.2 Character Level Detections

Figure 5.4 shows the character level results on the validation set. Characters completely inside detected bounding box D_C can be either text character D_T or math characters D_M. All the characters that are completely inside the ground truth bounding box are considered as a ground truth math characters G_M. Character level results can be calculated using Equation 5.1, Equation 5.2, and, Equation 5.3.

$$Precision = \frac{\mid D_M \mid}{\mid D_M \cup D_T \mid} \tag{5.1}$$

$$Recall = \frac{\mid D_M \mid}{\mid G_M \mid} \tag{5.2}$$

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.3}$$

Figure 5.4 shows the character level results. It can be seen that we get over 90% F-score for all the models. It is mainly because our method can accurately detect math characters. The F-score for formula detection is lower than character level detection because of splitting (detecting multiple bounding boxes for a math expression) and merging (detecting one bounding box for multiple math expressions).

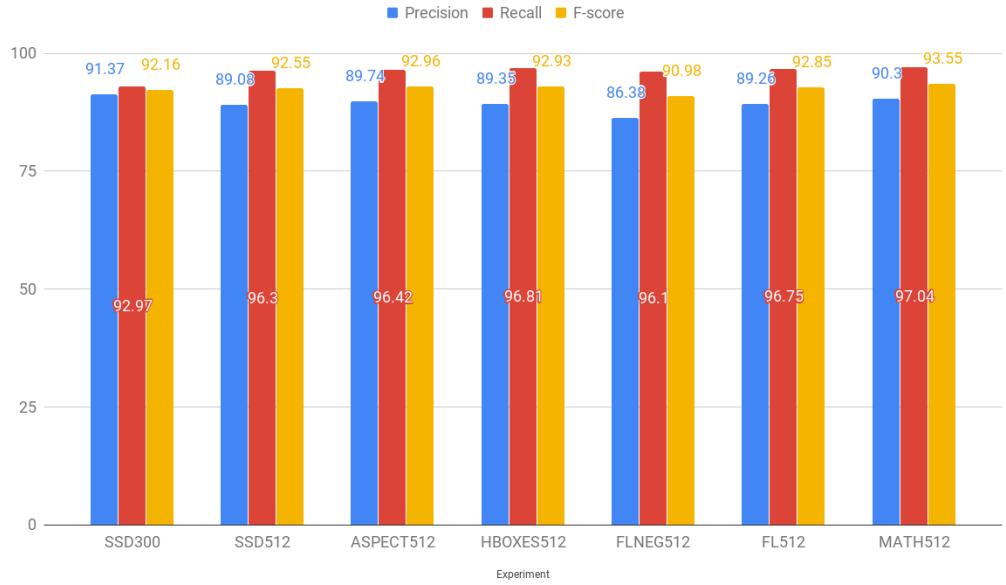


Figure 5.4: Character level results on the validation set

5.3.3 Single-character Vs. Multi-character Math Expression Detection

It can be seen from Figure 5.5 that the recall for single character math expressions is always lesser than the multi-character math expressions. All of our models clearly perform worse for single-character math expressions because they detect many single-digit numbers like references, footnotes, figure number etc. that are not math regions. Also, it is difficult to get IOU score more than the threshold for single character math regions as they are very small in size. It can also be seen that when we moved from input image size of 300×300 (in SSD300) to 512×512 (in other models) the recall for single character math expressions significantly improved. It is because we get more context while

detecting a single character.

5.4 Results on the Test Dataset

In this section, we provide the results on the test dataset. We trained MATH512 model with training and validation data for four epochs (165k iterations). The total number of sub-images with math regions generated using sliding windows from for training is 656,717. The number of sub-images generated using sliding windows from test dataset is 415,796. Our model took 24 hours, 18 minutes to train on two Nvidia GeForce GTX 1080 Ti. Our model operates at 27 frames per second (FPS) at inference time using one Nvidia GeForce GTX 1080 Ti GPU. If we assume about 1200 sub-images per page, it will take about 45 seconds per page.

5.4.1 Effect of Postprocessing

We perform two types of postprocessing: postprocessing-1 and postprocessing-2. In postprocessing-1, we adjust the detected boxes before pooling and in postprocessing-2, we adjust detected boxes after pooling. We studied the effect of both postprocessing on the detection results. It can be seen from Figure 5.6 that when we apply both postprocessing we get the highest F-score. When applied independently postprocessing-1 performs better than postprocessing-2. Postprocessing the detections definitely improves F-score, but on the other hand, takes additional time. The time taken per page by each type of postprocessing is shown in Figure 5.7. We observed that postprocessing-1 takes

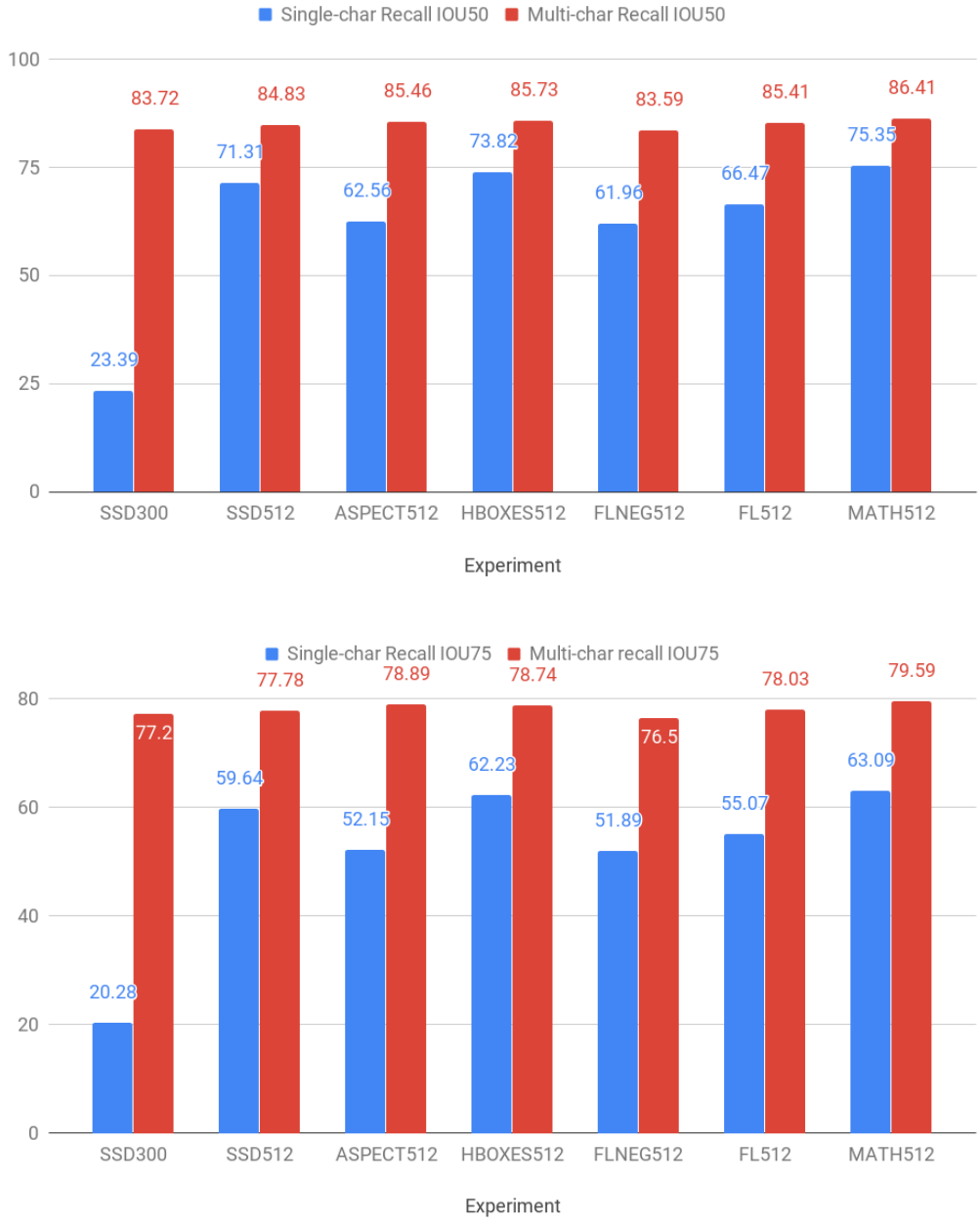


Figure 5.5: Single-character vs. multi-character math expression detection results on the validation set for IOU50 and IOU75

Figure 5.6: Effect of postprocessing

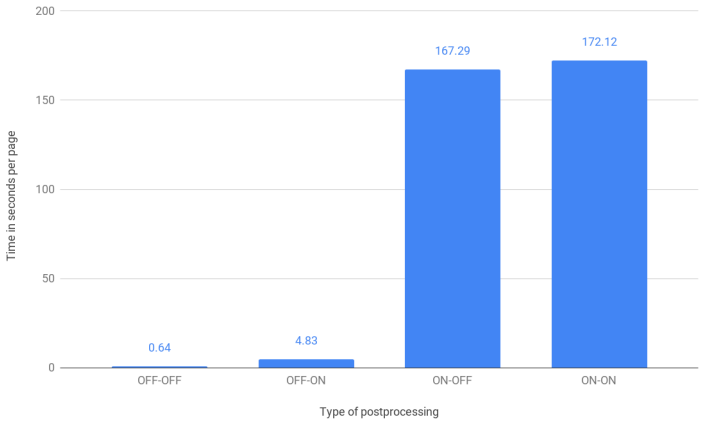


Figure 5.7: Time taken by postprocessing methods on test set

a significant amount of time. The labels on the x-axis shows which postprocessing was used (ON-ON means both the postprocessing were used, ON-OFF means just the first postprocessing was used, etc).

5.4.2 Final Results on Test Dataset

We would like to point out that in TFD-ICDAR2019 test dataset relations between the characters are given. It is possible to group math characters following their relationships with other characters just using a graph traversal method. The bounding box for a group of characters found this way represents a multi-character math region. It is not possible to detect single character math regions using this method as they do not have any relationships with other characters. A tree is generated when characters are grouped by following their relationships with other characters. Such a tree is called a symbol layout tree (SLT). We provide the results of this method in Figure 5.8.

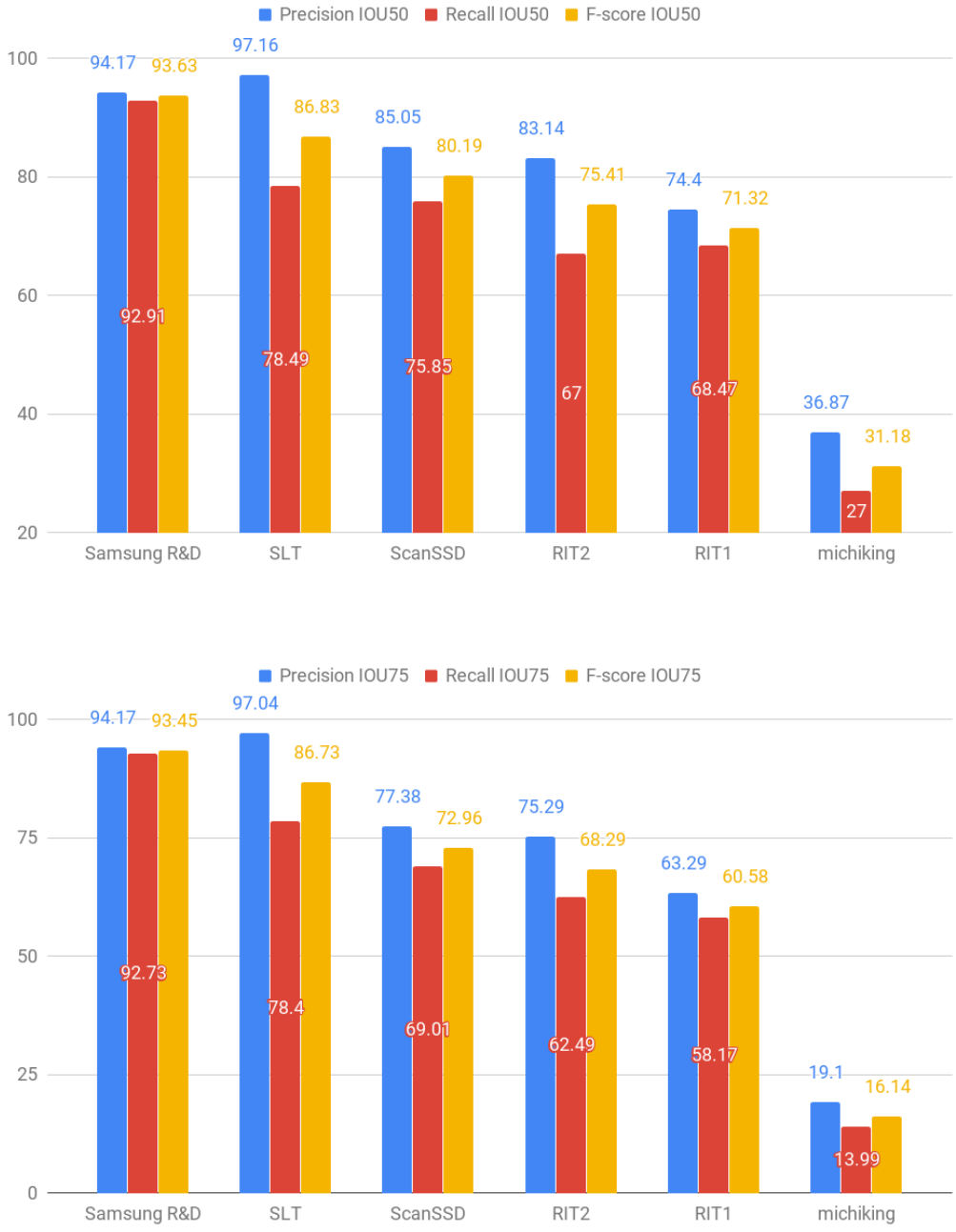


Figure 5.8: Comparision of ScanSSD with other systems on TFD-ICDAR2019 test dataset

We believe Samsung R&D used the relationships available in the dataset for finding the locations of multi-character regions and statistical methods for finding single character math regions. Our method that reads given relationships to form symbol layout tree (SLT) shows really good results. But, in the real world, the relationships between characters are generally not available and hence this thesis focuses on developing a method which works just from document images. RIT2 was an earlier version of our system submitted to ICDAR2019 competition. Similar to RIT2, RIT1 works from the images and uses YOLOv3 object detection for finding math regions. Our method MATH512 has about 5% increase in the F-score for IOU50 and about 4.5% increase in F-score for IOU75 as compared to RIT2.

5.4.3 Filewise Detection Results

Figure 5.9 shows filewise detection results. We obtained highest precision of 91.32% (Gidas79) and highest recall of 85.79% (Erbe94) for IOU50 and the highest precision of 87.69% (Gidas79) and highest recall of 83.61% (Erbe94) for IOU75 (The filename corresponding to each number is given in the parentheses). We obtained lowest precision of 79.13% (Katz99) and lowest recall of 62.88% (Emden76) for IOU50 and lowest precision of 68.29% (Emden) and lowest recall of 62.88% (Emden76) for IOU75. The highest F-score was 86.87% (Gidas79) and 84.76% (Erbe94) for IOU50 and IOU75, respectively. We observed the highest drop in the precision of 13.97% (Emden76) and highest drop in the recall of 11.36% (Kazhdan79) when the IOU threshold was

increased from 50% to 75%. We think that page layout plays an important role in the detection results. If lines are well-spaced and characters are not vertically close to each other, it reduces the number of cases where only one bounding box is detected for multiple math regions. For example, lines in Emden76 are very close to each other as compared to lines in the Erbe94. Hence, we get better F-score for Emden76 as compared to Erbe94. Also, if a document has more displayed math expressions than embedded math expressions we get more F-score.

5.4.4 IOU vs. F-score

Figure 5.10 shows the decline in the F-score with the increase in the IOU threshold. As expected the highest F-score is at lowest IOU threshold 0.05 and lowest F-score is at IOU threshold 1.0. The rate of decline in the F-score is high towards the high IOU threshold values. This graph shows, our method can detect math regions with 58.95 F-score for very strict IOU threshold of 0.95.

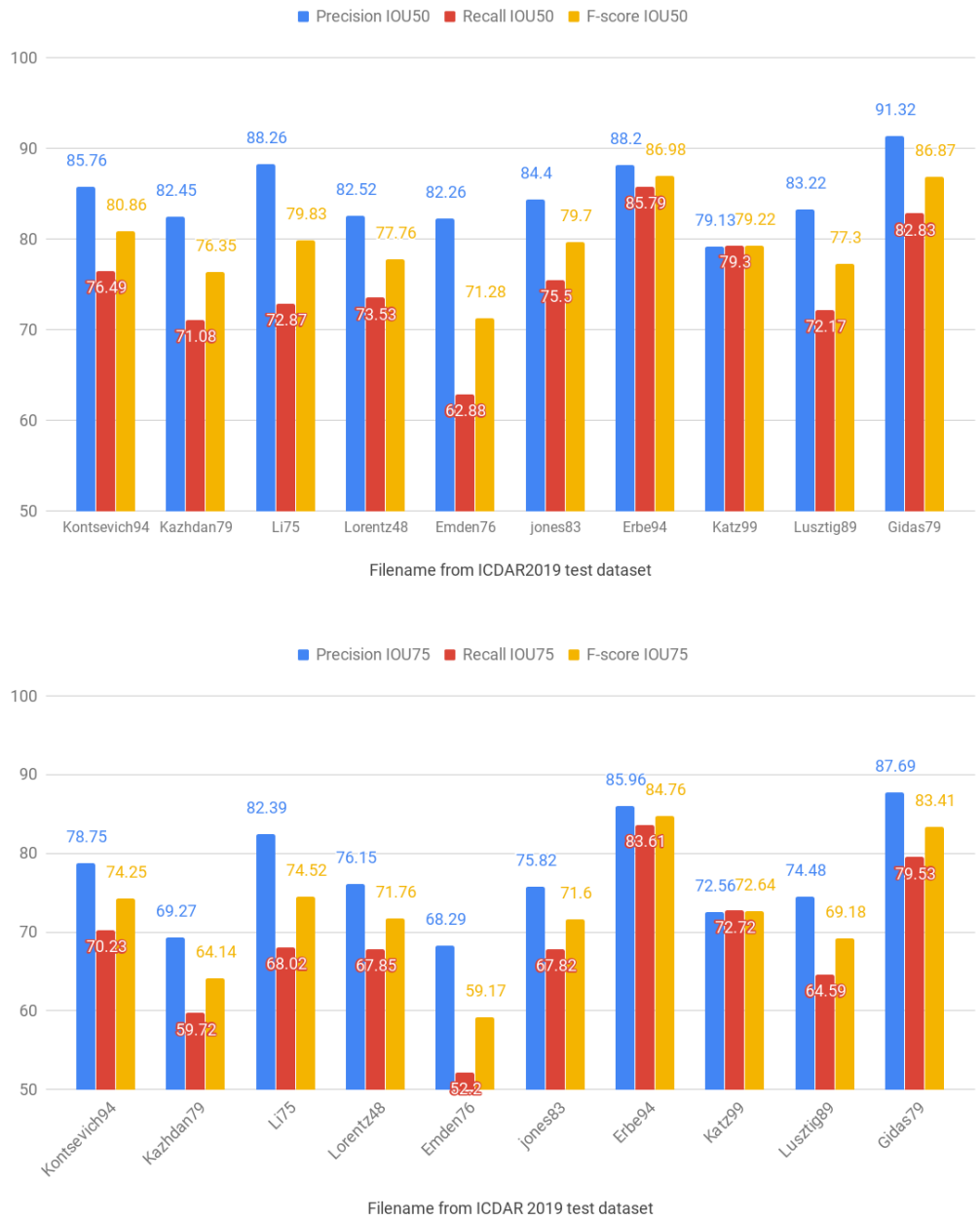


Figure 5.9: Filewise detection results on the test set for IOU50 and IOU75

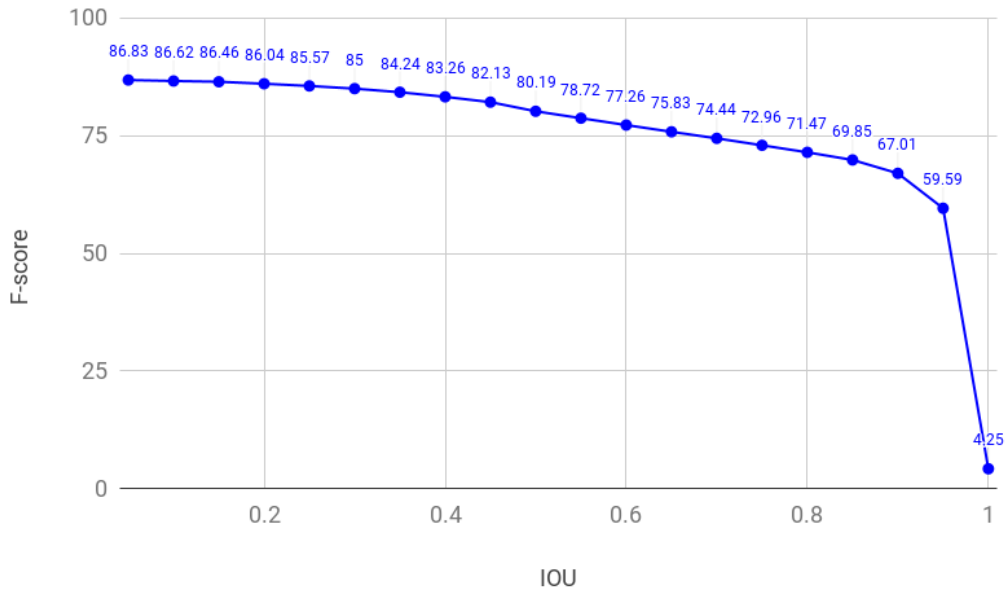


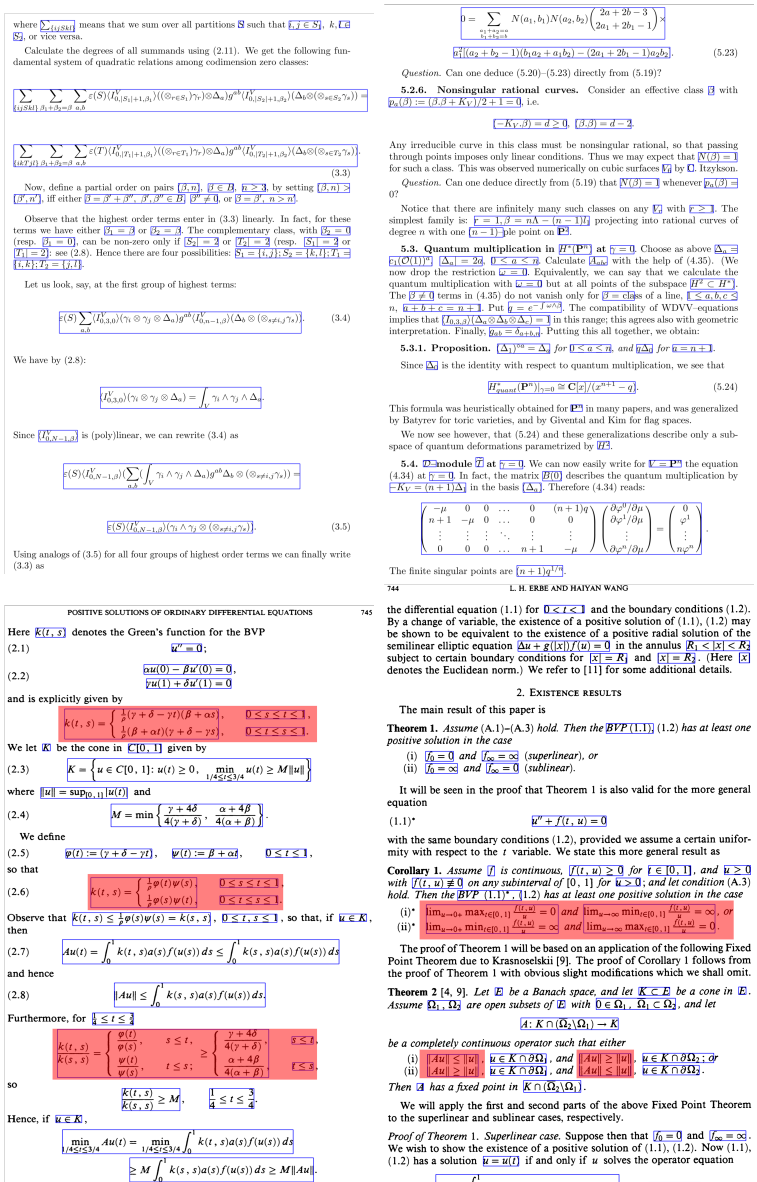
Figure 5.10: Decline in the F-score with increase in the IOU threshold

5.4.5 Examples of Positive Detections

Figure 5.11 shows math regions correctly detected by our method. Our method can detect math regions of arbitrary size, from one character to hundreds' of characters. Also, it detects matrices and correctly rejects equation numbers, page numbers and other numbers which are not math expressions.

5.4.6 Examples of Negative Detections

Figure 5.11 shows three cases where our method fails. First, when there is a large space between characters from the same math expression, our method generates multiple detections for one math expression (split de-



where $\sum_{\{ij|Sk\}}$ means that we sum over all partitions S such that $i,j \in S$, $k,l \in S$, or vice versa.

Calculate the degrees of all summands using (2.11). We get the following fundamental system of quadratic relations among codimension zero classes:

$$\sum_{\{ij|Sk\}} \sum_{\beta_1+\beta_2=\beta} \sum_{a,b} \varepsilon(S)\langle I^V_{0,|S_1|+1,\beta_1}\rangle((\otimes_{r\in S_1}\gamma_r)\otimes\Delta_a)g^{ab}\langle I^V_{0,|S_2|+1,\beta_2}\rangle(\Delta_b\otimes(\otimes_{s\in S_2}\gamma_s)) =$$

$$\sum_{\{kl|Tj\}} \sum_{\beta_1+\beta_2=\beta} \sum_{a,b} \varepsilon(T)\langle I^V_{0,|T_1|+1,\beta_1}\rangle((\otimes_{r\in T_1}\gamma_r)\otimes\Delta_a)g^{ab}\langle I^V_{0,|T_2|+1,\beta_2}\rangle(\Delta_b\otimes(\otimes_{s\in T_2}\gamma_s)). \tag{3.3}$$

Now, define a partial order on pairs $[\beta,n]$, $\beta \in B$, $n \geq 3$, by setting $[\beta,n] \succ [\beta',n']$, iff either $\beta = \beta'+\beta''$, $\beta',\beta'' \in B$, $\beta'' \neq 0$, or $\beta = \beta'$, $n > n'$.

Observe that the highest order terms enter in (3.3) linearly. In fact, for these terms we have either $\beta_1 = \beta$ or $\beta_2 = \beta$. The complementary class, with $\beta_2 = 0$ (resp. $\beta_1 = 0$), can be non-zero only if $|S_2| = 2$ or $|T_2| = 2$ (resp. $|S_1| = 2$ or $|T_1| = 2$); see (2.8). Hence there are four possibilities: $S_1 = \{i,j\}; S_2 = \{k,l\}; T_1 = \{i,k\}; T_2 = \{j,l\}$.

Let us look, say, at the first group of highest terms:

$$\varepsilon(S)\sum_{a,b}\langle I^V_{0,3,0}\rangle(\gamma_i\otimes\gamma_j\otimes\Delta_a)g^{ab}\langle I^V_{0,n-1,\beta}\rangle(\Delta_b\otimes(\otimes_{s\neq i,j}\gamma_s)). \tag{3.4}$$

We have by (2.8):

$$\langle I^V_{0,3,0}\rangle(\gamma_i\otimes\gamma_j\otimes\Delta_a) = \int_V \gamma_i\wedge\gamma_j\wedge\Delta_a.$$

Since $\langle I^V_{0,N-1,\beta}\rangle$ is (poly)linear, we can rewrite (3.4) as

$$\varepsilon(S)\langle I^V_{0,N-1,\beta}\rangle(\sum_{a,b}(\int_V \gamma_i\wedge\gamma_j\wedge\Delta_a)g^{ab}\Delta_b\otimes(\otimes_{s\neq i,j}\gamma_s)) =$$

$$\varepsilon(S)\langle I^V_{0,N-1,\beta}\rangle(\gamma_i\wedge\gamma_j\otimes(\otimes_{s\neq i,j}\gamma_s)). \tag{3.5}$$

Using analogs of (3.5) for all four groups of highest order terms we can finally write (3.3) as

$$0 = \sum_{\substack{a_1,b_1\geq 0\\ a_1+b_1=a}} N(a_1,b_1)N(a_2,b_2)\binom{2a+2b-3}{2a_1+2b_1-1}\times$$
$$\beta_1^2[(a_2+b_2-1)(b_1a_2+a_1b_2)-(2a_1+2b_1-1)a_2b_2]. \tag{5.23}$$

Question. Can one deduce (5.20)–(5.23) directly from (5.19)?

5.2.6. Nonsingular rational curves. Consider an effective class β with $p_a(\beta) := (\beta,\beta+K_Y)/2+1 = 0$, i.e.

$$(-K_Y,\beta) = d > 0, \quad (\beta,\beta) = d-2.$$

Any irreducible curve in this class must be nonsingular rational, so that passing through points imposes only linear conditions. Thus we may expect that $N(\beta) = 1$ for such a class. This was observed numerically on cubic surfaces Y_d by G. Ilzrykson.

Question. Can one deduce directly from (5.19) that $N(\beta) = 1$ whenever $p_a(\beta) = 0$?

Notice that there are infinitely many such classes on any Y_d with $d \geq 1$. The simplest family is: $r = 1, \beta = n\Lambda - (n-1)l_1$ projecting into rational curves of degree n with one $(n-1)$-ple point on $\mathbb{P}^2$.

5.3. Quantum multiplication in $H^*(\mathbb{P}^n)$ at $\tau = 0$. Choose as above $\Delta_a = c_1(\mathcal{O}(1))^a$, $\langle\Delta_a\rangle = 2a$, $0 < a < n$. Calculate $\langle I_{0,3,0}\rangle$ with the help of (4.35). (We now drop the restriction $\zeta = 0$. Equivalently, we can say that we calculate the quantum multiplication with $\omega = 0$ but at all points of the subspace $H^2 \subset H^*$.) The $\beta \neq 0$ terms in (4.35) do not vanish only for $\beta = cl$ of a line, $k \leq a,b,c \leq n$, $k+l+c = n+1$. Put $q = e^{-\int\omega/\hbar}$. The compatibility of WDVV-equations implies that $\langle I_{0,3,0}\rangle(\Delta_a\otimes\Delta_b\otimes\Delta_c) = 1$ in this range; this agrees also with geometric interpretation. Finally, $g_{ab} = \delta_{a+b,n}$. Putting this all together, we obtain:

5.3.1. Proposition. *$(\Delta_1)^{*a} = \Delta_a$ for $1 \leq a < n$, and $1\Delta_0$ for $a = n+1$.*

Since Δ_0 is the identity with respect to quantum multiplication, we see that

$$H^*_{quant}(\mathbb{P}^n)|_{\tau=0} \cong \mathbf{C}[x]/(x^{n+1}-q). \tag{5.24}$$

This formula was heuristically obtained for $\mathbb{P}^n$ in many papers, and was generalized by Batyrev for toric varieties, and by Givental and Kim for flag spaces.

We now see however, that (5.24) and these generalizations describe only a subspace of quantum deformations parametrized by H^2.

5.4. $\mathcal{D}$-module $\widetilde{H}$ at $\tau = 0$. We can now easily write for $V = \mathbb{P}^n$ the equation (4.34) at $\tau = 0$. In fact, the matrix $B(0)$ describes the quantum multiplication by $-K_V = (n+1)\Delta_1$ in the basis $\{\Delta_a\}$. Therefore (4.34) reads:

$$\begin{pmatrix} -\mu & 0 & 0 & \dots & 0 & (n+1)q \\ n+1 & -\mu & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & n+1 & -\mu \end{pmatrix}\begin{pmatrix} \partial\varphi^0/\partial\mu \\ \partial\varphi^1/\partial\mu \\ \vdots \\ \partial\varphi^n/\partial\mu \end{pmatrix} = \begin{pmatrix} 0 \\ \varphi^1 \\ \vdots \\ n\varphi^n \end{pmatrix}.$$

The finite singular points are $(n+1)q^{1/n}$.

Here $k(t,s)$ denotes the Green's function for the BVP

$$u'' = 0; \tag{2.1}$$
$$\alpha u(0) - \beta u'(0) = 0, \quad \gamma u(1) + \delta u'(1) = 0 \tag{2.2}$$

and is explicitly given by

$$k(t,s) = \begin{cases} \frac{1}{\rho}(\gamma+\delta-\gamma t)(\beta+\alpha s), & 0 \leq s \leq t \leq 1, \\ \frac{1}{\rho}(\beta+\alpha t)(\gamma+\delta-\gamma s), & 0 \leq t \leq s \leq 1. \end{cases}$$

We let K be the cone in $C[0,1]$ given by

$$K = \left\{u \in C[0,1]: u(t) \geq 0, \ \min_{1/4\leq t\leq 3/4} u(t) \geq M\|u\|\right\} \tag{2.3}$$

where $\|u\| = \sup_{[0,1]}|u(t)|$ and

$$M = \min\left\{\frac{\gamma+4\delta}{4(\gamma+\delta)}, \frac{\alpha+4\beta}{4(\alpha+\beta)}\right\}. \tag{2.4}$$

We define

$$\varphi(t) := (\gamma+\delta-\gamma t), \quad \psi(t) := \beta+\alpha t, \quad 0 \leq t \leq 1, \tag{2.5}$$

so that

$$k(t,s) = \begin{cases} \frac{1}{\rho}\varphi(t)\psi(s), & 0 \leq s \leq t \leq 1, \\ \frac{1}{\rho}\varphi(s)\psi(t), & 0 \leq t \leq s \leq 1. \end{cases} \tag{2.6}$$

Observe that $k(t,s) \leq \frac{1}{\rho}\varphi(s)\psi(s) = k(s,s)$, $0 \leq t,s \leq 1$, so that, if $u \in K$, then

$$Au(t) = \int_0^1 k(t,s)a(s)f(u(s))\,ds \leq \int_0^1 k(s,s)a(s)f(u(s))\,ds \tag{2.7}$$

and hence

$$\|Au\| \leq \int_0^1 k(s,s)a(s)f(u(s))\,ds. \tag{2.8}$$

Furthermore, for $\frac{1}{4} \leq t \leq \frac{3}{4}$

$$\frac{k(t,s)}{k(s,s)} = \begin{cases} \frac{\varphi(t)}{\varphi(s)}, & s \leq t, \\ \frac{\psi(t)}{\psi(s)}, & t \leq s; \end{cases} \geq \begin{cases} \frac{\gamma+4\delta}{4(\gamma+\delta)}, & s \leq t, \\ \frac{\alpha+4\beta}{4(\alpha+\beta)}, & t \leq s. \end{cases}$$

so

$$\frac{k(t,s)}{k(s,s)} \geq M, \qquad \frac{1}{4} \leq t \leq \frac{3}{4}.$$

Hence, if $u \in K$,

$$\min_{1/4\leq t\leq 3/4} Au(t) = \min_{1/4\leq t\leq 3/4}\int_0^1 k(t,s)a(s)f(u(s))\,ds$$
$$\geq M\int_0^1 k(s,s)a(s)f(u(s))\,ds \geq M\|Au\|.$$

the differential equation (1.1) for $0 < t < 1$ and the boundary conditions (1.2). By a change of variable, the existence of a positive solution of (1.1), (1.2) may be shown to be equivalent to the existence of a positive radial solution of the semilinear elliptic equation $\Delta u + g(|x|)f(u) = 0$ in the annulus $R_1 < |x| < R_2$ subject to certain boundary conditions for $|x| = R_1$ and $|x| = R_2$. (Here $|x|$ denotes the Euclidean norm.) We refer to [11] for some additional details.

2. Existence results

The main result of this paper is

Theorem 1. *Assume* (A.1)–(A.3) *hold. Then the BVP* (1.1), (1.2) *has at least one positive solution in the case*

(i) $f_0 = 0$ *and* $f_\infty = \infty$ (*superlinear*), *or*
(ii) $f_0 = \infty$ *and* $f_\infty = 0$ (*sublinear*).

It will be seen in the proof that Theorem 1 is also valid for the more general equation

$$u'' + f(t,u) = 0 \tag{1.1}*$$

with the same boundary conditions (1.2), provided we assume a certain uniformity with respect to the t variable. We state this more general result as

Corollary 1. *Assume f is continuous, $f(t,u) \geq 0$ for $t \in [0,1]$, and $u \geq 0$ with $f(t,u) \not\equiv 0$ on any subinterval of $[0,1]$ for $u \geq 0$; and let condition* (A.3) *hold. Then the BVP* (1.1)*, (1.2) *has at least one positive solution in the case*

(i)* $\lim_{u\to 0+}\max_{t\in[0,1]}\frac{f(t,u)}{u} = 0$ *and* $\lim_{u\to\infty}\min_{t\in[0,1]}\frac{f(t,u)}{u} = \infty$, *or*
(ii)* $\lim_{u\to 0+}\min_{t\in[0,1]}\frac{f(t,u)}{u} = \infty$ *and* $\lim_{u\to\infty}\max_{t\in[0,1]}\frac{f(t,u)}{u} = 0$.

The proof of Theorem 1 will be based on an application of the following Fixed Point Theorem due to Krasnoselskii [9]. The proof of Corollary 1 follows from the proof of Theorem 1 with obvious slight modifications which we shall omit.

Theorem 2 [4, 9]. *Let E be a Banach space, and let $K \subset E$ be a cone in E. Assume Ω_1, Ω_2 are open subsets of E with $0 \in \Omega_1$, $\bar{\Omega}_1 \subset \Omega_2$, and let*

$$A: K \cap (\bar{\Omega}_2 \setminus \Omega_1) \to K$$

be a completely continuous operator such that either

(i) $\|Au\| \leq \|u\|$, $u \in K \cap \partial\Omega_1$, *and* $\|Au\| \geq \|u\|$, $u \in K \cap \partial\Omega_2$; *or*
(ii) $\|Au\| \geq \|u\|$, $u \in K \cap \partial\Omega_1$, *and* $\|Au\| \leq \|u\|$, $u \in K \cap \partial\Omega_2$.

Then A has a fixed point in $K \cap (\bar{\Omega}_2 \setminus \Omega_1)$.

We will apply the first and second parts of the above Fixed Point Theorem to the superlinear and sublinear cases, respectively.

Proof of Theorem 1. Superlinear case. Suppose then that $f_0 = 0$ and $f_\infty = \infty$. We wish to show the existence of a positive solution of (1.1), (1.2). Now (1.1), (1.2) has a solution $u = u(t)$ if and only if u solves the operator equation

$$u(t) = \int_0^1 k(t,s)a(s)f(u(s))\,ds := Au(t), \qquad u \in C[0,1].$$

Figure 5.11: Examples of positive and negative detections. Invalid detections are highlighted in red

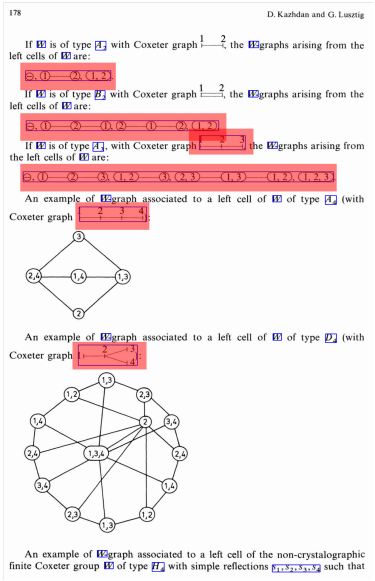


Figure 5.12: Math regions detected by our method. Invalid detections are highlighted in red

tections). Second, when math expressions are very close to each other, our method detects only one box for multiple expressions (merged detections). We observed that most of the failure cases are because of split detections or merged detections. Even though we are detecting math regions in both merged or split detections, precision and recall go down as our evaluation criteria use IOU threshold. Hence, we also provide character level results in Section 5.4.7 and Section 5.4.8. Third, sometimes our method detects wide and embedded graphs (visually similar to functions) as math expressions. Examples of this case are shown in Figure 5.12.

Detection results for all pages from one PDF are given in Appendix B.

5.4.7 Character Level Detections

We found that our method detects almost all the math characters present in the documents. Figure 5.13 shows character level results for MATH512. Our architecture is good at detecting math characters, i.e., it is good at classification, but not as good at localizing the math expressions. The recall of our method is about 96.62% and precision is 90.36% (i.e. 90% of detected characters are math character). The F-score at the character level is 91.44% and it drops by 11.25% for math expression detection for IOU50 and 18.48% for IOU75.

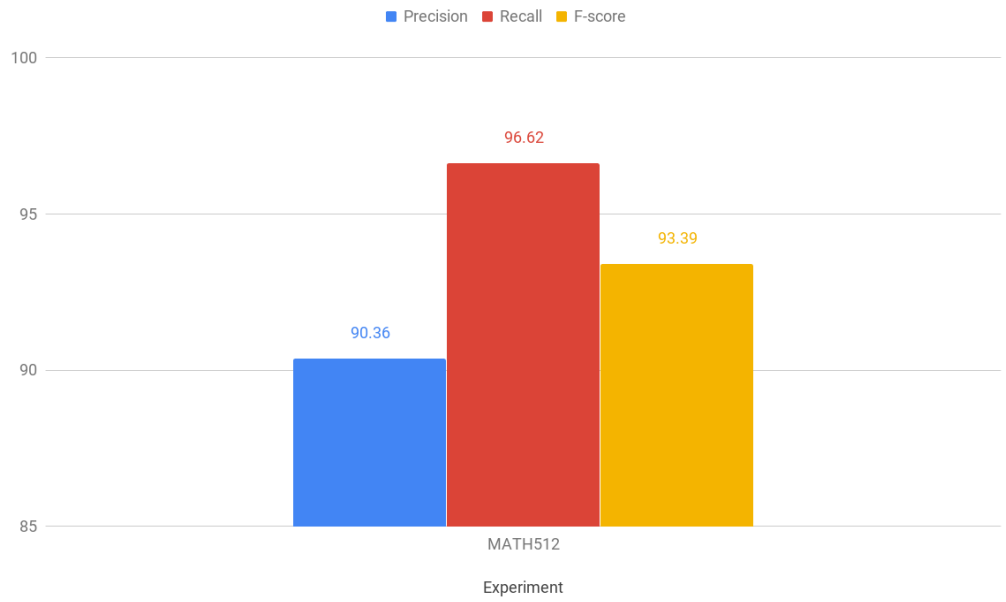


Figure 5.13: Character level results for test set

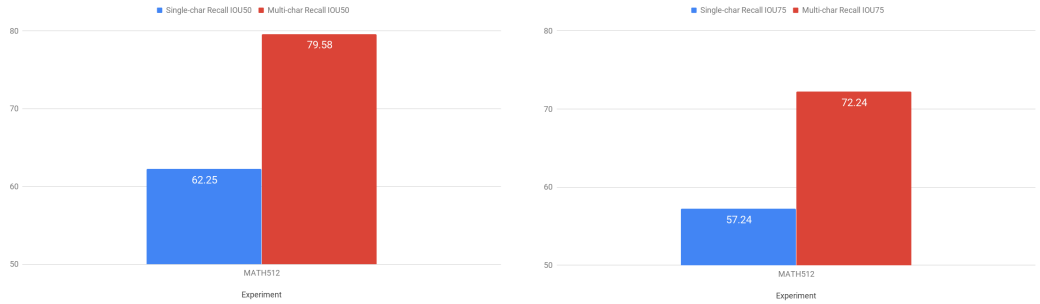


Figure 5.14: Single-character vs. multi-character math expression detection results on the test dataset for IOU50 and IOU75

5.4.8 Single-character Vs. Multi-character Math Expression Detection

ScanSSD clearly performs better for multi-character math regions. It can be seen from Figure 5.14 that, ScanSSD achieves 17.33% more recall for multi-character math regions for IOU50 and 15% more recall for multi-character math regions for IOU75. As explained earlier in Section 5.3.3, our model clearly performs worse for single-character math expressions because it detects many single-digit numbers like references, footnotes, figure number, etc. that are not math regions. Also, it is difficult to get the IOU score more than the threshold for single character math regions as they are very small in size.

Few applications like indexing research documents for math formula search that might not require single character math detection can train the same model ignoring the single character math expressions and can get better performance.

5.5 Additional Examples

We use pages from Chapter 4, Section 4.3.1 and give them as input to ScanSSD. Figure 5.15 shows results obtained by ScanSSD. Our method detects most of the math regions from these pages. It does not have any problem in detecting displayed math expressions. Few of the single character math regions are not detected. We observe the same behavior on ICDAR 2019 test dataset, where results for multi-character math regions for our method are better than single-character math regions. One of the reasons that single-character math regions were missed is that they do not have any distinguishing property like the bold font that will differentiate them from text characters. Another reason can be that the single-character regions are less wide.

5.6 Summary

To summarize, we observed that:

1. Focal loss does not perform better than cross-entropy with hard negative mining for our architecture.

2. Focal loss performs worse if combined with hard negative mining.

3. Use of a very large number of default boxes can affect performance because of noise introduced due to unnecessary default boxes.

4. Aspect ratios of default boxes should be carefully chosen based on the object to be detected as default boxes play an important role in the

Bounding Box Regression Let us define the predicted box l as $l = l^{cx}, l^{cy}, l^{w}, l^{h}$, where superscript cx, cy is the x and y co-ordinate of the center of the box, w is the width, and, h is the height. Similarly, the ground truth bounding box g can be defined as $l = l^{cx}, l^{cy}, l^{w}, l^{h}$. We configure the bounding box regressor to learn four functions, f_{cx}, f_{cy} for scale-invariant transformation between centers of l and g and f_w, f_h for log-scale transformation between widths and heights. In general, we define f_i, where $i \in cx, cy, w, h$ as a bounding box correction function. f_i takes l as the input. The target values $\hat{g}$ for bounding box correction functions f_i are defined below:

$$\begin{aligned} \hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^{w} \\ \hat{g}_j^{cy} &= (g_j^{cy} - d_i^{cy})/d_i^{h} \\ \hat{g}_j^{w} &= \log(\frac{g_j^{w}}{d_i^{w}}) \\ \hat{g}_j^{h} &= \log(\frac{g_j^{h}}{d_i^{h}}) \end{aligned} \tag{4.5}$$

We use $Smooth_{L1}$ loss defined by Girshick[19] for the bounding box regression problem. It is given by Equation 4.6. It is claimed to be less sensitive to outliers.

$$smooth_{L1}(t) = \begin{cases} 0.5t^2, & \text{if } \mid t \mid < 1 \\ \mid t \mid -0.5, & \text{otherwise} \end{cases} \tag{4.6}$$

Localization Loss The localization loss is the smooth L1 loss [12] between the predicted box(l) and the ground truth box(g). We calculate the overall localization loss as:

$$L_{loc}(x, l, g, f) = \sum_{i \in POS}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(f_m(l_i) - \hat{g}_j^{m}) x_{ij}^{k} smooth_{L1}(f_m(l_i) - \hat{g}_j^{m}) \tag{4.7}$$

where POS are positive examples.POS and NEG (negative examples) are defined inthe Section 4.2.5. x is the indicator function, l is the predicted box, g is the ground truth, and, f_i is the set of bounding box correction functions. From Equation 4.7 we can see that the localization lossis calculated only for positive examples.

Confidence Loss The confidence loss is the cross entropy loss over multiple class confidences c. It is given by Equation 4.8.

$$\begin{aligned} L_{conf}(x, c) &= -\sum_{i \in POS}^{N} x_{ij}^{p} \log \hat{c}_i^{p} - \sum_{i \in NEG} \log(\hat{c}_i^{0}) \\ &\text{where,} \\ \hat{c}_i^{p} &= exp(c_i^{p}) / \sum_{p} exp(c_i^{p}) \end{aligned} \tag{4.8}$$

Overall Loss The overall objective loss function used in our model is given by Equation 4.9:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{4.9}$$

where N is the number of matched default boxes. If N is 0,the lossis set to 0. We set the weightterm α to 1 like original SSD[37].

4.3.2 Dealing with Class Imbalance

In this section we discuss two techniques we used to address the class imbalance problem we get from having many more non-math than math regions during training: focal loss and hard negative mining.

4.3.2.1 Focal Loss

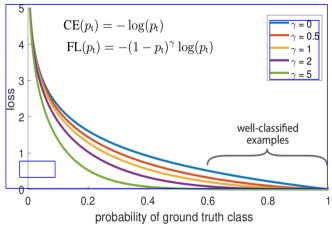


Figure 4.9: Behavior of focal loss function for different values of γ. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > 0.5$), putting more focus on hard, misclassified examples. The blue line is the CE loss. The purple line shows the curve for the FL with $\gamma = 2$ which was found to be working best by Lin et al [32]. This figure is taken from Lin et al. [32].

The original SSD architecture[37] used the cross-entropy (CE) function (refer to Equation 4.8) for confidence loss calculation. Cross-entropy is defined by Equation 4.10. Here $y \in \{\pm 1\}$ which is the ground truth class and $p \in [0, 1]$ is the model's estimated probability.

$$CE(p, y) = \begin{cases} -\log p, & \text{if } y = 1 \\ -\log(1 - p), & \text{otherwise} \end{cases} \tag{4.10}$$

The FL is modification of CE. Lin et al.[32] define p_t as given in Equation 4.11 and rewrite the cross-entropy function (Equation 4.10) as Equation 4.12. The FL is defined in the Equation 4.13.

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases} \tag{4.11}$$

$$CE(p, y) = CE(p_t) = -\log p_t \tag{4.12}$$

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{4.13}$$

If $\gamma = 2$ and $p_t = 0.9$, then FL will be 100 times lower than the CE. When $p_t = 0.968$, FL will be 1000 times lower than the CE. This allows the network to focus more on correcting the misclassified examples. Also, when $\gamma = 0$, FL is equivalent to CE. Figure 4.9 shows the behavior of FL for different γ values. We call $(1 - p_t)^{\gamma}$ a modulating factor. As $p_t \to 1$, the $(1 - p_t)^{\gamma} \to 0$ and effect of easy examples is down-weighted. The focusing factor γ smoothly adjusts

Figure 5.15: Detections in the real world document images. These images are from this thesis Chapter 4.

effective training of SSD.

5. Use of kernel size in CONF and LOC layers of SSD (shown in Chapter 4, Figure 4.8) that creates the shape of receptive field similar to the shape of the object we want to detect results in performance improvement.

6. Our model performs better for multi-character math expression detection.

7. Many failure cases for formula detection are because of the split and merge of the bounding boxes as explained in Section 5.4.6.

8. We obtain very good detection results at character level that suggests improvement in the pooling methods will improve the overall F-score for formula detection.

Chapter 6

Conclusion

We introduced ScanSSD architecture for detecting both embedded and displayed math expressions in document images using a single-stage network that does not require page layout, font, or, character information. We used sliding windows to generate sub-images of a large document page image rendered at 600 dpi and applied single shot detector (SSD) on each sub-image. We introduced pooling methods based on the confidence scores and density of detections to generate page-level results.

Liu et al. [38] discussed that SSD struggles to detect small objects. Also, we found that the conversion of a page image rendered at 600 dpi to lower sizes lost a large amount of information. In our initial experiments, SSD missed many math expressions when we used lower resolution images as input. Hence, for SSD to detect the math expressions effectively we use sliding windows to generate sub-images so that math expression cover a relatively bigger area of input images instead of using images with lower resolution where math expressions cover a smaller area of images.

Through our experiments described in Chapter 4, Table 3.1, we observed that:

1. Focal loss does not perform better than cross entropy with hard negative mining for our architecture where maximum number of default boxes used is 65,532. Focal loss performs worse if combined with hard negative mining.

2. Use of a very large number of default boxes can affect performance because of noise introduced due to unnecessary default boxes. Aspect ratios of default boxes should be carefully chosen based on the object to be detected as default boxes play an important role in the effective training of SSD.

3. Use of kernel size in CONF and LOC layers of SSD (shown in Chapter 4, Figure 4.8) that creates the shape of receptive field similar to the shape of the object we want to detect results in performance improvement.

We combined our findings in MATH512 model that uses cross-entropy with hard negative mining with only wide default boxes and kernel of size 1×5 and obtained the best results.

An earlier version of ScanSSD placed 2^{nd} in the ICDAR 2019 competition on the Typeset Formula Detection (TFD). ScanSSD achieves 80.19% F-score at IOU50 and 72.96% F-score at IOU75 on ICDAR 2019 test dataset. For very strict IOU threshold of 95% we get 59.59% F-score. Also, we get 90.36% precision and 96.62% recall at character level. Hence we conclude that our hypothesis, both embedded and displayed math expressions in typeset document images can be detected accurately in one framework using deep

learning-based object detection methods without the use of additional information, was confirmed.

6.1 Contributions

New dataset for math detection. We created a new dataset for math expression detection and made it available to the public at `https://github.com/MaliParag/ICDAR2019`. This dataset contains annotations for both embedded and displayed math expressions. It also provides character locations and OCR codes. Our dataset was used for Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection (CROHME + TFD 2019) at the 15th International Conference on Document Analysis and Recognition (ICDAR 2019) [40]. After the competition, we fixed a few problems with the ICDAR 2019 dataset and released version 2 of the same dataset. In Chapter 3, we describe the dataset creation process and statistics for both versions 1 and 2 of the dataset.

ScanSSD architecture for math expression detection. In Chapter 4, we introduced an architecture for math expression detection. Our method just requires document page image as input and does not require additional information like character labels, character locations, page layout information, etc. like other math detection method discussed in the Chapter 2. Our method learns the features while training and uses those features to predict the locations of the math regions. We do not rely on the manually designed features or

algorithms like OCR, XY-cut or projection profile cutting to find the locations of the math regions. It uses sliding windows to generate small sub-images from the document image and applies SSD over each sub-image to detect math regions. Then we use a polling method on the math regions detected from the sub-images to generate bounding boxes for the math expressions at the page level. In our modular architecture, we have a sub-image generation module, an object detection module, a pooling module, and, a postprocessing module. An earlier version of ScanSSD placed 2^{nd} in the ICDAR 2019 competition on the Typeset Formula Detection (TFD). Our code is publicly available at `https://github.com/MaliParag/ScanSSD`.

6.2 Future Work

We discuss future work in two parts: improving performance and reducing the time taken for detection.

6.2.1 Improving Detections

In this section, we describe different methods that we would like to try to improve detection results.

Pooling Methods One of the key differences in math object detection in the typeset documents and object detection in the natural scenes is that the objects in the typeset documents will not be occluded by the other objects. We believe this property will help us design better algorithm for non-maximal

supression as original non-maximal supression algorithm is designed for natural scenes where objects might overlap. Also, we observed that there are many merged boxes. It is because of the pooling methods. We would like to use the modified version of the pooling methods based on agglomerative clustering like a fusion algorithm introduced by Yu et al.[55]. We believe that improving the pooling methods will reduce the number of merged detections and improve both precision and recall.

Ensemble Models In our experiments, we observed that the smaller input image size 300×300 achieved at least 2% more precision than any other method with a bigger input image size of 512×512, but the recall is at least 10% less when we used input image size 300×300. The higher precision for input image size 300×300 might be because detected number of bounding boxes is less as compared to image size 512×512. We would like to use both create an ensemble of models for better detection. Also, we can generate a multi-scale input image pyramid and get predictions from multiple image scales and pool them later. It will slow down the detection process but might improve the detection results.

Modifications in SSD Architecture We would like to explore different layouts of the default boxes, different sizes of feature maps, different base networks, etc. in SSD architecture. Deeper base networks like Resnet-101 [25] can be used as a base network in SSD to get better features. But, it will slow down the detection process.

Using additional information. We believe that the math expression detection problem can be solved without any additional information like character locations and labels. If additional information is available, we think we can develop simpler and faster detection models. But, these models will have use limited to detecting math expressions in documents where additional information is available. ICDAR 2019 dataset provides character locations and labels and it will be a good starting point in developing such a model.

6.2.2 Speed-up

In this section, we discuss methods that we would like to try to reduce the time taken by the overall detection process.

Fewer sub-images while testing In ScanSSD architecture described in Chapter 4, we used a stride of 10% while generating sub-images using sliding windows for training and testing. We would like to try stride over 10% so we get less number of sub-images during inference. It will result in a faster method for detection.

Smaller Base Network in SSD We used VGG16 base network in ScanSSD, we think we can get similar results with specially designed smaller networks. Use of smaller networks will speed-up the detction process.

6.2.3 End Goal

End to End Training In our current architecture, we train the model to detect the math expressions in the sub-images generated using sliding windows and then use a fixed pooling method. We would like to design an architecture where we can train the model end-to-end so it can learn the pooling method itself.

Multiple Object Detection ScanSSD allows the use of multiple classes. While detecting multiple objects, each page object can be assigned a class.

Search Engine We would like to extract and index page objects in documents. We can use the generated index to retrieve related documents in search engines.

Appendices

Appendix A

Validation Dataset

We split the ICDAR 2019 training data into approximately 80% and 20%. Table A.1 shows the training and validation split. 116 out of 569 pages are used for validation.

Training set	**Pages**	**Validation pages**
Burstall77	24	23,20,3,5,15
BAMS_1998_123_143	21	2,17,12,11
AIF_1999_375_404	30	16,17,18,9,24,15
ASENS_1997_367_384	18	11,17,7,6
Brezis83	5	5
MA_1977_275_292	18	4,2,5,7
Borcherds86	4	2
BAMS_1971_1974_1	3	1
BAMS_1971_1974_2	4	4
MA_1999_175_196	22	19,1,10,16
JMS_1975_497_506	10	6,3
JMKU_1971_377_379	3	3
BAMS_1971_1974_3	4	1
AnnM_1970_550_569	20	7,18,8,17
AIF_1970_493_498	6	1
JMS_1975_281_288	8	8,7
TMJ_1990_163_193	32	1,23,31,28,5,9
TMJ_1973_317_331	16	7,11,16
MA_1970_26_38	13	7,1,12
InvM_1999_163_181	19	18,10,12,2
InvM_1970_121_134	14	9,3,2
BSMF_1970_165_192	28	22,20,7,27,23,13
ActaM_1998_283_305	23	15,17,20,6,10
ASENS_1970_273_284	12	5,7
TMJ_1973_333_338	6	2
Cline88	15	14,9,11
ActaM_1970_37_63	27	21,19,8,27,23
JMS_1975_289_293	5	4
BSMF_1998_245_271	27	11,1,25,3,10
Alford94	20	10,11,5,13
KJM_1999_17_36	20	11,6,3,10
JMKU_1971_181_194	14	8,14,1
Bergweiler83	37	18,11,9,24,34,13,1
Arkiv_1997_185_199	15	9,6,2
Arkiv_1971_141_163	23	16,7,1,10,5
JMKU_1971_373_375	3	2

Table A.1: Training and validation split.

Appendix B

Whole page detection results

Following pages show whole page detection results for one PDF Lifile from the dataset.

PERIOD THREE IMPLIES CHAOS

TIEN-YIEN LI AND JAMES A. YORKE

1. Introduction. The way phenomena or processes evolve or change in time is often described by differential equations or difference equations. One of the simplest mathematical situations occurs when the phenomenon can be described by a single number as, for example, when the number of children susceptible to some disease at the beginning of a school year can be estimated purely as a function of the number for the previous year. That is, when the number x_{n+1} at the beginning of the $n+1$st year (or time period) can be written

$$x_{n+1} = F(x_n), \tag{1.1}$$

where F maps an interval I into itself. Of course such a model for the year by year progress of the disease would be very simplistic and would contain only a shadow of the more complicated phenomena. For other phenomena this model might be more accurate. This equation has been used successfully to model the distribution of points of impact on a spinning bit for oil well drilling, as mentioned in [**8, 11**], knowing this distribution is helpful in predicting uneven wear of the bit. For another example, if a population of insects has discrete generations, the size of the $n+1$st generation will be a function of the nth. A reasonable model would then be a generalized logistic equation

$$x_{n+1} = rx_n[1 - x_n/K]. \tag{1.2}$$

A related model for insect populations was discussed by Utida in [**10**]. See also Oster *et al* [**14, 15**].

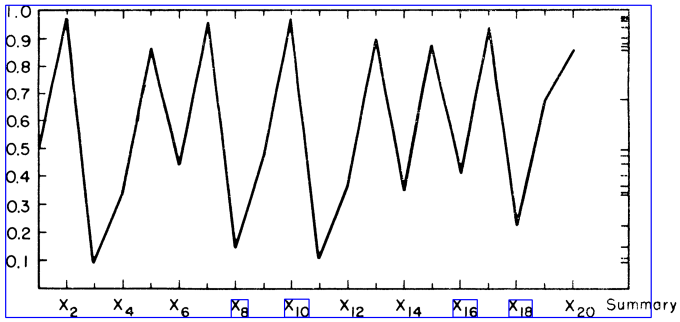


FIG. 1. For $K = 1$, $r = 3.9$ with $x_1 = .5$, the above graph is obtained by iterating Eq. (1.2) 19 times. At right the 20 values are repeated in summary. No value occurs twice. While $x_2 = .975$ and $x_{10} = .973$ are close together, the behavior is not periodic with period 8 since $x_{18} = .222$.

These models are highly simplified, yet even this apparently simple equation (1.2) may have surprisingly complicated dynamic behavior. See Figure 1. We approach these equations with the viewpoint that irregularities and chaotic oscillations of complicated phenomena may sometimes be understood in terms of the simple model, even if that model is not sufficiently sophisticated to allow accurate numerical predictions. Lorenz [**1–4**] took this point of view in studying turbulent behavior in a fascinating series of papers. He showed that a certain complicated fluid flow could be modelled

by such a sequence $x, F(x), F^2(x), \cdots$, which retained some of the chaotic aspects of the original flow. See Figure 2. In this paper we analyze a situation in which the sequence $\{F^n(x)\}$ is non-periodic and might be called "chaotic." Theorem 1 shows that chaotic behavior for (1.1) will result in any situation in which a "population" of size x can grow for two or more successive generations and then having reached an unsustainable height, a population bust follows to the level x or below.

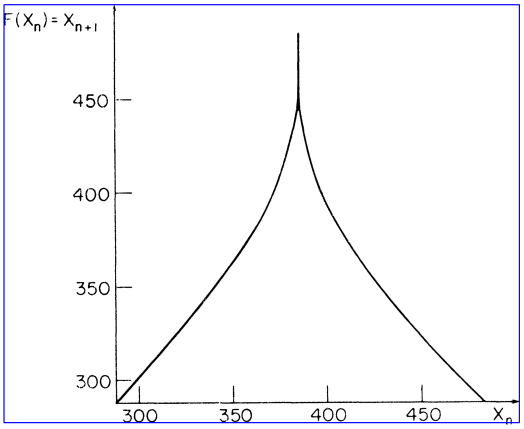


FIG. 2. Lorenz [1] studied the equations for a rotating water-filled vessel which is circularly symmetric about its vertical axis. The vessel is heated near the rim and cooled near its center. When the vessel is annular in shape and the rotation rate high, waves develop and alter their shape irregularly. From a simplified set of equations solved numerically, Lorenz let X_n be in essence the maximum kinetic energy of successive waves. Plotting X_{n+1} against X_n, and connecting the points, the above graph is obtained.

In section 3 we give a well-known simple condition which guarantees that a periodic point is stable and then in section 4 we quote a result applicable when F is like the one in Figure 2. It implies that there is an interval $J_s \subset J$ such that for almost every $x \in J$, the set of limit points of the sequence $\{F^n(x)\}$ is J_s.

A number of questions remain unanswered. For example, is the closure of the periodic points an interval or at least a finite union of intervals? Other questions are mentioned later.

Added in proof. May has recently discovered other strong properties of these maps in his independent study of how the behavior changes as a parameter is varied [17].

2. The main theorem. Let $F: J \to J$. For $x \in J$, $F^0(x)$ denotes x and $F^{n+1}(x)$ denotes $F(F^n(x))$ for $n = 0, 1, \cdots$. We will say p is a **periodic point with period** n if $p \in J$ and $p = F^n(p)$ and $p \neq F^k(p)$ for $1 \leq k < n$. We say p is **periodic** or is a **periodic point** if p is periodic for some $n \geq 1$. We say q is **eventually periodic** if for some positive integer m, $p = F^m(q)$ is periodic. Since F need not be one-to-one, there may be points which are eventually periodic but are not periodic. Our objective is to understand the situations in which iterates of a point are very irregular. A special case of our main result says that if there is a periodic point with period 3, then for each integer $n = 1, 2, 3, \cdots$, there is a periodic point with period n. Furthermore, there is an uncountable subset of points x in J which are not even "asymptotically periodic."

THEOREM 1. *Let J be an interval and let $F: J \to J$ be continuous. Assume there is a point $a \in J$ for which the points $b = F(a)$, $c = F^2(a)$ and $d = F^3(a)$, satisfy*

$$d \le a < b < c \quad (\text{or } d \ge a > b > c).$$

Then

T1: *for every $k = 1, 2, \cdots$ there is a periodic point in J having period k.*

Furthermore,

T2: *there is an uncountable set $S \subset J$ (containing no periodic points), which satisfies the following conditions:*

(A) *For every $p, q \in S$ with $p \ne q$,*

$$\limsup_{n\to\infty} |F^n(p) - F^n(q)| > 0 \tag{2.1}$$

and

$$\liminf_{n\to\infty} |F^n(p) - F^n(q)| = 0. \tag{2.2}$$

(B) *For every $p \in S$ and periodic point $q \in J$,*

$$\limsup_{n\to\infty} |F^n(p) - F^n(q)| > 0.$$

REMARKS. Notice that if there is a periodic point with period 3, then the hypothesis of the theorem will be satisfied.

An example of a function satisfying the hypotheses of the theorem is $F(x) = rx[1 - x/K]$ as in (1.2) for $r \in (3.84, 4]$ with $J = [0, K]$ and for $r > 4$, $F(x) = \max\{0, rx[1 - x/K]\}$ with $J = [0, K]$. See [2] for a detailed description of iterates of this function for $r \in [0, 4]$. The case $r = 4$ is discussed in [**6, 7, 12**].

While the existence of a point of period 3 implies the existence of one of period 5, the converse is false. (See Appendix 1).

We say $x \in J$ is **asymptotically periodic** if there is a periodic point p for which

$$F^n(x) - F^n(p) \to 0 \quad \text{as} \quad n \to \infty. \tag{2.3}$$

It follows from (B) that the set S contains no asymptotically periodic points. We remark that it is unknown what the infimum of r is for which the equation (1.2) has points which are not asymptotically periodic.

Proof of Theorem 1. The proof of T1 introduces the main ideas for both T1 and T2. We now give the proof of T1 with necessary lemmas and relegate the tedious proof of T2 to Appendix 2.

LEMMA 0. *Let $G: I \to R$ be continuous, where I is an interval. For any compact interval $I_1 \subset G(I)$ there is a compact interval $Q \subset I$ such that $G(Q) = I_1$.*

Proof. Let $I_1 = [G(p), G(q)]$, where $p, q \in I$. If $p < q$, let r be the last point of $[p, q]$ where $G(r) = G(p)$ and let s be the first point after r where $G(s) = G(q)$. Then $G([r, s]) = I_1$. Similar reasoning applies when $p > q$.

LEMMA 1. *Let $F: J \to J$ be continuous and let $\{I_n\}_{n=0}^{\infty}$ be a sequence of compact intervals with $I_n \subset J$ and $I_{n+1} \subset F(I_n)$ for all n. Then there is a sequence of compact intervals Q_n such that $Q_{n+1} \subset Q_n \subset I_0$ and $F^n(Q_n) = I_n$ for $n \ge 0$. For any $x \in Q = \cap Q_n$ we have $F^n(x) \in I_n$ for all n.*

Proof. Define $Q_0 = I_0$. Then $F^0(Q_0) = I_0$. If Q_{n-1} has been defined so that $F^{n-1}(Q_{n-1}) = I_{n-1}$, then $I_n \subset F(I_{n-1}) = F^n(Q_{n-1})$. By Lemma 1 applied to $G = F^n$ on Q_{n-1} there is a compact interval $Q_n \subset Q_{n-1}$ such that $F^n(Q_n) = I_n$. This completes the induction.

The technique of studying how certain sequences of sets are mapped into or onto each other is often used in studying dynamical systems. For instance, Smale uses this method in his famous "horseshoe example" in which he shows how a homeomorphism on the plane can have infinitely many periodic points [13].

LEMMA 2. *Let $G: J \to R$ be continuous. Let $I \subset J$ be a compact interval. Assume $I \subset G(I)$. Then there is a point $p \in I$ such that $G(p) = p$.*

Proof. Let $I = [\beta_0, \beta_1]$. Choose $\alpha_i (i = 0, 1)$ in I such that $G(\alpha_i) = \beta_i$. It follows $\alpha_0 - G(\alpha_0) \geqq 0$ and $\alpha_1 - G(\alpha_1) \leqq 0$ and so continuity implies $G(\beta) - \beta$ must be 0 for some β in I.

Assume $d \leq a < b < c$ as in the theorem. The proof for the case $d \geq a > b > c$ is similar and so is omitted. Write $K = [a, b]$ and $L = [b, c]$.

Proof of T1: Let k be a positive integer. For $k > 1$ let $\{I_n\}$ be the sequence of intervals $I_n = L$ for $n = 0, \cdots, k-2$ and $I_{k-1} = K$, and define I_n to be periodic inductively, $I_{n+k} = I_n$ for $n = 0, 1, 2, \cdots$. If $k = 1$, let $I_n = L$ for all n.

Let Q_n be the sets in the proof of Lemma 1. Then notice that $Q_k \subset Q_0$ and $F^k(Q_k) = Q_0$ and so by Lemma 2, $G = F^k$ has a fixed point p_k in Q_k. It is clear that p_k cannot have period less than k for F; otherwise we would need to have $F^{k-1}(p_k) = b$, contrary to $F^{k+1}(p_k) \in L$. The point p_k is a periodic point of period k for F.

3. Behavior near a periodic point. For some functions F the asymptotic behavior of iterates of a point can be understood simply by studying the periodic points. For

$$F(x) = ax(1 - x) \tag{3.1}$$

a detailed discussion of the points of period 1 and 2 may be found in [1] for $a \in [0, 4]$ and we now summarize some of those results. For $a \in [0, 4]$, $F: [0, 1] \to [0, 1]$.

For $a \in [0, 1]$, $x = 0$ is the only point of period 1; in fact, for $x \in [0, 1]$, the sequence $F^n(x) \to 0$ as $n \to \infty$.

For $a \in (1, 3]$, there are two points of period 1, namely 0 and $1 - a^{-1}$, and for $x \in (0, 1)$, $F^n(x) \to 1 - a^{-1}$ as $n \to \infty$.

For $a > 3$ there are also two points of period 2 which we may call p and q and of course $F(p) = q$ and $F(q) = p$. For $a \in (3, 1 + \sqrt{6} \approx 3.449)$ and $x \in (0, 1)$, $F^{2n}(x)$ converges to either p or q while $F^{2n+1}(x)$ converges to the other, except for those x for which there is an n for which $F^n(x)$ equals the point $1 - a^{-1}$ of period 1. There are only a countable number of such points so that the behavior of $\{F^n(x)\}$ can be understood by studying the periodic points.

For $a > 1 + \sqrt{6}$, there are 4 points of period 4 and for a slightly greater than $1 + \sqrt{6}$, $F^{4n}(x)$ tends to one of these 4 unless for some n, $F^n(x)$ equals one of the points of period 1 or 2. Therefore we may summarize this situation by saying that each point in $[0, 1]$ is asymptotically periodic.

For those values of a for which each point is asymptotically periodic, it is sufficient to study only the periodic points and their "stability properties." For any function F a point $y \in J$ with period k is said to be **asymptotically stable** if for some interval $I = (y - \delta, y + \delta)$ we have

$$|F^k(x) - y| < |x - y| \quad \text{for all} \quad x \in I.$$

If F is differentiable at the points $y, F(y), \cdots, F^{k-1}(y)$, there is a simple condition that will guarantee this behavior, namely

$$\left| \frac{d}{dx} F^k(x) \right| < 1.$$

By the chain rule

$$\frac{d}{dx}F^k(y) = \frac{d}{dx}F(F^{k-1}(y)) \cdot \frac{d}{dx}F^{k-1}(y)$$

$$= \frac{d}{dx}F(F^{k-1}(y)) \times \frac{d}{dx}F(F^{k-2}(y)) \times \cdots \times \frac{d}{dx}F(y) \tag{3.2}$$

$$= \prod_{n=0}^{k-1} \frac{d}{dx}F(y_n),$$

where y_n is the nth iterate, $F^n(y)$. Therefore y is asymptotically stable if

$$\left| \prod_{i=0}^{k-1} \frac{d}{dx}F(y_i) \right| < 1, \quad \text{where} \quad y_i = F^i(y).$$

This condition of course guarantees nothing about the limiting behavior of points which do not start "near" the periodic point or one of its iterates. The function in Figure 2 which was studied by Lorenz has the opposite behavior, namely, where the derivative exists we have

$$\left| \frac{d}{dx}F(x) \right| > 1.$$

For such a function every periodic point is "unstable" since for x near a periodic point y of period k, the kth iterate $F^k(x)$ is further from y than x is. To see this, approximate $F^k(x)$ by

$$F^k(y) + \frac{d}{dx}F^k(y)[y - x] = y + \frac{d}{dx}F^k(y)[y - x].$$

Thus for x near y, $|F^k(x) - y|$ is approximately $|x - y|\,|(d/dx)F^k(y)|$. From (3.2) $|(d/dx)F^k(y)|$ is greater than 1. Therefore $F^n(x)$ is further from y than x is.

We do not know when values of a begin to occur for which F in (3.1) has points which are not asymptotically periodic. For $a = 3.627$, F has a periodic point (which is asymptotically stable) of period 6 (approx. $x = .498$). This x is therefore a point of period 3 for F^2 and so Theorem 1 may be applied to F^2. Since F^2 has points which are not asymptotically periodic, the same is true of F.

In order to contrast the situations in this section with other possible situations discussed in the next section, we define the limit set of a point x. The point y is a **limit point** of a sequence $\{x_n\} \subset J$ if there is a subsequence x_{n_i} converging to y. The **limit set** $L(x)$ is defined to be the set of limit points of $\{F^n(x)\}$. If x is asymptotically periodic, then $L(x)$ is the set $\{y, F(y), \cdots, F^{k-1}(y)\}$ for some periodic point y of period k.

4. Statistical properties of $\{F^n(x)\}$. Theorem 1 establishes the irregularity of the behavior of iterates of points. What is also needed is a description of the regular behavior of the sequence $\{F^n(x)\}$ when F is piecewise continuously differentiable (as is Lorenz's function in Figure 2) and

$$\inf_{x \in J_1} \left| \frac{dF}{dx} \right| > 1 \quad \text{where} \quad J_1 = \left\{ x : \frac{dF}{dx} \text{ exists} \right\}. \tag{4.1}$$

One approach to describing the asymptotic behavior for such functions is to describe $L(x)$, if possible. A second approach, which turns out to be related, is to examine the average behavior of $\{F^n(x)\}$. The fraction of the iterates $\{x, \cdots, F^{N-1}(x)\}$ of x that are in $[a_1, a_2]$ will be denoted by $\phi(x, N, [a_1, a_2])$. The limiting fraction will be denoted

$$\phi(x, [a_1, a_2]) = \lim_{N \to \infty} \phi(x, N, [a_1, a_2])$$

when the limit exists. The subject of ergodic theory, which studies transformation on general spaces,

motivates the following definition. We say g is the **density of** x (for F) if the limiting fraction satisfies

$$\phi(x,[a_1,a_2]) = \int_{a_1}^{a_2} g(x)dx \quad \text{for all} \quad a_1, a_2 \in J; \quad a_1 \le a_2.$$

The techniques for the study of densities use non-elementary techniques of measure theory and functional analysis, so that we shall only summarize the results. But their value lies in the fact that for certain F almost all $x \in I$ have the same density. Until recently the existence of such densities had not been proved, except for the simplest of functions F. The following result has recently been proved:

THEOREM 2. **[5]**. *Let $F: I \to I$ satisfy the following conditions*:
1) *F is continuous.*
2) *Except at one point $t \in I$, F is twice continuously differentiable.*
3) *F satisfies* (4.1).
Then there exists a function $g: I \to [0,\infty)$, such that for almost all $x \in I$, g is the density of x. Also for almost all $x \in I$, $L(x) = \{y: g(y) > 0\}$ which is an interval. Moreover, the set $I_\infty = \{y: g(y) > 0\}$ is an interval, and $L(x) = I_\infty$ for almost all x.

The proof makes use of results in **[8]**. The problem of computationally finding the density is solved in **[9]**.

A detailed discussion of (3.1) is given in **[16]**, describing how $L(x)$ varies as the parameter a in (3.1) varies between 3.0 and 4.0.

A major question left unsolved is whether (for some nice class of functions F) the existence of a stable periodic point implies that almost every point is asymptotically periodic.

Appendix 1: Period 5 does not imply period 3. In this Appendix we give an example which has a fixed point of period 5 but no fixed point of period 3.

Let $F: [1,5] \to [1,5]$, be defined such that $F(1) = 3$, $F(2) = 5$, $F(3) = 4$, $F(4) = 2$, $F(5) = 1$ and on each interval $[n, n+1]$, $1 \le n \le 4$, assume F is linear. Then

$$F^3([1,2]) = F^2([3,5]) = F([1,4]) = [2,5].$$

Hence, F^3 has no fixed points in $[1,2]$. Similarly, $F^3([2,3]) = [3,5]$ and $F^3([4,5]) = [1,4]$, so neither of these intervals contains a fixed point of F^3. On the other hand,

$$F^3([3,4]) = F^2([2,4]) = F([2,5]) = [1,5] \supset [3,4].$$

Hence, F^3 must have a fixed point in $[3,4]$. We shall now demonstrate that the fixed point of F^3 is unique and is also a fixed point of F.

Let $p \in [3,4]$ be a fixed point of F^3. Then $F(p) \in [2,4]$. If $F(p) \in [2,3]$, then $F^3(p)$ would be in $[1,2]$ which is impossible since then p could not be a fixed point. Hence $F(p) \in [3,4]$ and $F^2(p) \in [2,4]$. If $F^2(p) \in [2,3]$ we would have $F^3(p) \in [4,5]$, an impossibility. Hence p, $F(p)$, $F^2(p)$ are all in $[3,4]$. On the interval $[3,4]$, F is defined linearly and so $F(x) = 10 - 2x$. It has a fixed point $10/3$ and it is easy to see that F^3 has a unique fixed point, which must be $10/3$. Hence there is no point of period 3.

Appendix 2. Proof of T2 of Theorem 1. Let $\mathcal{M}$ be the set of sequences $M = \{M_n\}_{n=1}^{\infty}$ of intervals with

(A.1) $$M_n = K \quad \text{or} \quad M_n \subset L, \quad \text{and} \quad F(M_n) \supset M_{n+1}$$

if $M_n = K$ then

(A.2) $$n \text{ is the square of an integer and } M_{n+1}, M_{n+2} \subset L,$$

where $K = [a, b]$ and $L = [b, c]$. Of course if n is the square of an integer, then $n+1$ and $n+2$ are not, so the last requirement in (A.2) is redundant. For $M \in \mathcal{M}$, let $P(M, n)$ denote the number of i's in $\{1, \cdots, n\}$ for which $M_i = K$. For each $r \in (3/4, 1)$ choose $M^r = \{M_n^r\}_{n=1}^{\infty}$ to be a sequence in $\mathcal{M}$ such that

$$\lim_{n\to\infty} P(M^r, n^2)/n = r. \tag{A.3}$$

Let $\mathcal{M}_0 = \{M^r : r \in (3/4, 1)\} \subset \mathcal{M}$. Then $\mathcal{M}_0$ is uncountable since $M^{r_1} \neq M^{r_2}$ for $r_1 \neq r_2$. For each $M^r \in \mathcal{M}_0$, by Lemma 1, there exists a point x_r with $F^n(x_r) \in M_n^r$ for all n. Let $S = \{x_r : r \in (3/4, 1)\}$. Then S is also uncountable. For $x \in S$, let $P(x, n)$ denote the number of i's in $\{1, \cdots, n\}$ for which $F^i(x) \in K$. We can never have $F^k(x_r) = b$, because then x_r would eventually have period 3, contrary to (A.2). Consequently $P(x_r, n) = P(M^r, n)$ for all n, and so

$$\rho(x_r) = \lim_{n\to\infty} P(X_r, n^2) = r$$

for all r. We claim that

(A.4) for $p, q \in S$, with $p \neq q$, there exist infinitely many n's such that $F^n(p) \in K$ and $F^n(q) \in L$ or vice versa.

We may assume $\rho(p) > \rho(q)$. Then $P(p, n) - P(q, n) \to \infty$, and so there must be infintely many n's such that $F^n(p) \in K$ and $F^n(q) \in L$.

Since $F^2(b) = d \leqq a$ and F^2 is continuous, there exists $\delta > 0$ such that $F^2(x) < (b + d)/2$ for all $x \in [b - \delta, b] \subset K$. If $p \in S$ and $F^n(p) \in K$, then (A.2) implies $F^{n+1}(p) \in L$ and $F^{n+2}(p) \in L$. Therefore $F^n(p) < b - \delta$. If $F^n(q) \in L$, then $F^n(q) \geqq b$, so

$$|F^n(p) - F^n(q)| > \delta.$$

By claim (A.4), for any $p, q \in S$, $p \neq q$, it follows

$$\limsup_{n\to\infty} |F^n(p) - F^n(q)| \geqq \delta > 0.$$

Hence (2.1) is proved. This technique may be similarly used to prove (B) is satisfied.

Proof of 2.2. Since $F(b) = c$, $F(c) = d \leqq a$, we may choose intervals $[b^n, c^n]$, $n = 0, 1, 2, \cdots$, such that

(a) $[b, c] = [b^0, c^0] \supset [b^1, c^1] \supset \cdots \supset [b^n, c^n] \supset \cdots$,

(b) $F(x) \in (b^n, c^n)$ for all $x \in (b^{n+1}, c^{n+1})$,

(c) $F(b^{n+1}) = c^n$, $F(c^{n+1}) = b^n$.

Let $A = \bigcap_{n=0}^{\infty} [b^n, c^n]$, $b^* = \inf A$ and $c^* = \sup A$, then $F(b^*) = c^*$ and $F(c^*) = b^*$, because of (c).

In order to prove (2.2) we must be more specific in our choice of the sequences M^r. In addition to our previous requirements on $M \in M$, we will assume that if $M_k = K$ for both $k = n^2$ and $(n+1)^2$ then $M_k = [b^{2n-(2j-1)}, b^*]$ for $k = n^2 + (2j-1)$, $M_k = [c^*, c^{2n-2j}]$ for $k = n^2 + 2j$ where $j = 1, \cdots, n$. For the remaining k's which are not squares of integers, we assume $M_k = L$.

It is easy to check that these requirements are consistent with (A.1) and (A.2), and that we can still choose M^r so as to satisfy (A.3). From the fact that $\rho(x)$ may be thought of as the limit of the fraction of n's for which $F^{n^2}(x) \in K$, it follows that for any $r^*, r \in (3/4, 1)$ there exist infinitely many n such that $M_k^r = M_k^{r^*} = K$ for both $k = n^2$ and $(n+1)^2$. To show (2.2), let $x_r \in S$ and $x_{r^*} \in S$. Since $b^n \to b^*$, $c^n \to c^*$ as $n \to \infty$, for any $\varepsilon > 0$ there exists N with $|b^n - b^*| < \varepsilon/2$, $|c^n - c^*| < \varepsilon/2$ for all $n > N$. Then, for any n with $n > N$ and $M_k^r = M_k^{r^*} = K$ for both $k = n^2$ and $(n+1)^2$, we have

$$F^{n^2+1}(x_r) \in M_k^r = [b^{2n-1}, b^*]$$

with $k = n^2 + 1$ and $F^{n^2+1}(x_r)$ and $F^{n^2+1}(x_{r*})$ both belong to $[b^{2n-1}, b^*]$. Therefore, $|F^{n^2+1}(x_r) - F^{n^2+1}(x_{r*})| < \varepsilon$. Since there are infinitely many n with this property, $\liminf_{n\to\infty} |F^n(x_r) - F^n(x_{r*})| = 0$. □

REMARK. The theorem can be generalized by assuming that $F: J \to R$ without assuming that $F(J) \subset J$ and we leave this proof to the reader. Of course $F(J) \cap J$ would be nonempty since it would contain the points a, b, and c, assuming that b, c, and d are defined.

Research partially supported by National Science Foundation grant GP-31386X.

References

1. E. N. Lorenz, The problem of deducing the climate from the governing equations, Tellus, 16 (1964) 1–11.
2. ———, Deterministic nonperiodic flows, J. Atmospheric Sci., 20 (1963) 130–141.
3. ———, The mechanics of vacillation, J. Atmospheric Sci., 20 (1963) 448–464.
4. ———, The predictability of hydrodynamic flow, Trans. N.Y. Acad. Sci., Ser. II, 25 (1963) 409–432.
5. T. Y. Li and J. A. Yorke, Ergodic transformations from an interval into itself, (submitted for publication).
6. P. R. Stein and S. M. Ulam, Nonlinear transformation studies on electronic computers, Los Alamos Sci. Lab., Los Alamos, New Mexico, 1963.
7. S. M. Ulam, A Collection of Mathematical Problems, Interscience, New York, 1960, p. 150.
8. A. Lasota and J. A. Yorke, On the existence of invariant measures for piecewise monotonic transformations, Trans. Amer. Math. Soc., 186 (1973) 481–488.
9. T. Y. Li, Finite approximation for the Frobenius-Perron operator — A Solution to Ulam's conjecture, (submitted for publication).
10. Syunro Utida, Population fluctuation, an experimental and theoretical approach, Cold Spring Harbor Symposia on Quantitative Biology, 22 (1957) 139–151.
11. A. Lasota and P. Rusek, Problems of the stability of the motion in the process of rotary drilling with cogged bits, Archium Gornictua, 15 (1970) 205–216 (Polish with Russian and German Summary).
12. N. Metropolis, M. L. Stein and P. R. Stein, On infinite limit sets for transformations on the unit interval, J. Combinatorial Theory Ser. A 15, (1973) 25–44.
13. S. Smale, Differentiable dynamical systems, Bull. A.M.S., 73 (1967) 747–817 (see §1.5).
14. G. Oster and Y. Takahashi, Models for age specific interactions in a periodic environment, Ecology, in press.
15. D. Auslander, G. Oster and C. Huffaker, Dynamics of interacting populations, a preprint.
16. T. Y. Li and James A. Yorke, The "simplest" dynamics system, (to appear).
17. R. M. May, Biological populations obeying difference equations, stable cycles, and chaos, J. Theor. Biol. (to appear).

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF UTAH, SALT LAKE CITY, UT 84112.
INSTITUTE FOR FLUID DYNAMICS AND APPLIED MATHEMATICS, UNIVERSITY OF MARYLAND, COLLEGE PARK, MD 20742.

Appendix C

Additional Results

C.1 Effect of Size of Sliding Window

For all our experiments we used sliding window of size 1200×1200 for creating sub-images. To check if ScanSSD performs better with bigger sliding window that gives more context in each sub-image we increase the sliding window size to 1800×1800. With bigger sliding window we get 122,140 and 50,845 sub-images with math regions for training and testing, respectively.

Figure C.1 shows the results using window size of 1800×1800 in comparison with window size of 1200×1200. In this experiment, we used RIT2 system which was submitted to ICDAR 2019. RIT2 system used SSD512 with aspect ratios $a_r = \{\{1, 2, 3, 5\}, \{1, 2, 3, 5, 7\}, \{1, 2, 3, 5, 7\}, \{1, 2, 3\}, \{1, 2, 3\}, \{1, 2\}, \{1, 2\}\}$ where i^{th} member of a_r represents aspect ratios used in the i^{th} feature map. For 1200×1200 we used sum score algorithm with threshold of 20 for pooling and for 1800×1800 we used sum score algorithm with threshold of 1 for pooling. Thresholds were decided by the grid search.

We found that window size of 1200×1200 performs better than 1800×1800 for both IOU50 and IOU75. There can be two main reasons for window size 1800×1800 to perform worse. First, use of bigger window size results in

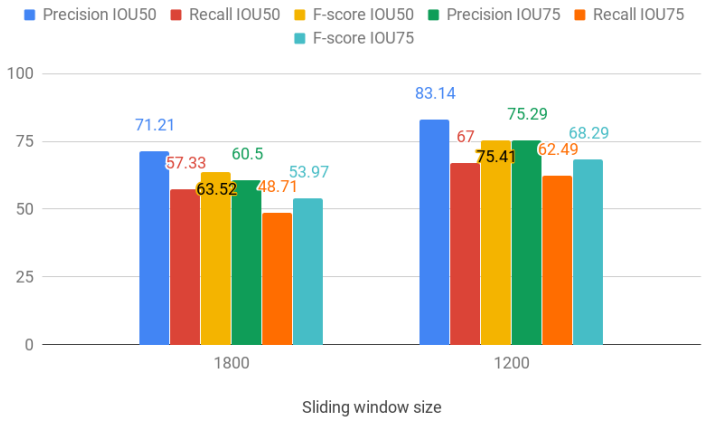


Figure C.1: Comparison of sliding window size of 1800 × 1800 with sliding window size of 1200 × 1200.

less number of images for training. Second, Liu et al. [38] discussed that SSD performs better while detecting large objects than small objects. Sub-images generated using sliding window of size 1800 × 1800 have smaller math regions as compared sub-images generated using sliding window of size 1200 × 1200. Hence, SSD might be performing better while detecting math regions in sub-images generated using 1200 × 1200.

In future we would like to use the validation dataset to perform grid search on the sliding window size and find the optimal size for the sliding window.

Bibliography

[1] Robert H Anderson. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, pages 436–459. ACM, 1967.

[2] Sumit Bhatia, Prasenjit Mitra, and C Lee Giles. Finding algorithms in scientific articles. In *Proceedings of the 19th international conference on World wide web*, pages 1061–1062. ACM, 2010.

[3] Sumit Bhatia, Suppawong Tuarob, Prasenjit Mitra, and C Lee Giles. An algorithm search engine for software developers. In *Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 13–16. ACM, 2011.

[4] BB Chaudhuri and Utpal Garain. An approach for processing mathematical expressions in printed document. In *International Workshop on Document Analysis Systems*, pages 310–321. Springer, 1998.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[6] Wei-Ta Chu and Fan Liu. Mathematical formula detection in heterogeneous document images. In *2013 Conference on Technologies and Applications of Artificial Intelligence*, pages 140–145. IEEE, 2013.

[7] Alireza Darvishy, Mark Nevill, and Hans-Peter Hutter. Automatic paragraph detection for accessible pdf documents. In *International Conference on Computers Helping People with Special Needs*, pages 367–372. Springer, 2016.

[8] Hervé Déjean and Jean-Luc Meunier. A system for converting pdf documents into structured xml format. In *International Workshop on Document Analysis Systems*, pages 129–140. Springer, 2006.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Derek M Drake and Henry S Baird. Distinguishing mathematics notation from english text using computational geometry. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1270–1274. IEEE, 2005.

[11] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[12] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. *CoRR*, abs/1312.2249, 2013.

[13] Robert P Futrelle, Mingyan Shao, Chris Cieslik, and Andrea Elaina Grimes. Extraction, layout analysis and classification of diagrams in pdf documents. In *null*, page 1007. IEEE, 2003.

[14] Liangcai Gao, Xiaohan Yi, Zhuoren Jiang, Leipeng Hao, and Zhi Tang. Icdar2017 competition on page object detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1417–1422. IEEE, 2017.

[15] Liangcai Gao, Xiaohan Yi, Yuan Liao, Zhuoren Jiang, Zuoyu Yan, and Zhi Tang. A deep learning-based formula detection method for pdf documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 553–558. IEEE, 2017.

[16] Utpal Garain and BB Chaudhuri. A syntactic approach for processing mathematical expressions in printed documents. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 523–526. IEEE, 2000.

[17] Utpal Garain and Bidyut B Chaudhuri. Ocr of printed mathematical expressions. In *Digital Document Processing*, pages 235–259. Springer, 2007.

[18] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. Table detection using deep learning. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 771–776. IEEE, 2017.

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

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

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] Leipeng Hao, Liangcai Gao, Xiaohan Yi, and Zhi Tang. A table detection method for pdf documents based on convolutional neural networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 287–292. IEEE, 2016.

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[24] Dafang He, Scott Cohen, Brian Price, Daniel Kifer, and C Lee Giles. Multi-scale multi-task fcn for semantic page segmentation and table detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 254–261. IEEE, 2017.

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

[26] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[27] Kenichi Iwatsuki, Takeshi Sagara, Tadayoshi Hara, and Akiko Aizawa. Detecting in-line mathematical expressions in scientific documents. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, pages 141–144. ACM, 2017.

[28] Afef Kacem, Abdel Belaïd, and M Ben Ahmed. Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. *International Journal on Document Analysis and Recognition*, 4(2):97–108, 2001.

[29] Hsi-Jian Lee and Jiumn-Shine Wang. Design of a mathematical expression recognition system. In *Proceedings of 3rd International Confer-*

ence on Document analysis and Recognition, volume 2, pages 1084–1087. IEEE, 1995.

[30] Sida Li, Liangcai Gao, Zhi Tang, and Yinyan Yu. Cross-reference identification within a pdf document. In *Document Recognition and Retrieval XXII*, volume 9402, page 940209. International Society for Optics and Photonics, 2015.

[31] Xiao-Hui Li, Fei Yin, and Cheng-Lin Liu. Page object detection from pdf document images by deep structured prediction and supervised clustering. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3627–3632. IEEE, 2018.

[32] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[34] Xiaofan Lin. Header and footer extraction by page association. In *Document Recognition and Retrieval X*, volume 5010, pages 164–172. International Society for Optics and Photonics, 2003.

[35] Xiaoyan Lin, Liangcai Gao, Zhi Tang, Josef Baker, Mohamed Alkalai, and Volker Sorge. A text line detection method for mathematical for-

mula recognition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 339–343. IEEE, 2013.

[36] Xiaoyan Lin, Liangcai Gao, Zhi Tang, Xuan Hu, and Xiaofan Lin. Identification of embedded mathematical formulas in pdf documents using svm. In *Document Recognition and Retrieval XIX*, volume 8297, page 82970D. International Society for Optics and Photonics, 2012.

[37] Xiaoyan Lin, Liangcai Gao, Zhi Tang, Xiaofan Lin, and Xuan Hu. Mathematical formula identification in pdf documents. In *2011 International Conference on Document Analysis and Recognition*, pages 1419–1423. IEEE, 2011.

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

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[40] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, and Utpal Garain. Icdar 2019 crohme + tfd: Competition on recognition of handwritten

mathematical expressions and typeset formula detection. In *15th International Conference on Document Analysis and Recognition (ICDAR 2019)*. IEEE, 2019.

[41] J-L Meunier. Optimized xy-cut for determining a page reading order. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 347–351. IEEE, 2005.

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

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

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

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

[47] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1162–1167. IEEE, 2017.

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

[49] J-Y Toumit, Sonia Garcia-Salicetti, and Hubert Emptoz. A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, pages 119–122. IEEE, 1999.

[50] Suppawong Tuarob, Sumit Bhatia, Prasenjit Mitra, and C Lee Giles. Automatic detection of pseudocodes in scholarly documents using machine learning. In *2013 12th International Conference on Document Analysis and Recognition*, pages 738–742. IEEE, 2013.

[51] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[52] Xing Wang and Jyh-Charn Liu. A font setting based bayesian model

to extract mathematical expression in pdf files. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 759–764. IEEE, 2017.

[53] Xiaohan Yi, Liangcai Gao, Yuan Liao, Xiaode Zhang, Runtao Liu, and Zhuoren Jiang. Cnn based page object detection in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 230–235. IEEE, 2017.

[54] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[55] Zheng Yu, Shujing Lyu, Yue Lu, and Patrick SP Wang. A fusion strategy for the single shot text detector. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3687–3691. IEEE, 2018.

[56] Richard Zanibbi, Dorothea Blostein, and James R. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(11):1455–1467, 2002.